



Protocoles Sécurisés et Analyse des Vulnérabilités

Introduction et Contexte

Définitions Fondamentales



Protocole Sécurisé

Un protocole sécurisé est un ensemble de règles et de procédures permettant l'échange d'informations entre entités tout en garantissant les propriétés de sécurité fondamentales : confidentialité, intégrité, authentification et non-répudiation.



Cryptographie Appliquée

Utilisation pratique des algorithmes cryptographiques dans des systèmes réels pour résoudre des problèmes de sécurité concrets, contrairement à la cryptographie théorique qui se concentre sur les preuves mathématiques.



Vulnérabilité (CVE)

Faible de sécurité dans un logiciel, un protocole ou un système qui peut être exploitée par un attaquant pour compromettre la sécurité. Les vulnérabilités sont référencées dans la base CVE (Common Vulnerabilities and Exposures).



Enjeux Actuels

DIGITAL SECURITY

Évolution des Menaces

Les cyberattaques évoluent constamment, exploitant :

- Les vulnérabilités zero-day
- Les erreurs de configuration
- L'ingénierie sociale
- Les chaînes d'approvisionnement

Cadre Réglementaire

Conformité



RGPD

Protection des données personnelles



NIS2

Cybersécurité des infrastructures critiques



ISO 27001

Système de management de la sécurité



ANSSI

Recommandations françaises de cybersécurité



Fondements Théoriques

Modèle OSI et Sécurité

Sécurité par Couches

Couche OSI	Protocoles Sécurisés	Menaces Principales
Application (7)	HTTPS, S/MIME, PGP	Injection, XSS, CSRF
Présentation (6)	TLS, SSL	Chiffrement faible
Session (5)	SSH, Kerberos	Hijacking, MITM
Transport (4)	TLS, DTLS	Déni de service
Réseau (3)	IPsec, OSPF-MD5	Spoofing, routage
Liaison (2)	802.1X, WPA3	ARP poisoning
Physique (1)	-	Écoute passive

Principe de Défense en Profondeur

Stratégie consistant à superposer plusieurs mécanismes de sécurité à différents niveaux pour qu'une faille dans un mécanisme ne compromette pas l'ensemble du système.

Propriétés de Sécurité



Confidentialité

Définition : Garantie que l'information n'est accessible qu'aux entités autorisées.

Mécanismes :

- Chiffrement symétrique (AES)
- Chiffrement asymétrique (RSA, ECC)
- Gestion des clés (PKI)

Exemple : Le chiffrement TLS empêche l'interception des données bancaires lors d'une transaction en ligne.



Intégrité

Définition : Assurance que l'information n'a pas été modifiée de manière non autorisée.

Mécanismes :

- Fonctions de hachage (SHA-256)
- Codes d'authentification (HMAC)
- Signatures numériques

Exemple : Le hash SHA-256 d'un fichier permet de vérifier qu'il n'a pas été corrompu.

Propriétés de Sécurité



Authentication

Définition : Vérification de l'identité d'une entité (utilisateur, serveur, appareil).

Types :

- **Facteur de connaissance :** mot de passe, PIN
- **Facteur de possession :** carte, token, smartphone
- **Facteur d'être :** biométrie

Exemple : L'authentification SSH par clé publique/privée.



Non-Répudiation

Définition : Impossibilité pour une entité de nier avoir effectué une action.

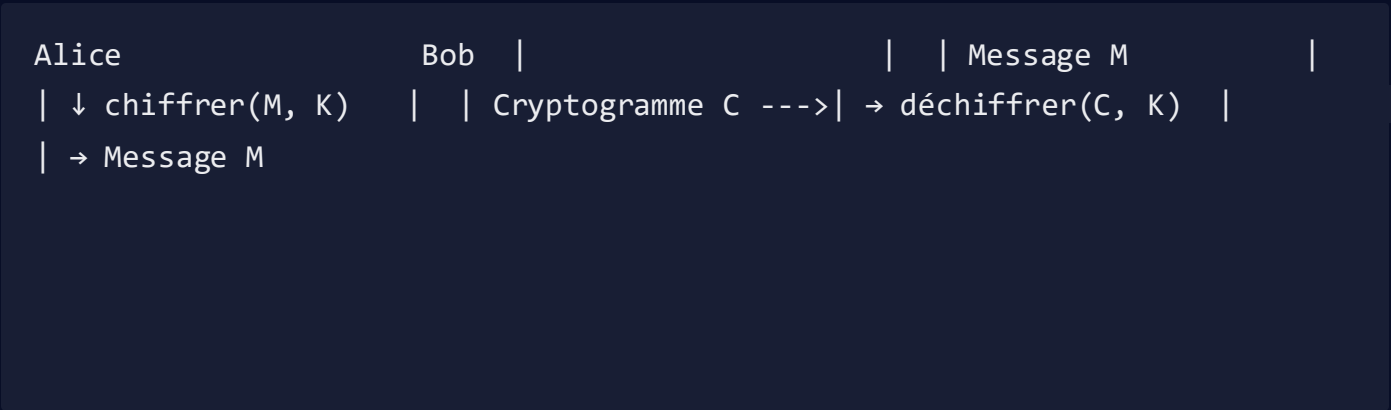
Mécanismes :

