Introduction à la Cryptographie

Fonction de hachage - MAC

Fonction de hachage cryptographique

Scénario

Imaginons qu'on télécharge la dernière version d'Ubuntu depuis un serveur cloud.

- Est-ce la copie authentique du système d'exploitation ?
- Comment savoir que ce n'est pas un *malware* ?

⇒ **Intégrité ?**

Scénario

- Télécharger la version depuis la plateforme des développeurs
  - Enlève tout l'intérêt du cloud
- Comparer une partie du fichier d'origine (par ex : quelques premiers et derniers bits)
  - Le *malware* peut avoir concervé ces quelques bits et modifié le reste
  - Il faut regarder l'intégralité du fichier
- Comparer l'**empreinte** du fichier (appelé aussi : **condensé**, **condensat**, **haché**, **hash**)
  - L'empreinte doit être "unique"

Empreinte : exemples d'applications

- Empreinte numérique
  - identification d'un malware, détection d'instrusion, ...
- Signatures numériques
- Code d'authentification de message
- Stockage de mots de passe
- ...

Fonction de hachage

Définition : Fonction de hachage

Une **fonction de hachage** H prend en entrée une séquence binaire x de **longueur arbitraire** et retourne une séquence binaire de taille fixe n.

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

Collision

Une fonction de hachage ne peut pas être injective : il existe $\mathbf{x}_1 \neq \mathbf{x}_2$ tel que $H(\mathbf{x}_1) = H(\mathbf{x}_2)$

Preuve : La cardinalité de $\{0,1\}^*$ est plus grande que la cardinalité de $\{0,1\}^n$.

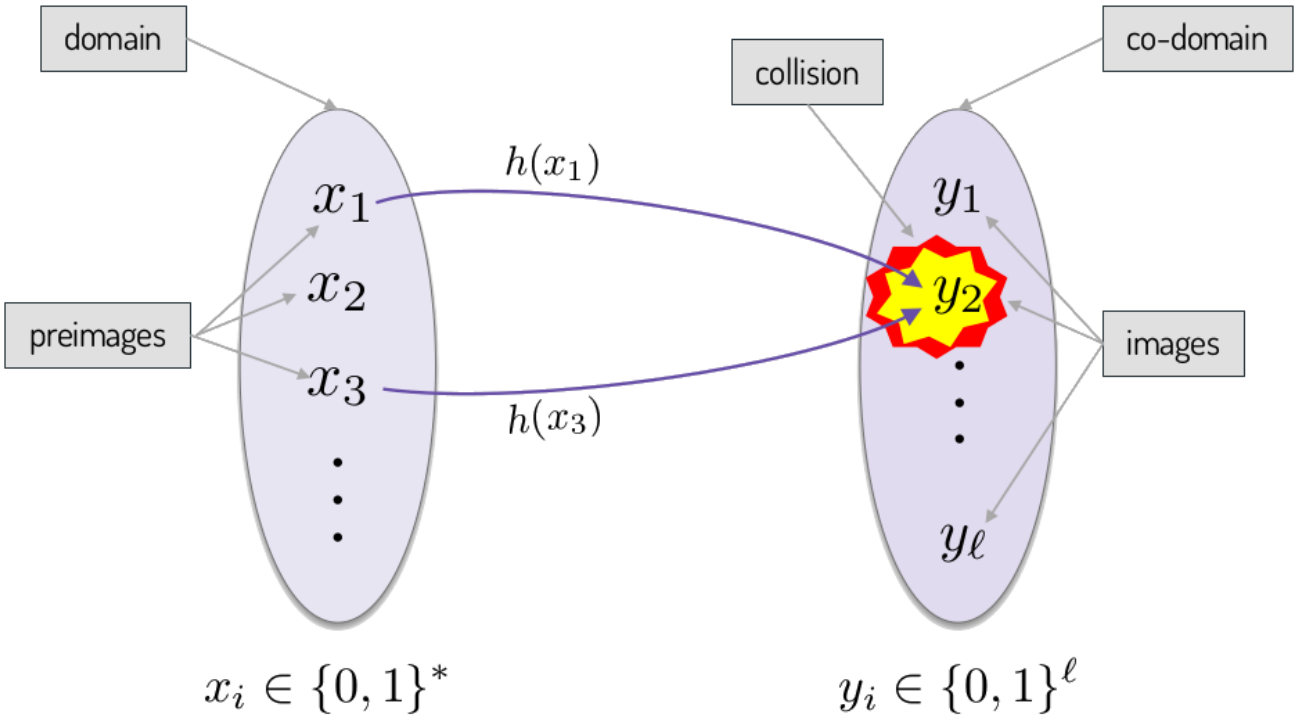


Figure – Source : WHISPER Lab - Aleksander Essex

Si la fonction de hachage est simple (ex : $H(x) = x \mod 256$), des images proches ont des préimages proches $\Rightarrow$ problème

Fonction de hachage cryptographique

Définition : Fonction de hachage cryptographique

Une **fonction de hachage cryptographique** H prend en entrée une séquence binaire x de **longueur arbitraire** et retourne une séquence binaire **pseudo-aléatoire** de taille fixe n.

