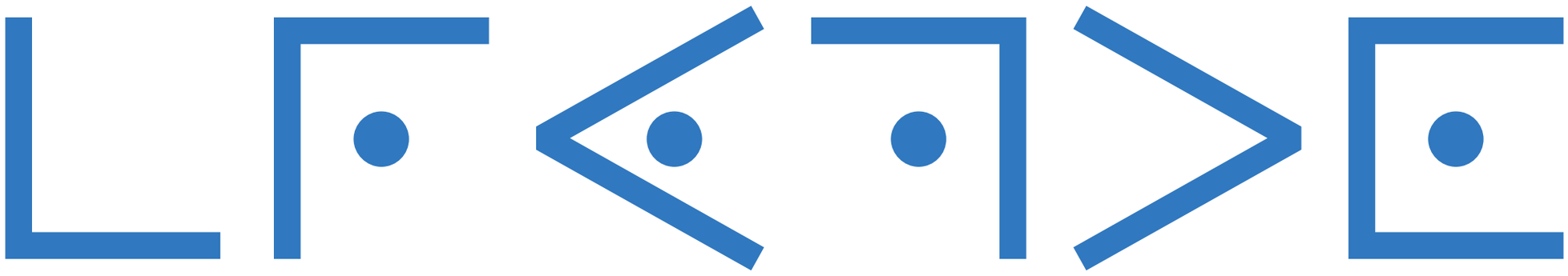


vers un canal sécurisé

Nicolas Ollinger

M1 informatique — 2024/2025

Protocole cryptographique

Un protocole cryptographique est construit à partir de briques, les primitives cryptographiques, dans le but d'assurer un certain nombre de propriétés, typiquement :

- confidentialité ;
- intégrité ;
- authenticité ;
- non répudiabilité.

Confidentialité

- Assurer que seules les deux parties ont accès aux données échangées.

- Empêche l'**écoute** des données en transit.

- Selon le contexte cette confidentialité peut être **persistante** dans le temps.

Intégrité

- Assurer la correction et la consistance des données transmises.
- Empêche de **modifier** les données en transit.
- Empêche de **forger** des nouvelles données.
- Selon le contexte ce contrôle d'intégrité peut se faire avec ou sans **répudiabilité**.

Authenticité

- Permettre aux deux parties en présence de **valider l'identité** de l'autre partie.

- Empêche les accès non autorisés mais aussi...

- Empêche les attaques de type **homme du milieu**.

- Affaibli parfois dans le cadre client/serveur par une authentification du serveur uniquement.

Attention à la notion d'identité (IP, DNS, ... ?)

Quelques primitives cryptographiques

- **Cryptographie symétrique** (à clé secrète) ;
- Cryptographie asymétrique (à clé publique) ;
- Générateurs de nombres pseudo-aléatoires de qualité cryptographique ;
- Fonctions de hachage de qualité cryptographique ;
- **Codes d'authentification de messages** (MAC) ;
- Algorithmes de signature numérique.

Canal sécurisé

Aujourd’hui ~~au tableau~~

Protocole pour un canal sécurisé

Alice et Bob ont échangé des clés secrètes de 256 bits et élu les algorithmes **AES-256-CTR** et **HMAC-SHA256**. Ils souhaitent établir un canal de communication sécurisé entre eux, sur une socket TCP. Eve écoute les échanges « sécurisés ».

Proposer un protocole cryptographique qui permet d’assurer la confidentialité et l’intégrité des échanges.

```
def init(K):
    etat = {}
    etat.KEAB = HMAC-SHA256(K, "ENCRYPT A→B")      } clés dérivées
    etat.KEBA = HMAC-SHA256(K, "ENCRYPT B→A")
    etat.KMAB = HMAC-SHA256(K, "AUTH A→B")
    etat.KMBA = HMAC-SHA256(K, "AUTH B→A")
    etat.COUNTAB = 0                               } derniers messages reçus
    etat.COUNTBA = 0
    return etat
```

```
def send(etat, msg, xtra):                    (Alice → Bob)
    num = etat.COUNTAB + 1                    message, données additionnelles à authentifier
    etat.COUNTAB += 1
    nonce = get_random_bits(32)
    assert num < 2^32
    cipher = AES.new(etat.KEAB, nonce = nonce || num,
                                mode = AES.CTR-MODE)
    C = cipher.encrypt(msg)
    H = HMAC-SHA256(etat.KMAB, num || nonce || len(xtra) || xtra
                                || C)
    send_msg(num || nonce || len(xtra) || xtra || C || H)
    return etat
```