- Signatures numériques
- Horodatage sécurisé
- Journalisation

Cryptographie Moderne

Chiffrement Symétrique

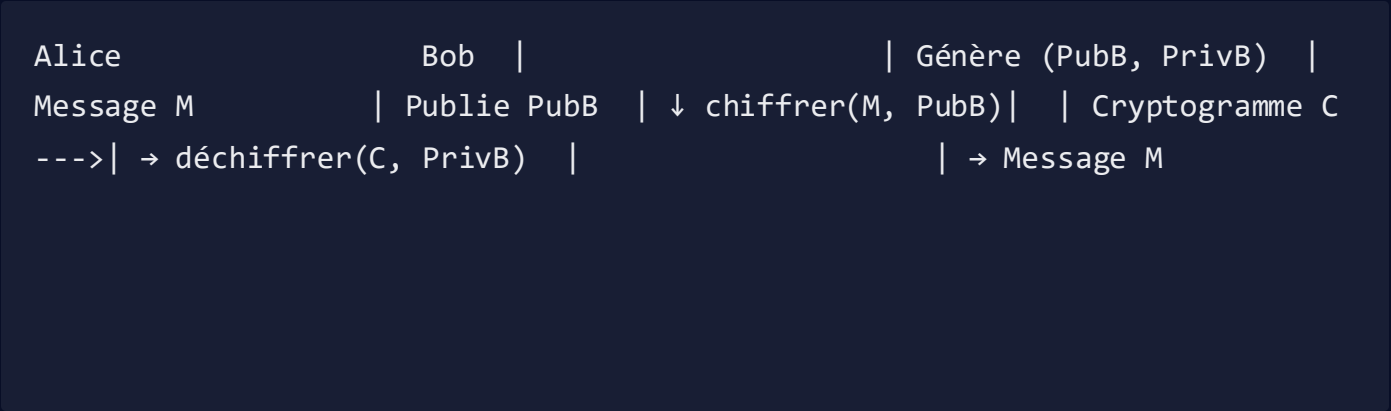
Principe : Une même clé secrète K sert au chiffrement et au déchiffrement.



- Algorithmes Standards :**
- **AES (Advanced Encryption Standard)** : 128, 192, 256 bits
 - **ChaCha20** : Alternative moderne et rapide
 - **Obsolètes** : DES, 3DES (à éviter)

Chiffrement Asymétrique

Principe : Deux clés mathématiquement liées – publique et privée.



- Algorithmes Standards :**
- **RSA** : 2048+ bits (factorisation)
 - **ECC** : 256+ bits (logarithme discret)
 - **Ed25519** : Courbes elliptiques modernes

Fonctions de Hachage

Propriétés Requises

- | | |
|--|--|
| <div>1</div> Déterministe
Même entrée → même sortie | <div>2</div> Unidirectionnelle
Impossible d'inverser |
| <div>3</div> Résistante aux collisions
Difficile de trouver $x \neq y$ avec $h(x) = h(y)$ | <div>4</div> Effet avalanche
Petit changement → grande différence |