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

Propriété d'une fonction de hachage cryptographique

- **résistance à la préimage** : Soit $\mathbf{y} \in \{0,1\}^n$, il devrait être difficile de trouver un message $\mathbf{x} \in \{0,1\}^*$ tel que $\mathbf{y} = H(\mathbf{x})$. On dit que H est une fonction à sens unique.
- **résistance à la seconde préimage** : Soit $\mathbf{x}_1 \in \{0,1\}^*$, il devrait être difficile de trouver un autre message $\mathbf{x}_2 \in \{0,1\}^*$ tel que $H(\mathbf{x}_1) = H(\mathbf{x}_2)$.
- **résistance aux collisions** : Il devrait être difficile de trouver deux messages différents $\mathbf{x}_1 \in \{0,1\}^*$ et $\mathbf{x}_2 \in \{0,1\}^*$ tel que $H(\mathbf{x}_1) = H(\mathbf{x}_2)$.

Remarque : Une fonction de hachage n'est pas une méthode de chiffrement !

Modèle de l'oracle aléatoire

Afin de prouver la sécurité, on considère une fonction de hachage cryptographique *idéale* définie comme suit :

- pour chaque message $\mathbf{x} \in \{0,1\}^*$ qui n'a jamais été haché auparavant, $H(\mathbf{x}) = \mathbf{u}$ où $\mathbf{u}$ est tiré uniformément au hasard sur $\{0,1\}^n$.
- si $\mathbf{x}$ a déjà été haché précédement, alors $H(\mathbf{x})$ correspond à la valeur qui a déjà été tirée.

Il s'agit d'un modèle théorique puisqu'il suppose que H est omniscient et qu'il sait à tout moment ce qui s'est passé à n'importe quel moment et n'importe où dans le système.

Attaques par brute force

Attaque de préimage

Exercice

Résistance à la préimage : Soit $\mathbf{y} \in \{0,1\}^n$, il devrait être difficile de trouver un message $\mathbf{x} \in \{0,1\}^*$ tel que $\mathbf{y} = H(\mathbf{x})$.

1. Décrire l'attaque par brute force sur la préimage.
2. Déduire que la complexité de l'attaque est de l'ordre de $O(2^n)$

Attaque de seconde préimage

Exercice

Résistance à la seconde préimage : Soit $\mathbf{x}_1 \in \{0,1\}^*$, il devrait être difficile de trouver un autre message $\mathbf{x}_2 \in \{0,1\}^*$ tel que $H(\mathbf{x}_1) = H(\mathbf{x}_2)$.

1. Décrire l'attaque par brute force sur la seconde préimage.
2. Déduire que la complexité de l'attaque est de l'ordre de $O(2^n)$

Attaque de collisions

Attaque des anniversaires

Résistance aux collisions : Il devrait être difficile de trouver deux messages différents $\mathbf{x}_1 \in \{0,1\}^*$ et $\mathbf{x}_2 \in \{0,1\}^*$ tel que $H(\mathbf{x}_1) = H(\mathbf{x}_2)$.

- On choisit *aléatoirement* $x_1, x_2, ... \in \{0,1\}^*$ jusqu'à ce qu'on trouve une collision, c'est à dire x_i, x_j avec $i \neq j$ et tels que $H(x_i) = H(x_j)$.
- Grâce au paradoxe des anniversaires, cette méthode, pourtant rudimentaire, peut se révéler être très efficace surtout si la taille de l'empreinte est trop petite.

Paradoxe des anniversaires

- On se pose la question de savoir combien doit-on réunir de personnes afin d'avoir plus de 50% de chances d'avoir au moins deux personnes nées le même jour de l'année (pour simplifier, on suppose toutes les années non bissextiles, ce qui ne change guère le résultat).
- La réponse, très contre-intuitive, est 23. Si on réunit 50 personnes, la probabilité est de 97,04% et, pour 80 personnes, elle est de 99,99%, une quasi certitude.
- Le calcul est simple. Si n est le nombre personnes et $P(n)$ la probabilité cherchée, on a :

$$P(n) = 1 - A_{365}^{n} \cdot \frac{1}{365^n} = 1 - \frac{365!}{(365-n)!} \cdot \frac{1}{365^n}$$

- En revanche, si on calcule la probabilité P_{23} que, sur un groupe de 23 personnes, une personne ait son anniversaire le 29 septembre, on constate que celle-ci est beaucoup plus faible. En effet, $P_{23} = 1 - (\frac{364}{365})^{23} \approx 6,11\%$.