Alice → BoG

```
def recv(stat, msg):
    num || nonce || len(xtra) || xtra || C || H = msg
    H' = HMAC-SHA256(stat.KMAB, num || nonce || len(xtra) ||
                                                xtra || C)
    if H ≠ H' or num <= stat.COUNTAB:
                                                    return None
    stat.COUNTAB = num
    cipher = AES.new(stat.KEAB, nonce = nonce || num,
                                                    mode = AES.CTR_MODE)
    data = cipher.decrypt(C)
    return (stat, data, xtra)
```

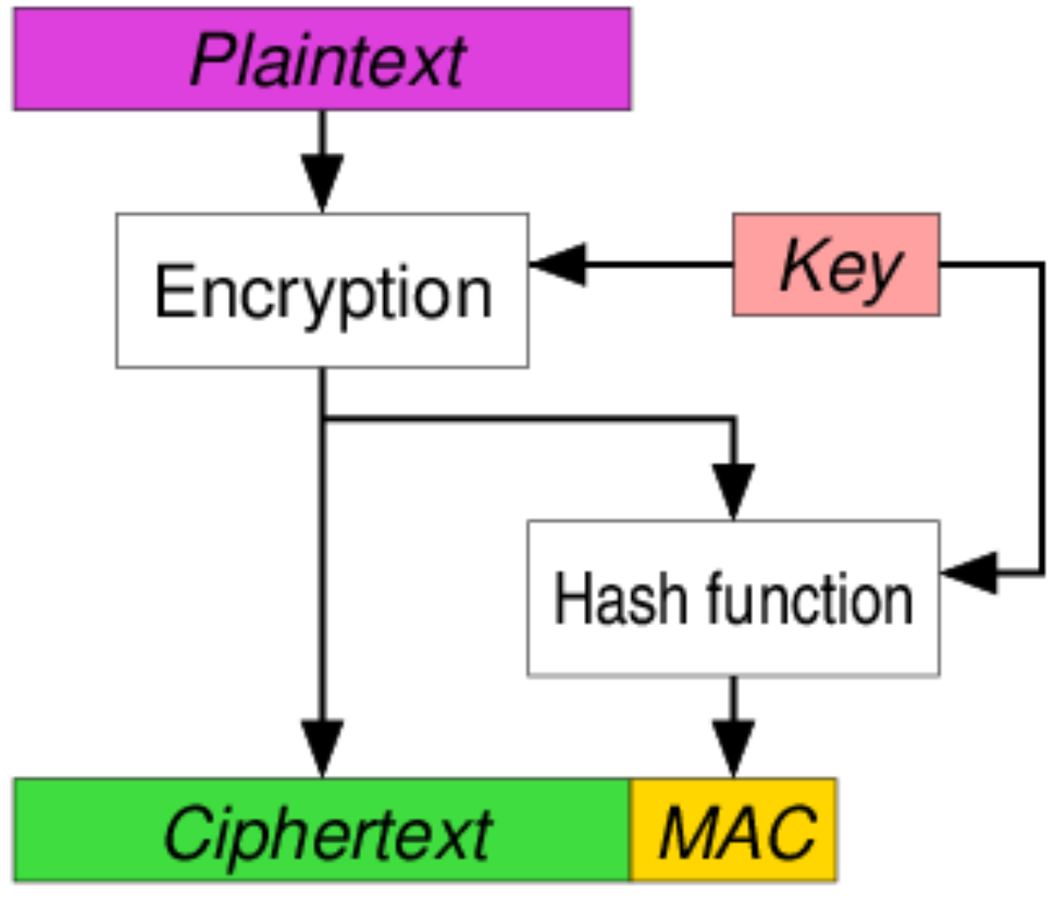
VALIDE H et num

EtM : Encrypt then MAC

Calculer le MAC du message déjà chiffré.

Utilisé par IPSec.