Algorithmes Standards

Recommandés

- **SHA-256/SHA-3** : sûrs actuellement

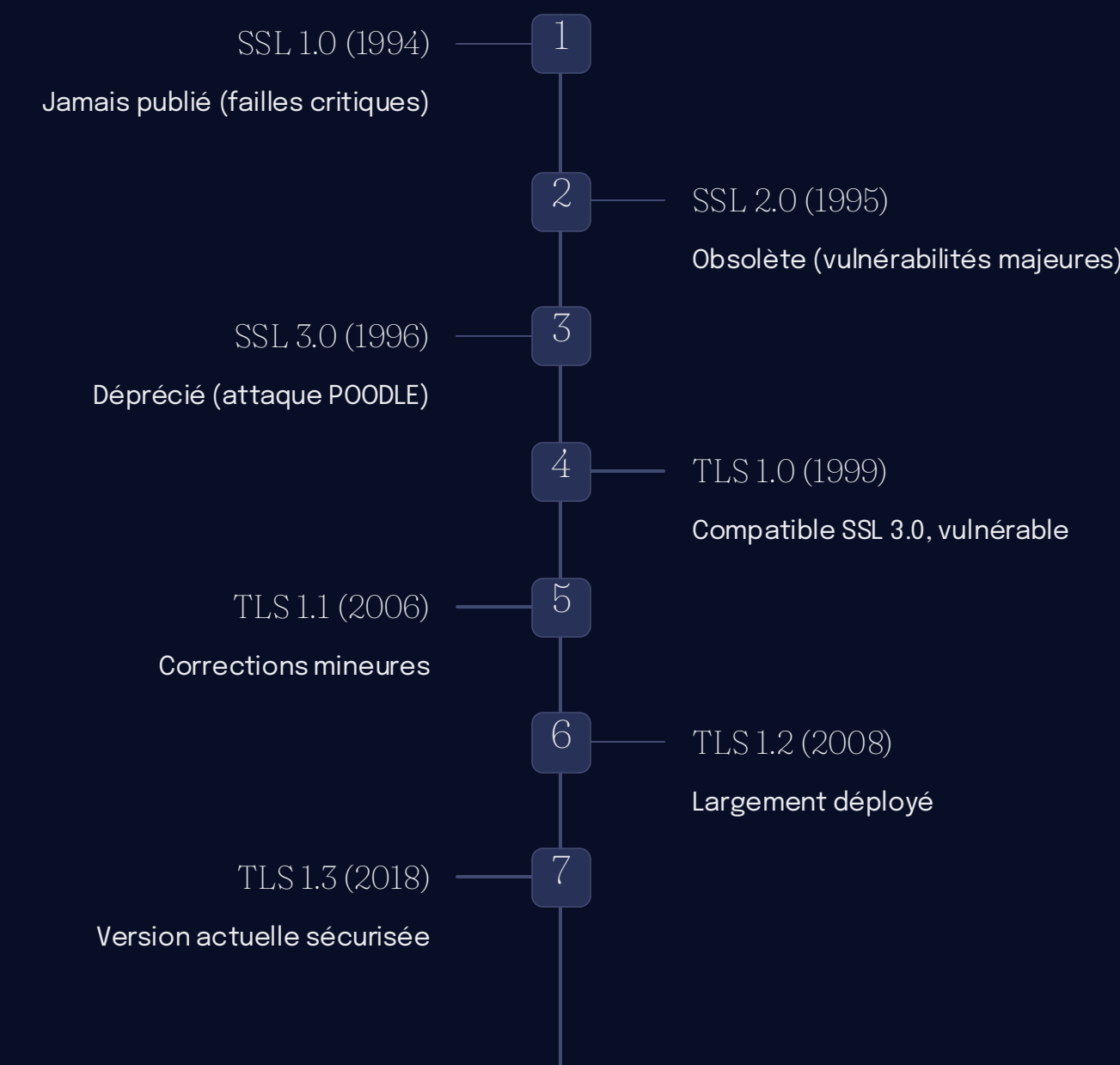
Déconseillés

- **SHA-1** : déprécié (collisions trouvées)
- **MD5** : obsolète (ne plus utiliser)

TLS/SSL - Transport Layer Security

Architecture et Évolution

Historique des Versions



Architecture TLS

TLS se compose de plusieurs sous-protocoles :