Attaque des anniversaires

En chiffre

- Très efficace surtout si la taille de l'empreinte est trop petite.
  - Si, par exemple, la taille de l'empreinte est de 64 bits, il faudra environ $7,2 \times 10^9$ essais pour que l'on ait au moins 75% de chances de découvrir une collision (on suppose que toutes les empreintes sont équiprobables). C'est tout à fait à la portée d'un PC. Il faut également comparer ce nombre au nombre d'empreintes possibles, à savoir environ $1,8 \times 10^{19}$.
  - Avec une empreinte de 512 bits et pour une même probabilité, il ne faudra pas moins de $1,9 \times 10^{77}$ essais, ce qui est hors de portée d'autant plus que cette attaque nécessite beaucoup de mémoire.

Résumé des attaques brute force

	Complexité
Préimage	$O(2^n)$
Seconde préimage	$O(2^n)$
Collision	$O(\sqrt{2^n})$

Fonction de hachage cryptographique itérative

Fonction de hachage cryptographique itérative

La plupart des fonctions de hachage cryptographique qui sont utilisées sont **itératives**.

- On considère une **fonction de compression**

$$h : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$$

- On divise le message $\mathbf{x}$ en t messages de k bits (on fait du *bourrage* si nécessaire)

$$\mathbf{x} = \mathbf{x}_1 || \mathbf{x}_2 || \cdots || \mathbf{x}_t$$

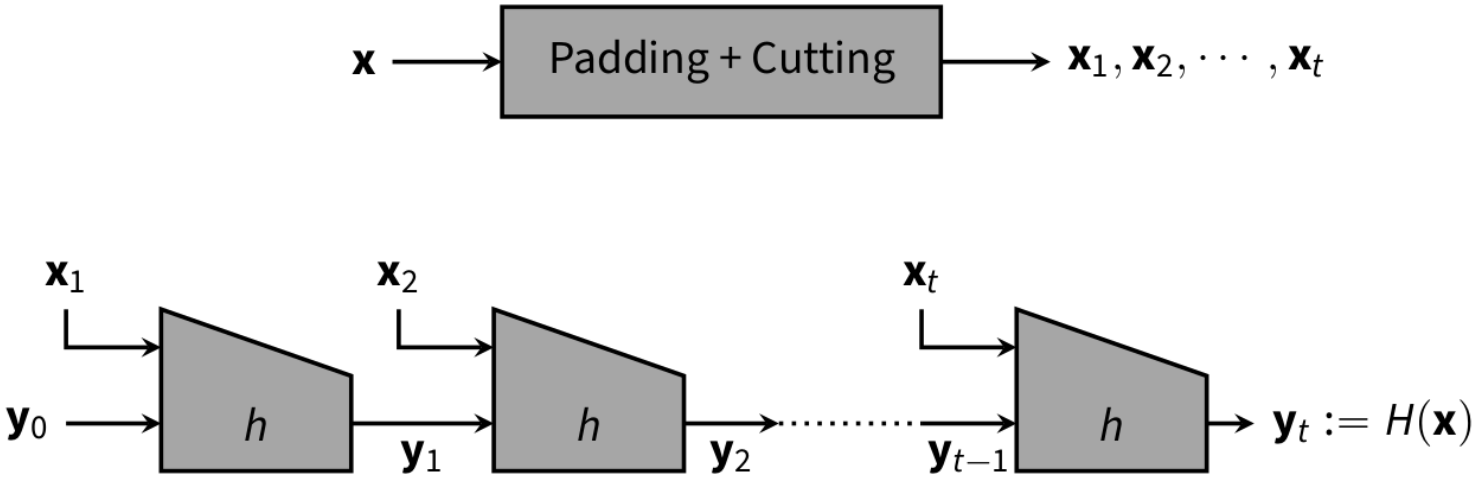
- Pour tout $1 \leq i \leq t$ et un **vecteur d'initialisation** $\mathbf{y}_0$, on calcule

$$\mathbf{y}_i = h(\mathbf{x}_i, \mathbf{y}_{i-1})$$

- La sortie de la fonction itérative de hachage est

$$H(\mathbf{x}) = \mathbf{y}_t$$

Fonction de hachage cryptographique itérative



Fonction de hachage cryptographique itérative

Théorème

Soit H une fonction de hachage cryptographique itérative.
S'il y a une collision sur H, alors il y a une collision sur h.

Ceci implique qu'une fonction de hachage cryptographique H est autant sécurisé contre les collisions que la fonction de compression h.

Construction de Merkle

Construction de Merkle

Certaines fonctions de hachage cryptographique utilisent un **arbre de Merkle**.