$$E_K(M) \| H(K', E_K(M))$$

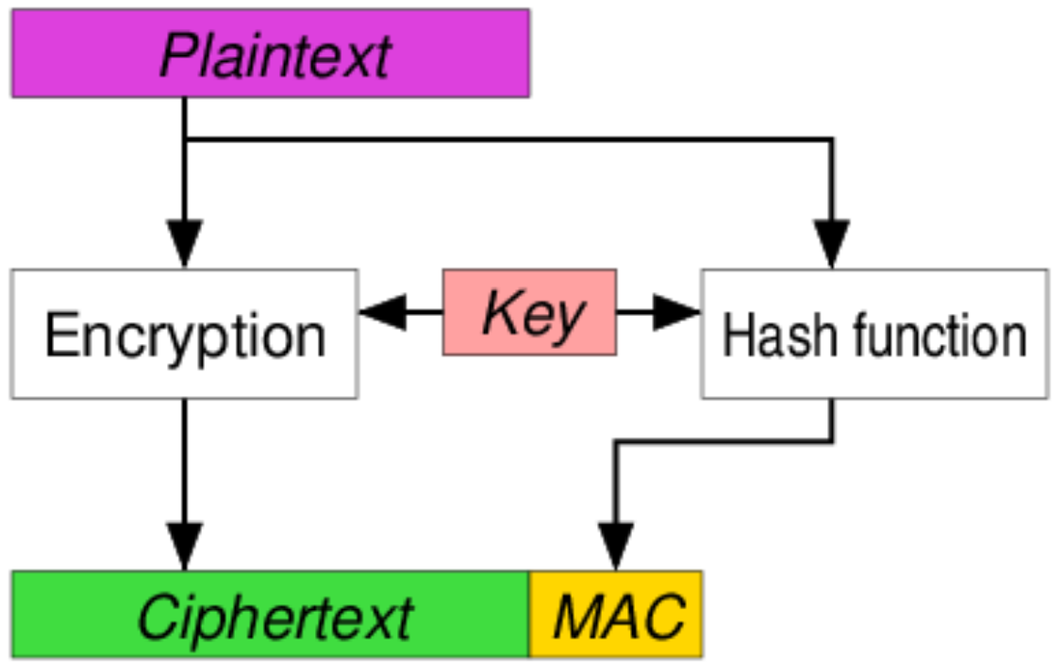


E&M : Encrypt and MAC

Chiffrer et calculer le MAC en parallèle.

Utilisé par SSH.