Handshake Protocol

Négociation des paramètres

Record Protocol

Chiffrement des données

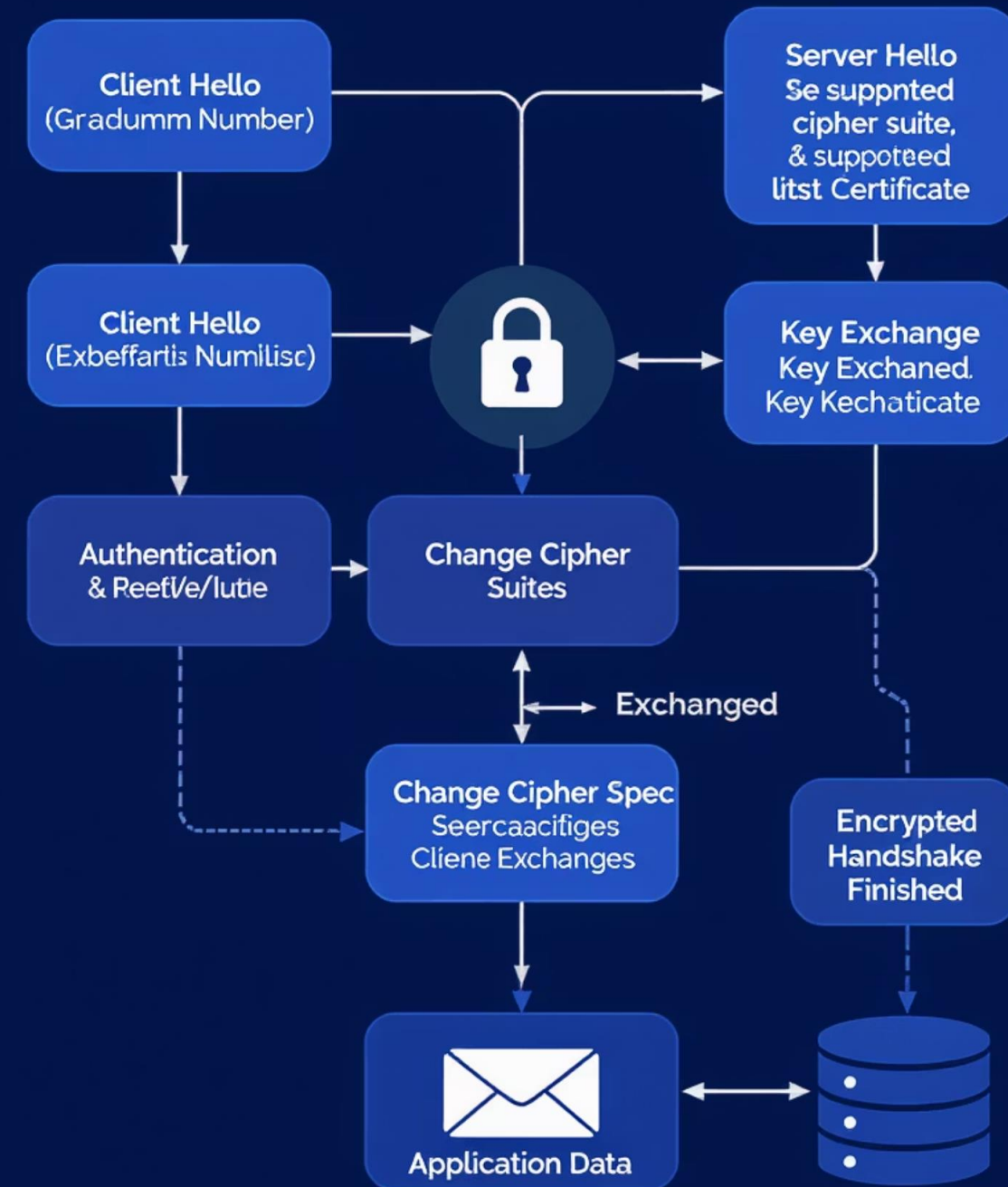
Alert Protocol

Gestion des erreurs

Change Cipher Spec

Activation du chiffrement

TLS Protocol Architecture



Handshake TLS 1.3

Phase 1 : Client Hello

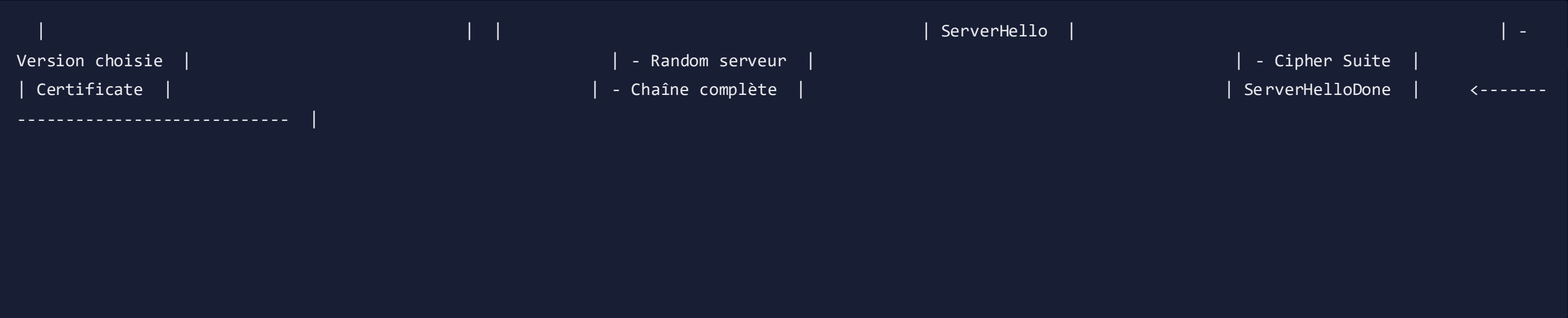
```
Client          Server  |                               | | ClientHello
| | - Version TLS supportées      | | - Random (32 bytes)         | | - Cipher Suites
| | - Extensions (SNI, ALPN...)   | | ----->                   |
```

Contenu Détaillé

- **Protocol Version** : TLS 1.3 (0x0304)
- **Random** : Nonce de 32 octets (protection replay)
- **Cipher Suites** : Liste des algorithmes supportés
- **Extensions** :
 - **SNI** : Server Name Indication (hébergement mutualisé)
 - **ALPN** : Application Layer Protocol Negotiation
 - **Key Share** : Clés publiques pour l'échange

Handshake TLS 1.3

Phase 2 : Server Hello + Certificat



Phase 3 : Vérification Certificat

Le client vérifie :

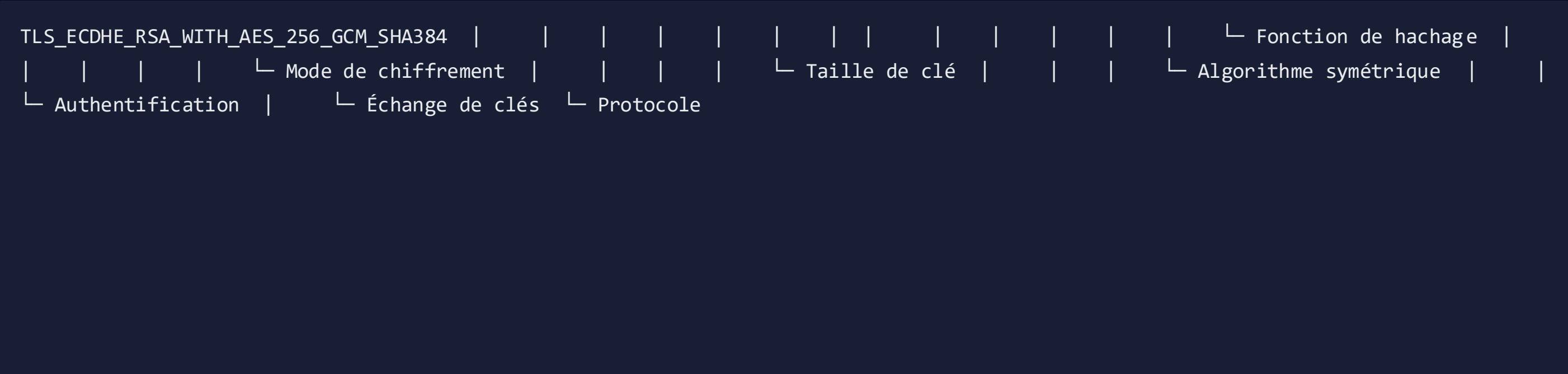
1. **Validité temporelle** : $\text{Not Before} \leq \text{Now} \leq \text{Not After}$
2. **Chaîne de confiance** : jusqu'à une CA racine
3. **Nom de domaine** : correspondance avec SNI
4. **Révocation** : consultation CRL/OCSP

Phase 4 : Génération Clés de Session

Master Secret = HKDF-Extract-Expand(Pre-Master Secret, "master secret", Client.Random + Server.Random)

Cipher

Format Standard



Recommandations 2024

Autorisées

- `TLS_AES_256_GCM_SHA384` (TLS 1.3)
- `TLS_CHACHA20_POLY1305_SHA256` (TLS 1.3)
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384` (TLS 1.2)

Interdites

- Toutes les suites avec RC4, MD5, SHA-1
- Chiffrement DES, 3DES
- Échange de clés statique (sans PFS)

Perfect Forward Secrecy (PFS)

Concept

Propriété garantissant que la compromission d'une clé privée à long terme ne permet pas de déchiffrer les communications passées.