On considère une **fonction de compression**

$$h : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$$

- On divise le message $\mathbf{x}$ en t messages de $2n$ bits

$$\mathbf{y}_1||\mathbf{y}_2||\cdots||\mathbf{y}_t = \mathbf{x}||bourrage$$

- On applique récursivement :

$$\begin{aligned} &\mathbf{y} = h(\mathbf{y}_1)||h(\mathbf{y}_2)||\cdots||h(\mathbf{y}_t) \\ &t = \left\lceil \frac{len(\mathbf{y})}{2n} \right\rceil \qquad \text{jusqu'à ce que } len(\mathbf{y}) = n \\ &\mathbf{y}_1||\mathbf{y}_2||\cdots||\mathbf{y}_t = \mathbf{y}||bourrage \end{aligned}$$

- La sortie de la fonction itérative de hachage est $H(\mathbf{x}) = \mathbf{y}$

Exercice : dessine l'arbre représentant la construction de Merkle

Construction de Merkle

Théorème

Soit H une fonction de hachage cryptographique définie par la construction de Merkle.
S'il y a une collision sur H, alors il y a une collision sur h.

Ceci implique qu'une fonction de hachage cryptographique H est autant sécurisé contre les collisions que la fonction de compression h.

Principaux standards

Principaux standards

Système	Empreinte	Blocs du message	Créateur	Année
MD4	128 bits	512 bits	Rivest	1990
MD5	128 bits	512 bits	Rivest	1991
SHA-0	160 bits	512 bits	US Gov.	1993
SHA-1	160 bits	512 bits	US Gov.	1995
SHA-256	256 bits	512 bits	US Gov.	2002
SHA-512	512 bits	1024 bits	US Gov.	2002
SHA-3	512 bits	4 variants	4 belges	2008

La plupart d'entre eux sont cassés. Il est préférable d'utiliser **SHA-3**.

Code d'authentification de message (CAM)

Message Authentication Code (MAC)

Code d'authentification de message

Objectif

Dans un **système de chiffrement symétrique**, Alice et Bob partage une **clé secrète k**.
Alice envoie un message chiffré $c_k(\mathbf{m})$ à Bob.

Comment Bob peut être certain que le message chiffré qu'il a reçu provient d'Alice, et qu'il n'a pas été modifié par Eve ?

En supposant que **m** est un message écrit en français. Si Eve modifie un ou plusieurs bits de $c_k(\mathbf{m})$, lorsque Bob déchiffrera le chiffré reçu, il obtiendra une séquence aléatoire de bits qui ont peu de chance d'être un texte lisible en français.

Code d'authentification de message

Objectif

Alice doit prouver à Bob que le message chiffré $c_k(\mathbf{m})$ qu'il a reçu est son message chiffré et qu'il n'a pas été remplacé ou modifié par un attanquant.

Alice doit montrer à Bob que l'auteur du texte chiffré $c_k(\mathbf{m})$ est celui qui possède la clé $\mathbf{k}$.

Remarque : En pratique, on évite d'utiliser la même clé pour chiffrer et pour authentifier un message.

Définition d'un CAM

Définition d'un CAM

Tout comme les fonctions de hachage cryptographique, un CAM prend en entrée une séquence binaire de **longueur arbitraire** et retourne une séquence binaire de **taille fixe**. La différence est que la sortie dépend de la **clé secrète** qui est partagé entre les parties.

Définition : Code d'authentification de message

Soit $\mathbf{k} \in \{0,1\}^t$ une clé secrète.

$$\begin{array}{rcl} CAM_{\mathbf{k}} : \{0,1\}^* & \longrightarrow & \{0,1\}^n \\ \mathbf{m} & \longmapsto & CAM_{\mathbf{k}}(\mathbf{m}) \end{array}$$

où

t est la taille de la clé secrète ($\sim$ 128 bits en pratique)

n est la taille du CAM (64, 128, 160, 256 ou 512 bits en pratique)