$$E_K(M) \| H(K', M)$$

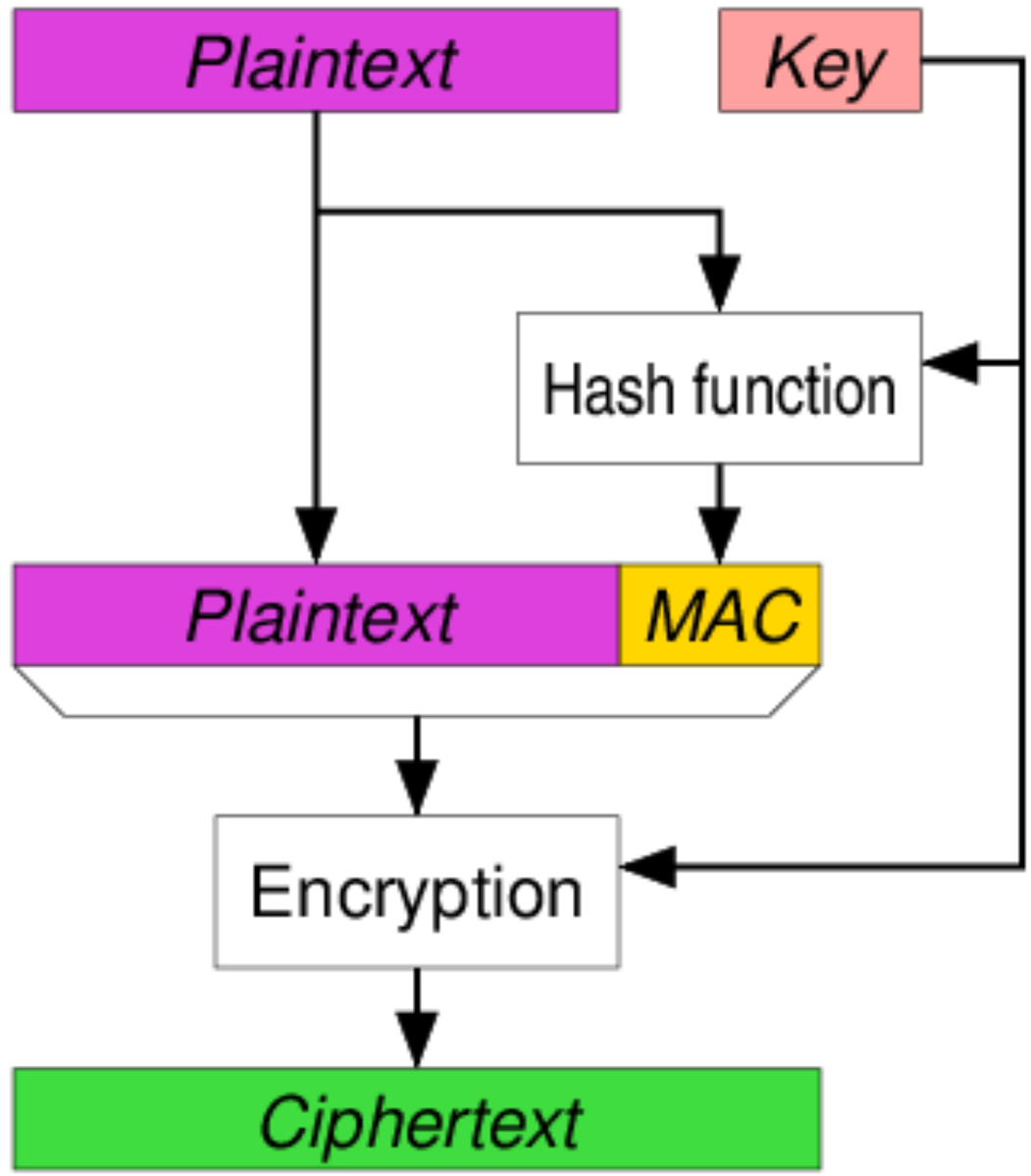


MtE : MAC then Encrypt

Ajouter le code MAC au texte en clair puis chiffrer l'ensemble.

Utilisé par SSL/TLS.

$$E_K(M \| H(K', M))$$



Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm

Mihir Bellare* Chanathip Namprempre†

July 14, 2007

Abstract

An authenticated encryption scheme is a symmetric encryption scheme whose goal is to provide both privacy and integrity. We consider two possible notions of authenticity for such schemes, namely integrity of plaintexts and integrity of ciphertexts, and relate them (when coupled with IND-CPA) to the standard notions of privacy (IND-CCA, NM-CPA) by presenting implications and separations between all notions considered. We then analyze the security of authenticated encryption schemes designed by "generic composition," meaning making black-box use of a given symmetric encryption scheme and a given MAC. Three composition methods are considered, namely *Encrypt-and-MAC*, *MAC-then-encrypt*, and *Encrypt-then-MAC*. For each of these, and for each notion of security, we indicate whether or not the resulting scheme meets the notion in question assuming the given symmetric encryption scheme is secure against chosen-plaintext attack and the given MAC is unforgeable under chosen-message attack. We provide proofs for the cases where the answer is "yes" and counter-examples for the cases where the answer is "no."

The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)*

Hugo Krawczyk

EE Department,
Technion, Haifa, Israel.
hugo@ee.technion.ac.il

Abstract. We study the question of how to generically compose *symmetric* encryption and authentication when building "secure channels" for the protection of communications over insecure networks. We show that any secure channels protocol designed to work with any combination of secure encryption (against chosen plaintext attacks) and secure MAC must use the encrypt-then-authenticate method. We demonstrate

Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation

Jonathan Katz[1] and Moti Yung[2]

[1] Department of Computer Science, Columbia University
jkatz@cs.columbia.edu
[2] CertCo, NY, USA
moti@cs.columbia.edu, moti@certco.com

Abstract. We find certain neglected issues in the study of private-key encryption schemes. For one, private-key encryption is generally held to the same standard of security as public-key encryption (i.e., indistinguishability) even though usage of the two is very different. Secondly, though the importance of secure encryption of single blocks is well known, the security of modes of encryption (used to encrypt multiple blocks) is often ignored. With this in mind, we present definitions of a new notion of security for private-key encryption called *encryption unforgeability* which captures an adversary's inability to generate valid ciphertexts. We show applications of this definition to authentication protocols and adaptive chosen ciphertext security.

Modes opératoires authentifiés chiffrés

Modes AEAD

Proposer des modes opératoires qui apportent en une seule brique le chiffrement et le contrôle d’intégrité et qui les composent de sorte à assurer la sécurité.