Mécanisme

Session 1 : Clé éphémère E1 → Détruite après usage
Session 2 : Clé éphémère E2 → Détruite après usage...
Même si la clé serveur est compromise, E1 et E2 ne peuvent pas être retrouvées.

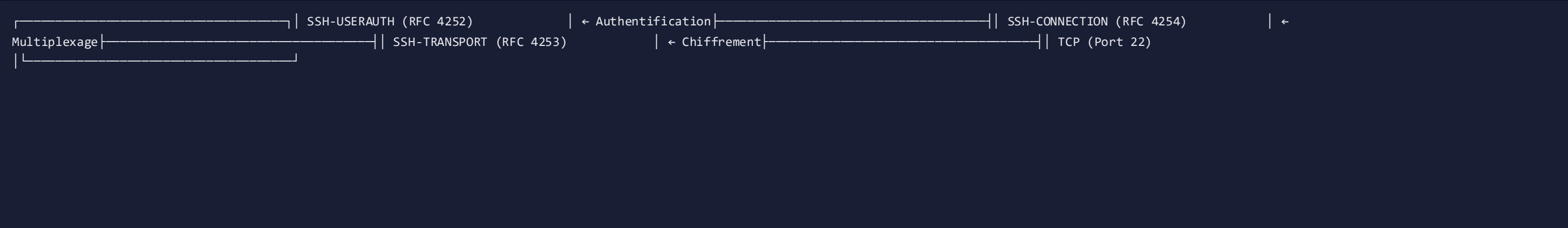
Implémentation

- **ECDHE** : Elliptic Curve Diffie-Hellman Ephemeral
- **DHE** : Diffie-Hellman Ephemeral
- **RSA statique** : ❌ Pas de PFS

SSH - Secure Shell

Architecture du Protocole

Couches SSH



Phases de Connexion



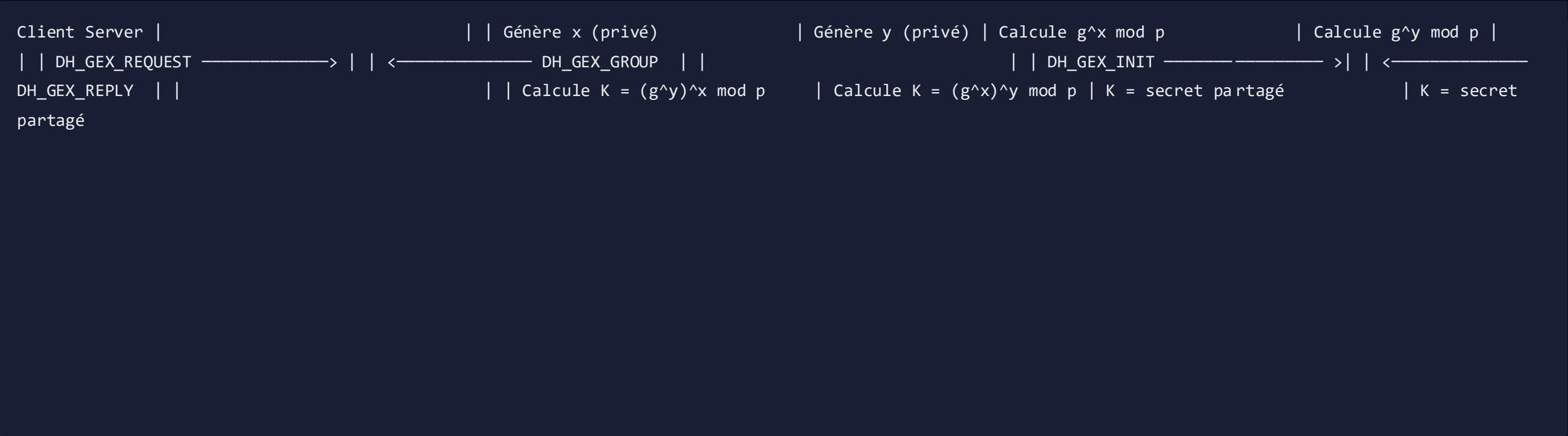
Échange de Clés (Key Exchange)

Algorithmes Disponibles

Kex Algorithms:

- curve25519-sha256 (Recommandé)
- diffie-hellman-group16-sha512 (Acceptable)
- diffie-hellman-group14-sha256 (Minimum)
- diffie-hellman-group1-sha1 (Obsolète)

Processus Diffie-Hellman



Authentication

Méthodes Standard

1. Authentication par Mot de Passe

```
Client                               Server |                               | | SSH_MSG_USERAUTH_REQUEST
| | - username: alice                | | - password: *****
| |                                  | Vérifie /etc/shadow |
SSH_MSG_USERAUTH_SUCCESS             |                               |
<-----| |
```

Vulnérabilités :