AEAD = authenticated encryption with associated data

NIST Special Publication 800-38C

Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality

Morris Dworkin

C O M P U T E R S E C U R I T Y

Mode CCM

CCM = CTR + CBC-MAC en mode MtE

Ici la réutilisation de la clé ne pose pas de problème.

$$(N, A, P) \rightarrow B_0, B_1, \ldots, B_n$$

$$Y_0 = E_K(B_0)$$

$$Y_i = E_K(B_i \oplus Y_{i-1}) \quad \forall i$$

$$T = Y_n$$

$$S_0 = E_K(\mathrm{Ctr}_0)$$

$$S = E_K(\mathrm{Ctr}_1) \| \cdots \| E_K(\mathrm{Ctr}_m)$$

$$C = (P \oplus T) \| (S \oplus S_0)$$

NIST Special Publication 800-38D
November, 2007

Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC

Morris Dworkin

National Institute of Standards and Technology

U.S. Department of Commerce

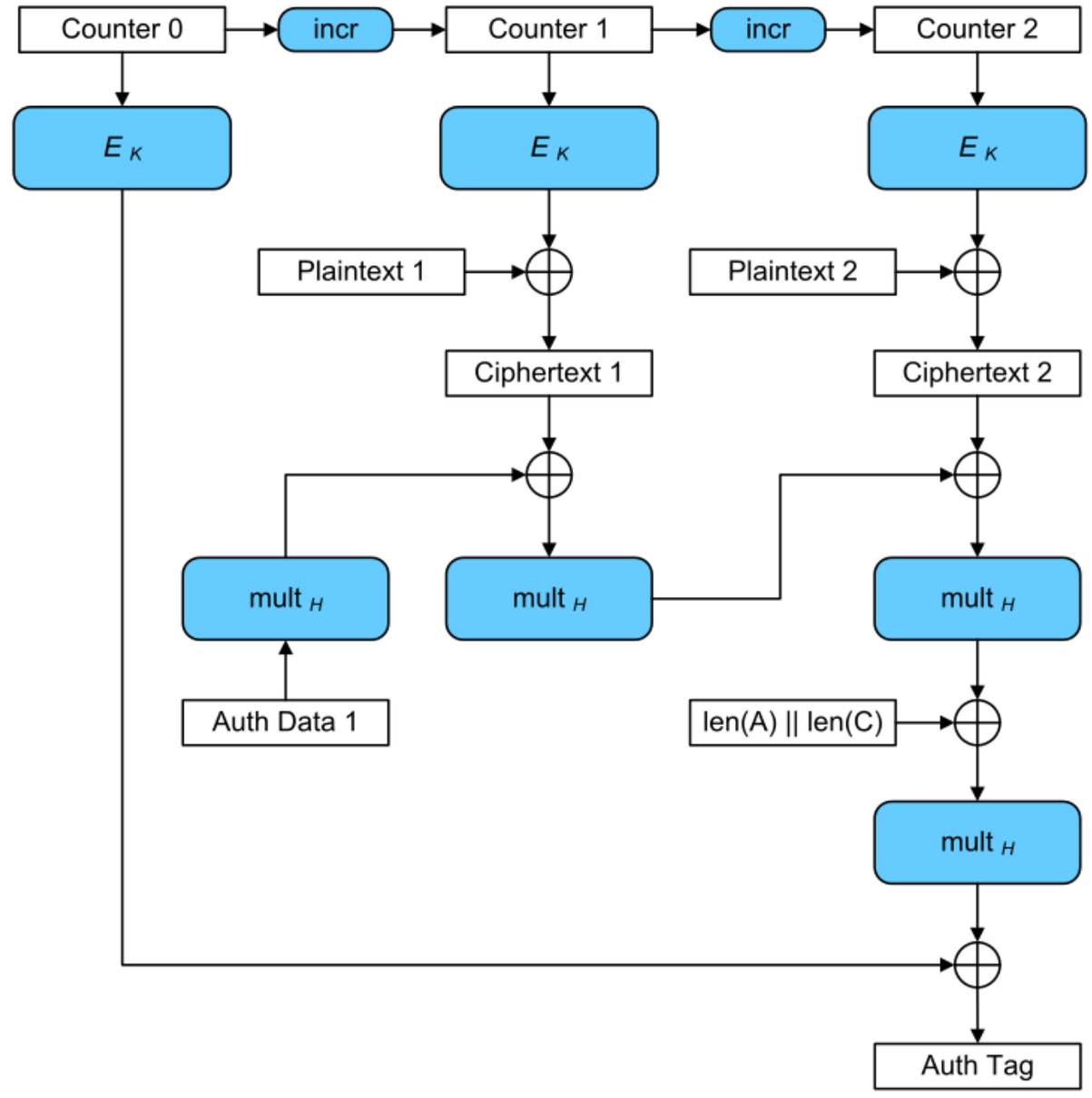
COMPUTER SECURITY

Mode GCM

GCM = CTR + GMAC
en mode EtM

où GMAC est un code MAC polynomial dans GF(2^{128}).

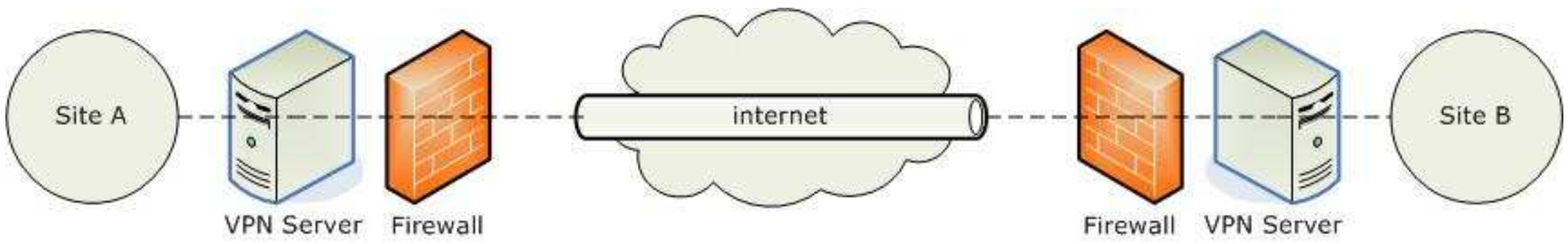
Primitive très efficace à calculer.