- Attaques par dictionnaire
- Brute force
- Keyloggers
- Transmission du hash

Authentication SSH

Authentication par Clé Publique



Avantages :

- Pas de transmission de secret
- Résistant au bruteforce
- Support des certificats
- Révocation possible

Génération de Clés Sécurisées

```
# Ed25519 (recommandé - modern, rapide, sûr)ssh-keygen -t ed25519 -C "alice@entreprise.com"# RSA 4096 bits (acceptable si Ed25519 non supporté)ssh-keygen -t rsa -b 4096 -C "alice@entreprise.com"# ECDSA P-256 (acceptable)ssh-keygen -t ecdsa -b 256 -C "alice@entreprise.com"
```

Algorithmes Cryptographiques

Chiffrement (Ciphers)

Recommandés

- `chacha20-poly1305@openssh.com`
- `aes256-gcm@openssh.com`
- `aes128-gcm@openssh.com`

À éviter

- `aes128-cbc`, `aes192-cbc`, `aes256-cbc`
- `3des-cbc`
- `arcfour*`

Authentification de Message (MACs)

Recommandés

- `hmac-sha2-512-etm@openssh.com`
- `hmac-sha2-256-etm@openssh.com`
- `umac-128-etm@openssh.com`

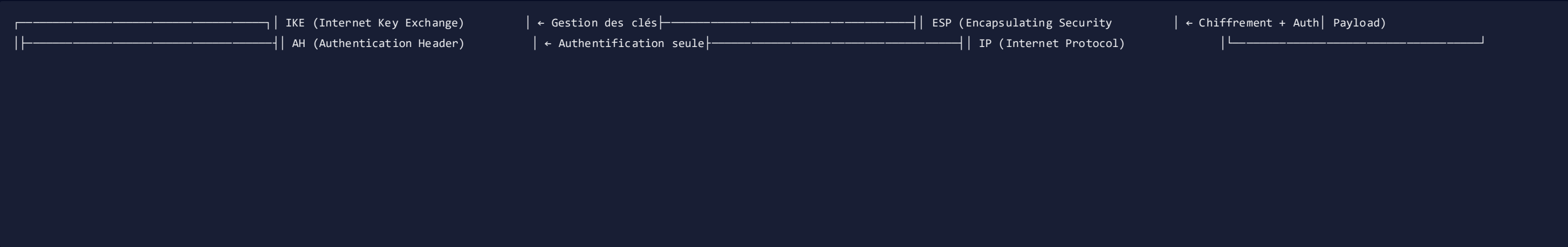
À éviter

- Tous les MACs non-ETM (Encrypt-then-MAC)
- `hmac-sha1*`
- `hmac-md5*`

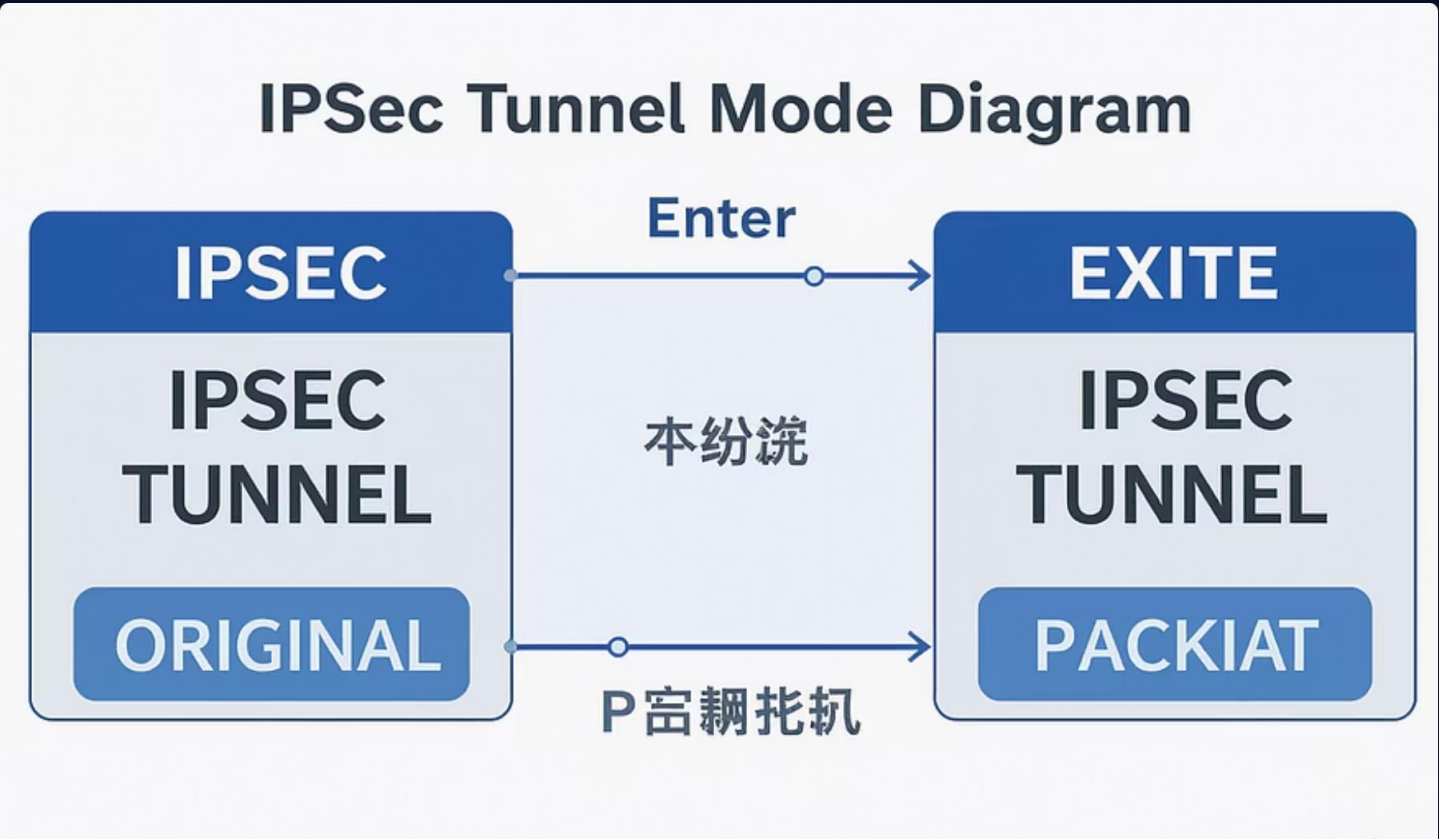
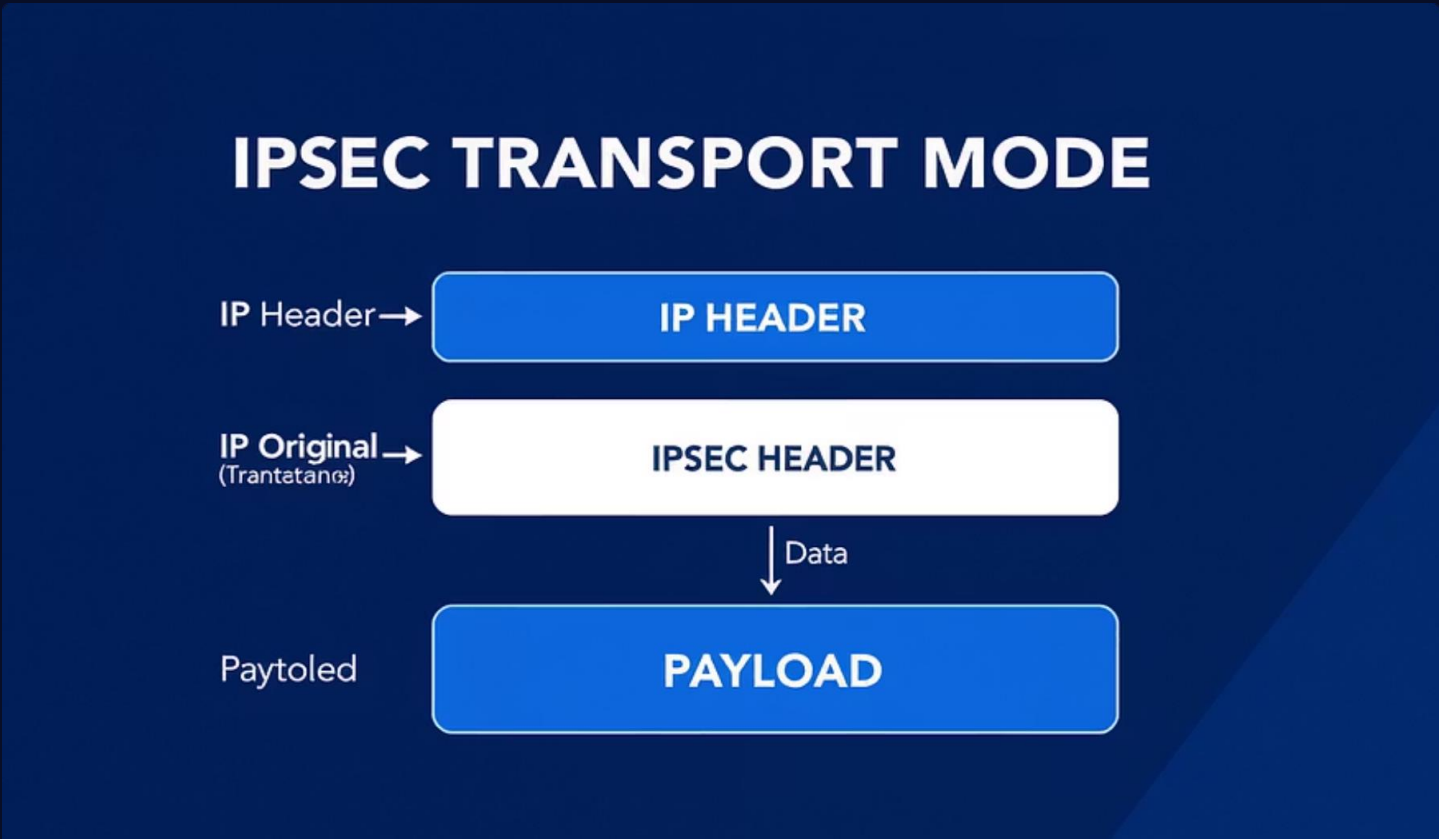
IPsec - Internet Protocol Security