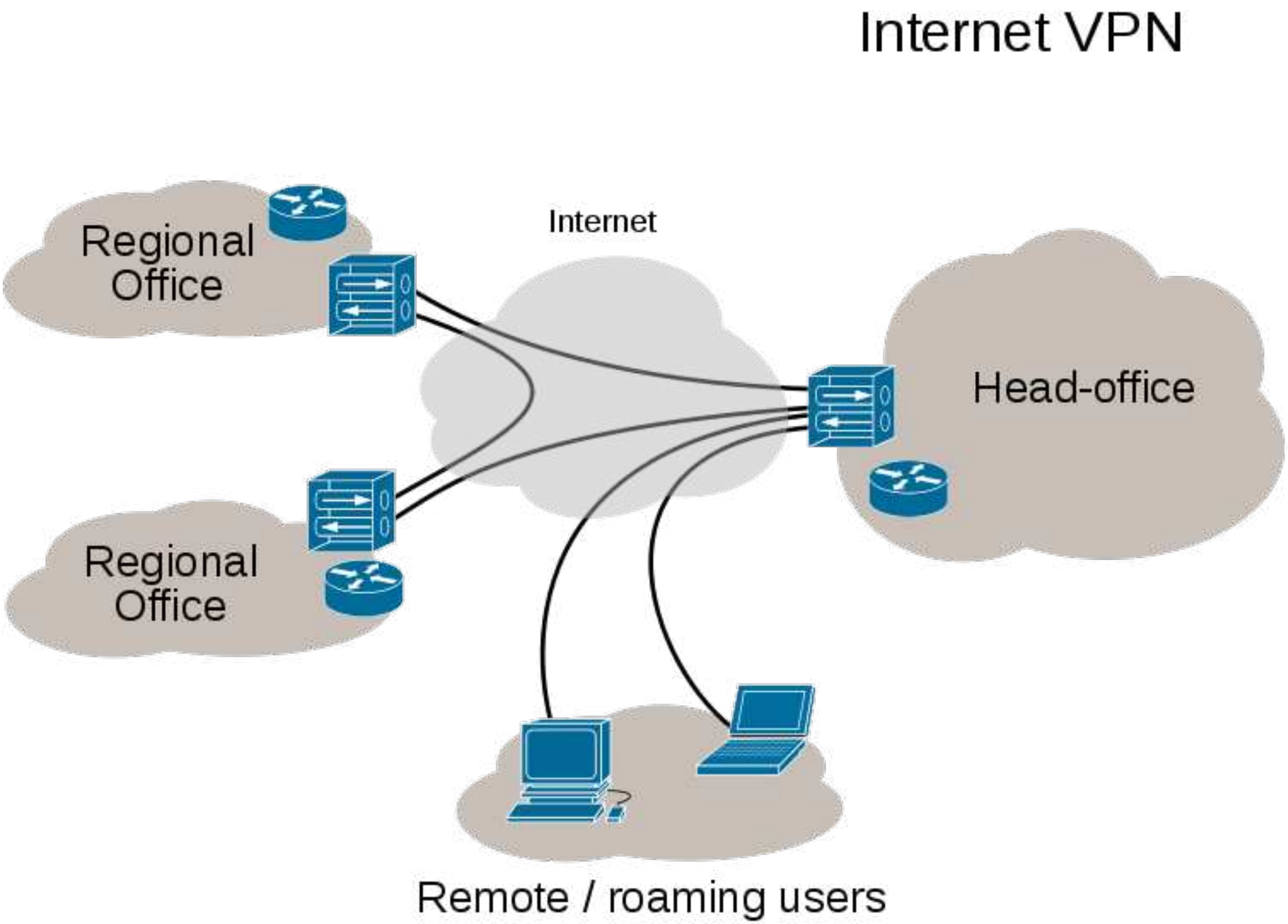
En pratique : VPN avec IPsec

Virtual **P**rivate **N**etwork

- Un **VPN** permet de créer une **liaison sécurisée** entre deux réseaux distants à travers un réseau non sûr (par exemple l'internet public).
- Se substitue (ou complète !) à la location d'une liaison spécialisée entre deux routeurs éloignés.



lan-to-lan, host-to-lan



Fonctionnalités

Authentification Vérifier l'identité des deux extrémités du VPN par authentification mutuelle.

Contrôle d'intégrité Empêcher la modification du flux réseau qui traverse le VPN (prévenir l'ajout de paquets ou la modification du contenu des paquets).

Confidentialité Empêcher l'écoute des données.

Technologies

- Protocoles ad hoc propres à une technologie :
  - ➡ OpenSSH (PermitTunnel + -w)
  - ➡ OpenVPN (tunnels SSL sur UDP/TCP)

- Protocoles normalisés :
  - ➡ IPsec en mode transport ou tunnel
  - ➡ IPsec/GRE : couche 3 + multicast + …
  - ➡ IPsec/L2TP : couche 2

IP security (IPsec)

RFC 4301

« (…) IPsec is designed to provide interoperable, high quality, cryptographically-based security for IPv4 and IPv6. The set of security services offered includes access control, connectionless integrity, data origin authentication, detection and rejection of replays (a form of partial sequence integrity), confidentiality (via encryption), and limited traffic flow confidentiality. (…) »

Une solution normalisée pour le déploiement de VPN IP interopérables mise en œuvre par la quasi-totalité des acteurs du marché : Cisco, Juniper, SUN, HP, Microsoft, Checkpoint, GNU/Linux, OpenBSD, *etc*

Composants d'IPsec

- Des protocoles pour le transport des paquets qui transitent à l'intérieur du VPN :
  - ➡ Authentification Header (**AH**)
  - ➡ Encapsulating Security Payload (**ESP**)

- Une famille d'algorithmes cryptographiques normalisés à combiner à ces protocoles.

- Un protocole (optionnel) de gestion automatique de clés : Internet Key Exchange (**IKEv2**) Protocol.

- Un mécanisme type pare-feu pour décider des paquets qui passent par le tunnel et doivent être encapsulés par AH ou ESP.

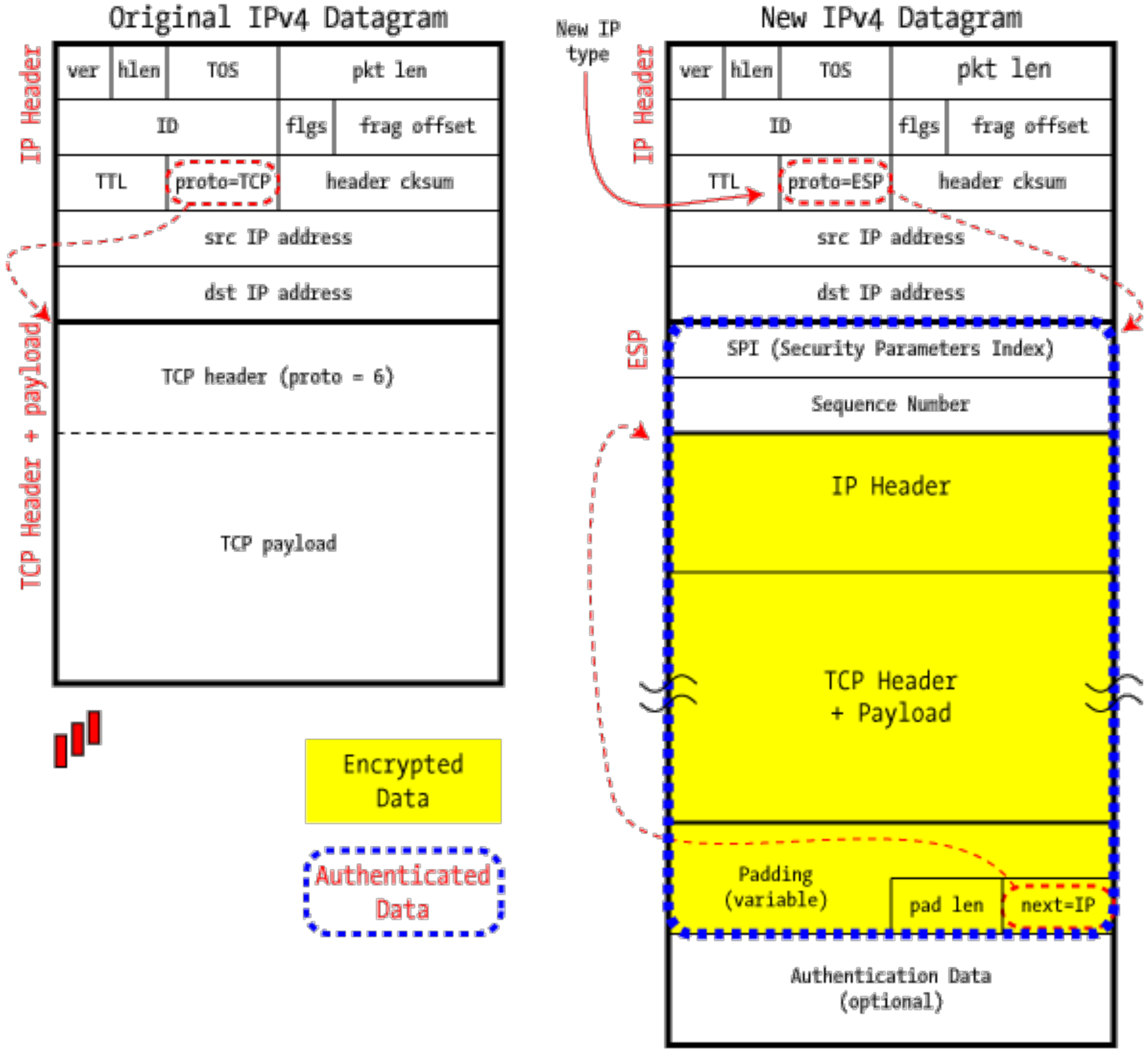
AH et **ESP**

- Encapsulation de chaque paquet IP qui transite par le VPN, nouvelle entête IP + entête AH/ESP +
  - ➡ la charge utile du paquet IP en **mode transport** ;
  - ➡ le paquet IP complet en **mode tunnel**.

- Information présente dans l'entête :
  - ➡ **SPI** (Security Parameters Index)
  - ➡ **SN** (Sequence Number)
  - ➡ **ICV** (Integrity Check Value) si contrôle d'intégrité

- Pour en savoir plus : **An Illustrated Guide to IPsec**

IPSec in ESP Tunnel Mode



Security **P**olicy **D**atabase

- Détermine la politique d'encapsulation IPsec du flux réseau entrant et sortant.
- Règle type pare-feu qui décide sur quels paquets appliquer quelle type d'encapsulation IPsec (AH vs ESP vs ESP+AH, transport vs tunnel).

```
spdadd 10.0.0.0/24 10.0.1.0/24 any -P in
  ipsec esp/tunnel/100.10.10.10-100.20.20.20/require;
spdadd 10.0.1.0/24 10.0.0.0/24 any -P out
  ipsec esp/tunnel/100.20.20.20-100.10.10.10/require;
```

Security **A**ssociation **D**atabase

- Détermine la politique de sécurité à partir du SPI, du type de protocole IPsec (AH, ESP) et éventuellement d'adresses IP.

- **S**ecurity **A**ssociation = protocoles + clés + …

```
add 100.10.10.10 100.20.20.20 esp 1337
                -E des-cbc 0xcafebabec00170ad
                -A hmac-md5 "there is no cake";
```

Gestion des clés

- Gestion **manuelle** : configuration statique des SA à l'aide de clés ou de certificats statiques.

- Gestion **automatique** des clés avec **IKE** : génération à la volée de clés pour chaque session IPsec (4 clés si ESP+AH bidirectionnel).

IKE et **IKEv2**

RFC 7296

1. Négociation sur la protection d'IKE
   (Diffie-Hellman à partir d'un secret partagé, …)

2. Négociation sur la protection IPsec
   (choix des protocoles et génération de clés)

3. Session IPsec jusqu'à expiration

sous GNU/Linux

- **https://wiki.debian.org/IPsec**

- ipsec-tools + racoon

- Configuration :
  racoon. conf + psk.txt + ipsec-tools.conf