Architecture Générale

Composants IPsec

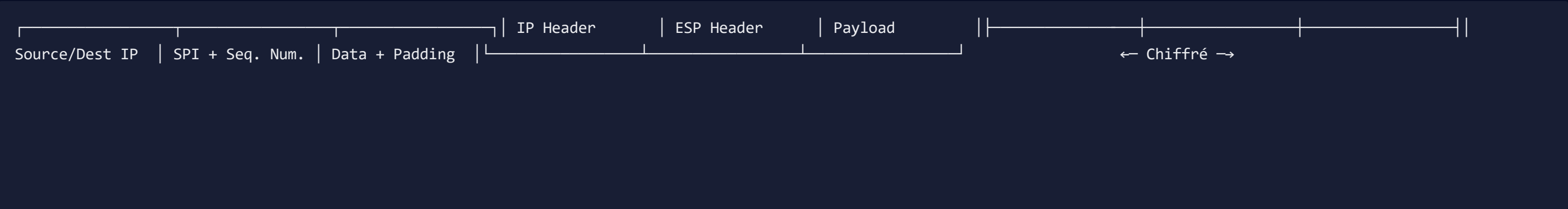


Modes de Fonctionnement



ESP - Encapsulating Security Payload

Format du Paquet ESP



Champs Importants

- **SPI** (Security Parameter Index) : Identifie l'association de sécurité
- **Sequence Number** : Protection contre le replay
- **Payload** : Données chiffrées et authentifiées

Algorithmes ESP Recommandés

Chiffrement + Authentification

- AES256-GCM (Authenticated Encryption)
- AES128-GCM
- ChaCha20-Poly1305

Obsolètes

- DES, 3DES
- AES-CBC sans authentification intégrée

IKE - Internet Key Exchange

Phases IKEv1 (Obsolète)

Phase 1 : Établissement d'un canal sécurisé (ISAKMP SA)**Phase 2** : Négociation des SAs IPsec

IKEv2 (Recommandé)

Simplification en un seul échange :



Authentification IKE

Pre-Shared Key (PSK)

Avantages : Simple à configurer**Inconvénients** : Partage de secret, pas de scalabilité**Usage** : Petits déploiements, tests

Certificats X.509

Avantages : Scalable, révocation possible, pas de secret partagé**Inconvénients** : Complexité PKI**Usage** : Déploiements d'entreprise

EAP (Extensible Authentication Protocol)

Avantages : Intégration avec annuaires (LDAP, AD)**Inconvénients** : Complexité additionnelle**Usage** : Accès distant utilisateurs

```
Linux IPSEC Tunnel Tunset/1
ane or OF IPSECERS=ME
ngSwan_Stigt IPSEC ISIL64:
structantied: ESTACBLISISHED).1
n conec: ourte pion
f loczintealierneill.confiurtission.
e0Desaintigt IPSeC SL
e0Desa_Stigt IPS6(CBLISPlurED),1
e0Desa_Stigt IPS6CL.L15L
e0Desa_Stigt IPS6CL.L15PlurED),1
e0Desa_Stigt IPS6CL.L15PlurED),1
e0Desa_Stigt IPS6CL.L15PlurED),1
e0Desa_Stigt IPS6CL.L15Plurtission.
e0Desa_Stigt IPS6CL.L15PlurED),1
```

Freeu Tunset/11				
291924	F116	0.901:	0.0053	- 1:32.6
109384	7228	0.901:	0.0078	- 0:32.7
199274	7318 1112	0.901:	0.0068	- 0:01.0
100584	7215 1152	0.901:	0.0033	- 0:22.7
109364	5216 1152	0.901:	0.0056	- 0:10.8
109594	7318 1152	0.901:	0.0056	- 0:28.0
100284	F212 1152	0.901:	0.0070	- 0:10.8
109384	3218 1152	0.901:	0.0093	- 2222.8
109364	6318 1152	0.901:	0.0090	- 0:22.9
100584	6118 1152	0.901:	0.0068	- 0:28.9
107384	6118 1152	0.901:	0.0001	- 2:11.0
108664	6318 1152	0.901:	0.0093	- 0:21.9
109984	5112 1152	0.901:	0.0063	- 2222.74
109574	5118 1152	0.901:	0.0000	- 0:34.74
100544	6313 1152	0.901:	0.0000	- 2232.1
200564	5314 1152	0.901:	0.0060	- 2215.9
198565	6118 1152	0.901:	0.0050	- 0732.14
100544	6118 1152	0.921:	0.0050	- 2219.9
100367	6218 1152	0.901:	0.0058	- 3:20.0
223148	6318 1152	0.901:	0.0076	- 2222.9
224386	7218 1152	0.901:	0.0003	- 2222.0
202364	6118 1152	0.901:	0.0004	- 2221.0
200584	6113 1152	0.901:	0.0066	- 2022.0
290584	5113 1152	0.901:	0.0004	- 2212.9

Configuration Pratique

Exemple strongSwan (Linux)

```
# /etc/ipsec.confconn tunnel-entreprise left=192.168.1.1 leftsubnet=192.168.1.0/24 leftid=@vpn.entreprise.com
leftcert=vpn-gateway.crt right=203.0.113.5 rightsubnet=10.0.0.0/16 rightid=@client.entreprise.com
keyexchange=ikev2 ike=aes256-sha256-modp2048! esp=aes256-sha256! auto=start dpdaction=restart
dpddelay=30s
```

Vérification et Debug

```
# Status des connexionsipsec status# Logs détaillésipsec up tunnel-entreprise# Trafic ESPtcpdump -i any esp# SAs
activesip xfrm stateip xfrm policy
```