Schéma de fonctionnement - Solution 1

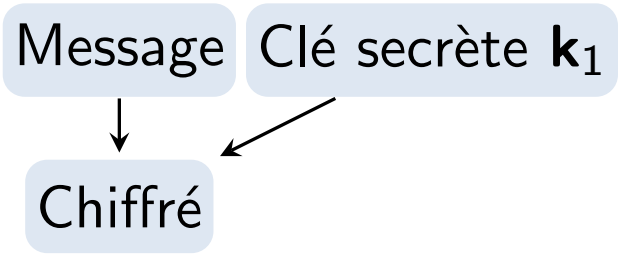
Encrypt-then-MAC configuration

Alice

Bob

Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration

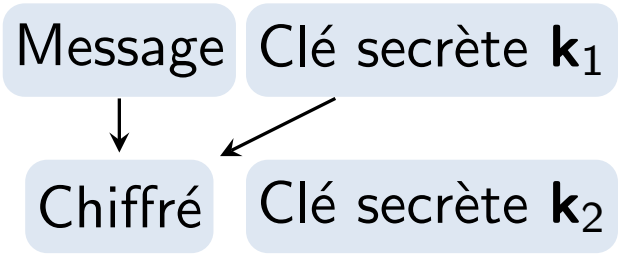


Alice

Bob

Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration



Alice

Bob

Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration

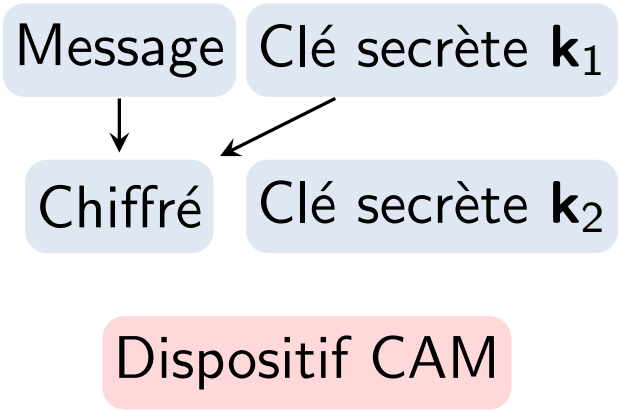
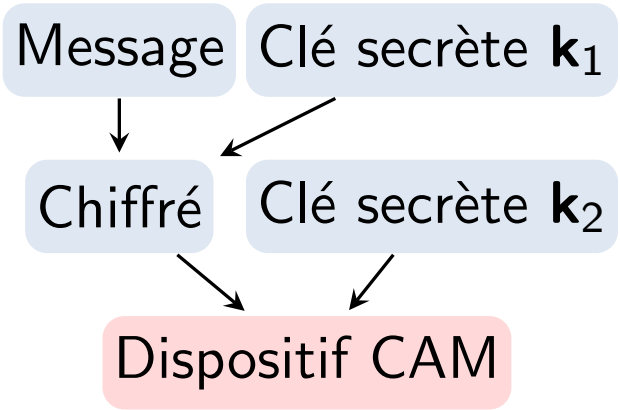


Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration

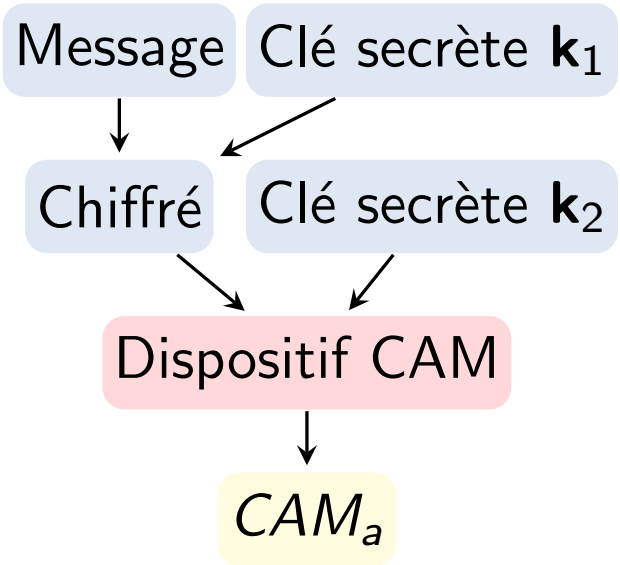


Alice

Bob

Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration



Alice

Bob

Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration

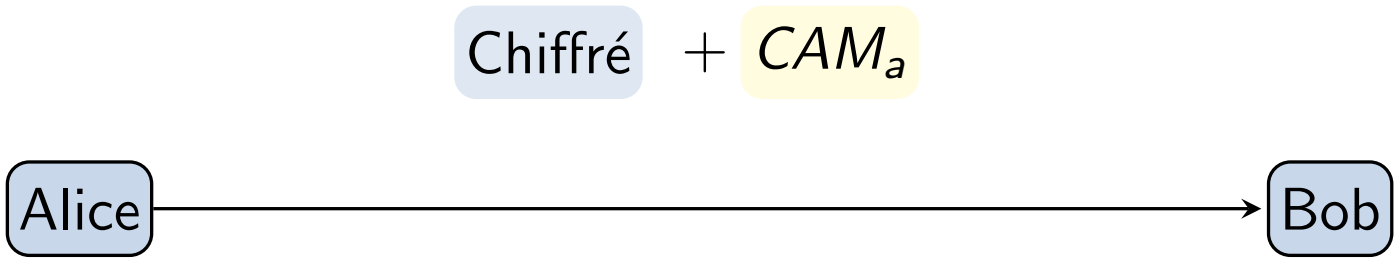


Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration

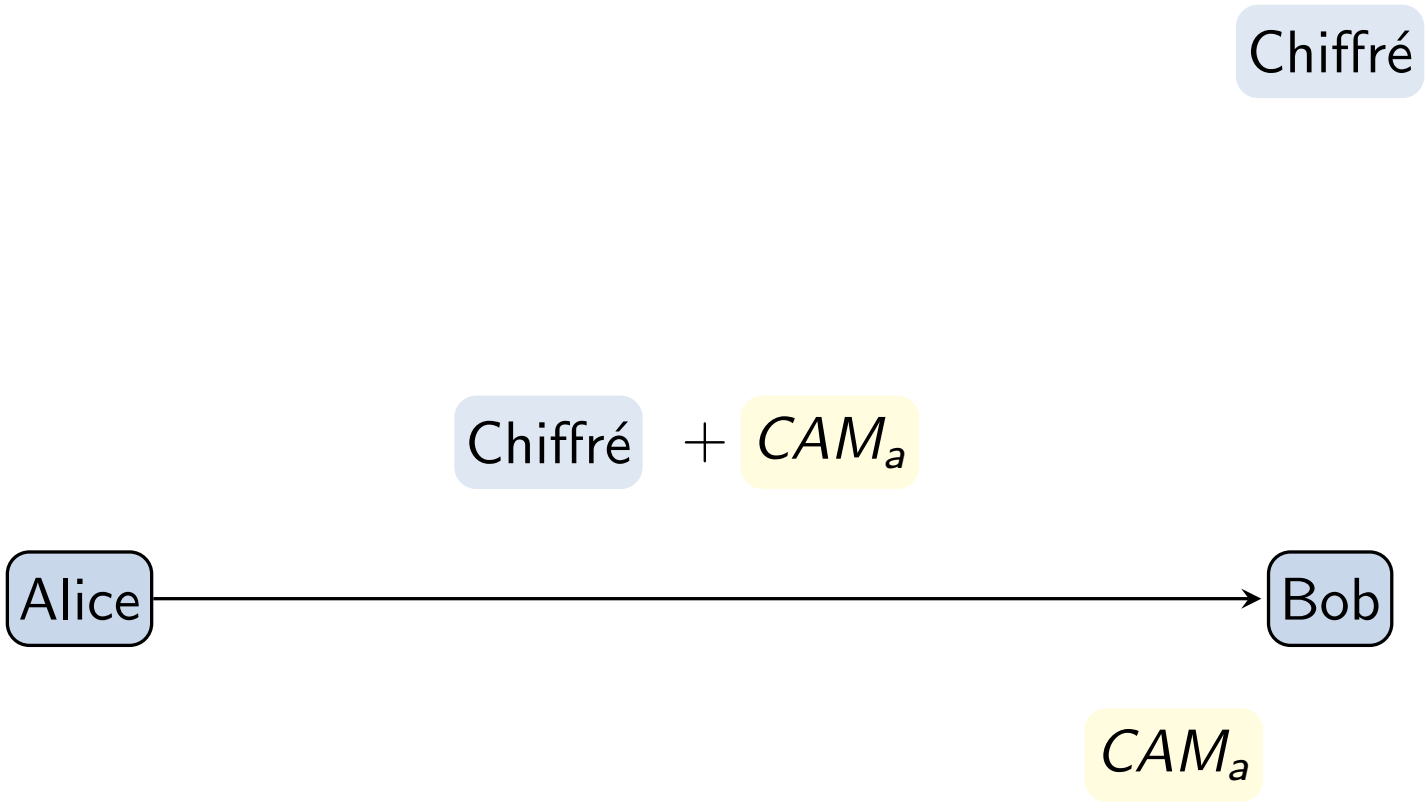


Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration

Clé secrète $\mathbf{k}_2$ Chiffré

Alice Bob

CAM_a

Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration

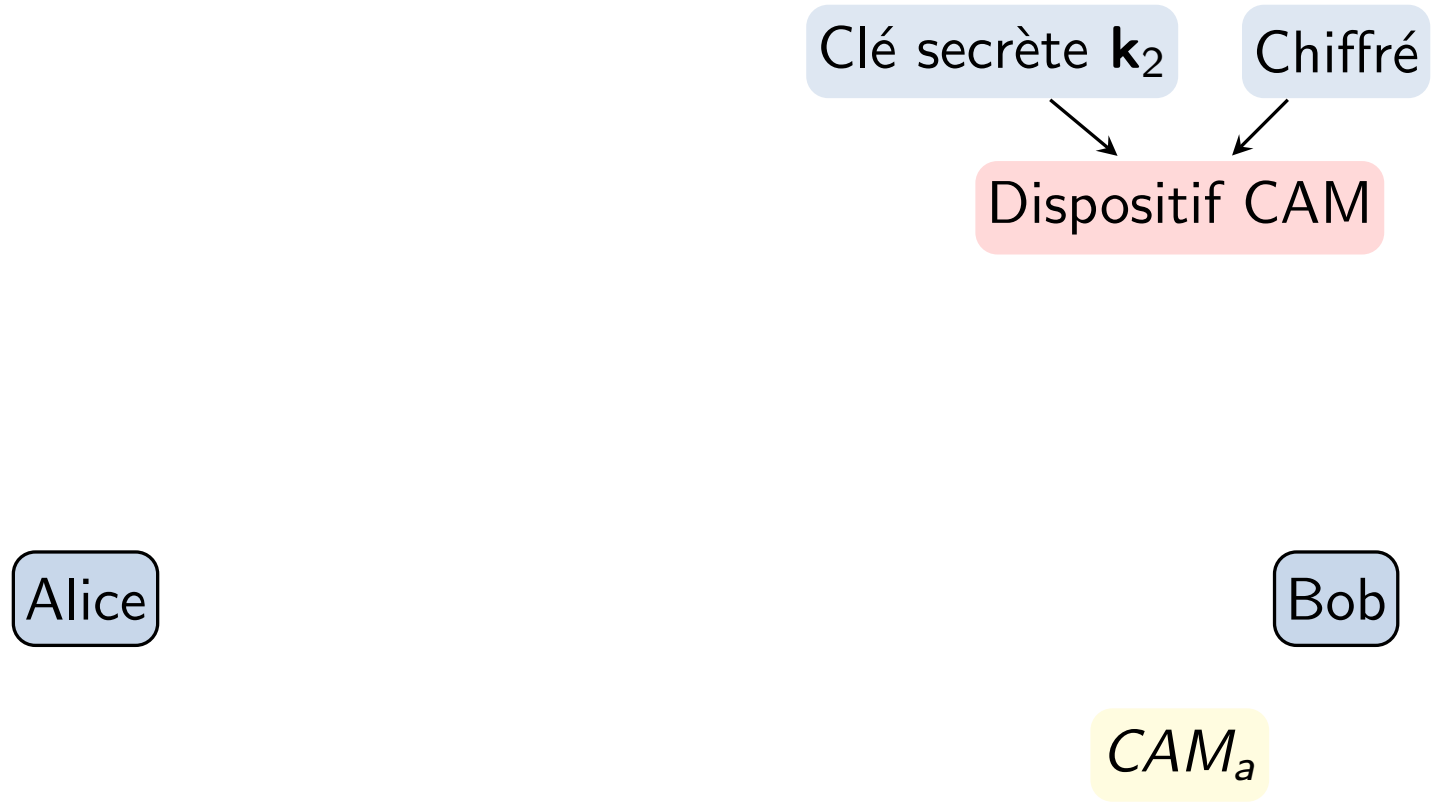


Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration

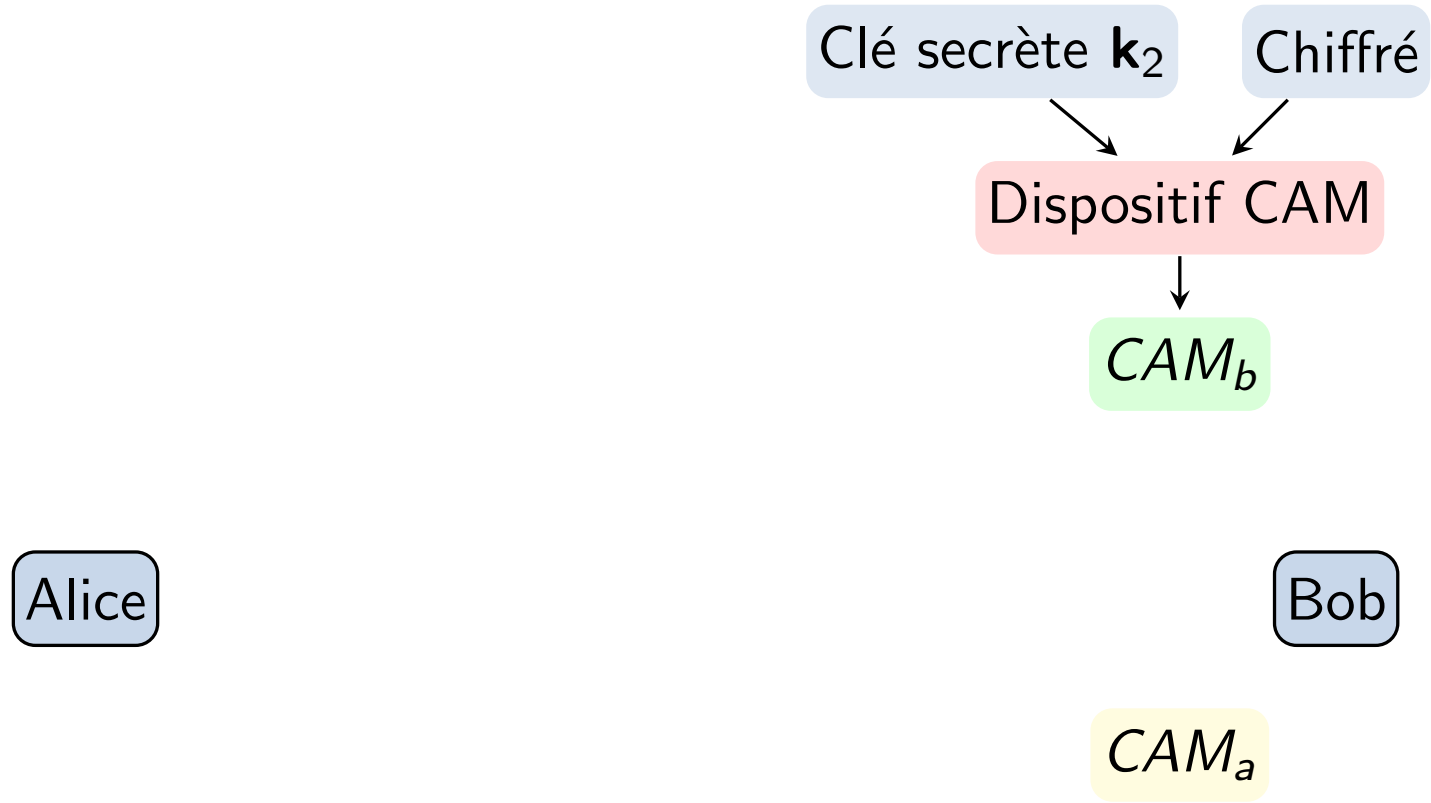


Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration

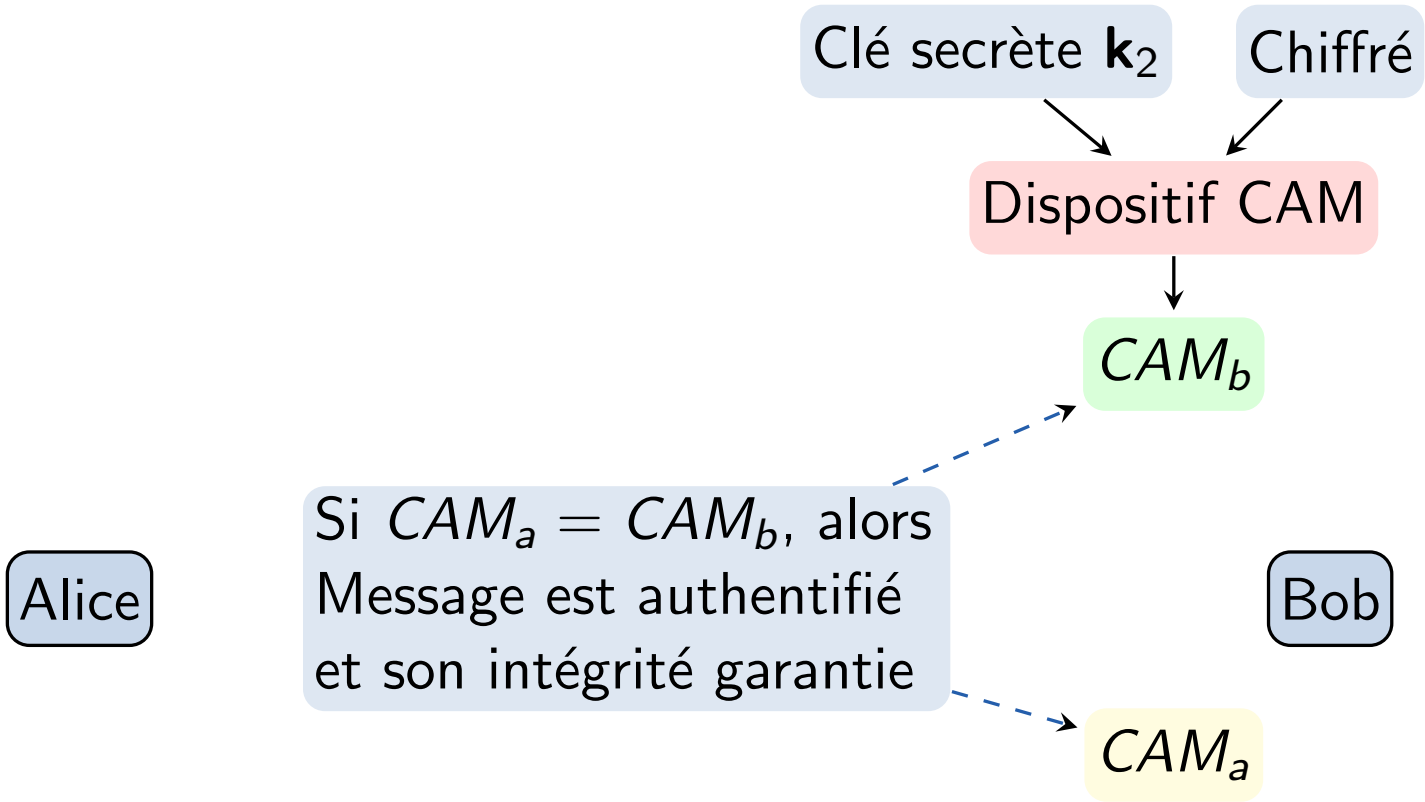


Schéma de fonctionnement - Solution 1

Encrypt-then-MAC configuration

Bob peut déchiffrer le message avec la clé secrète $\mathbf{k}_1$

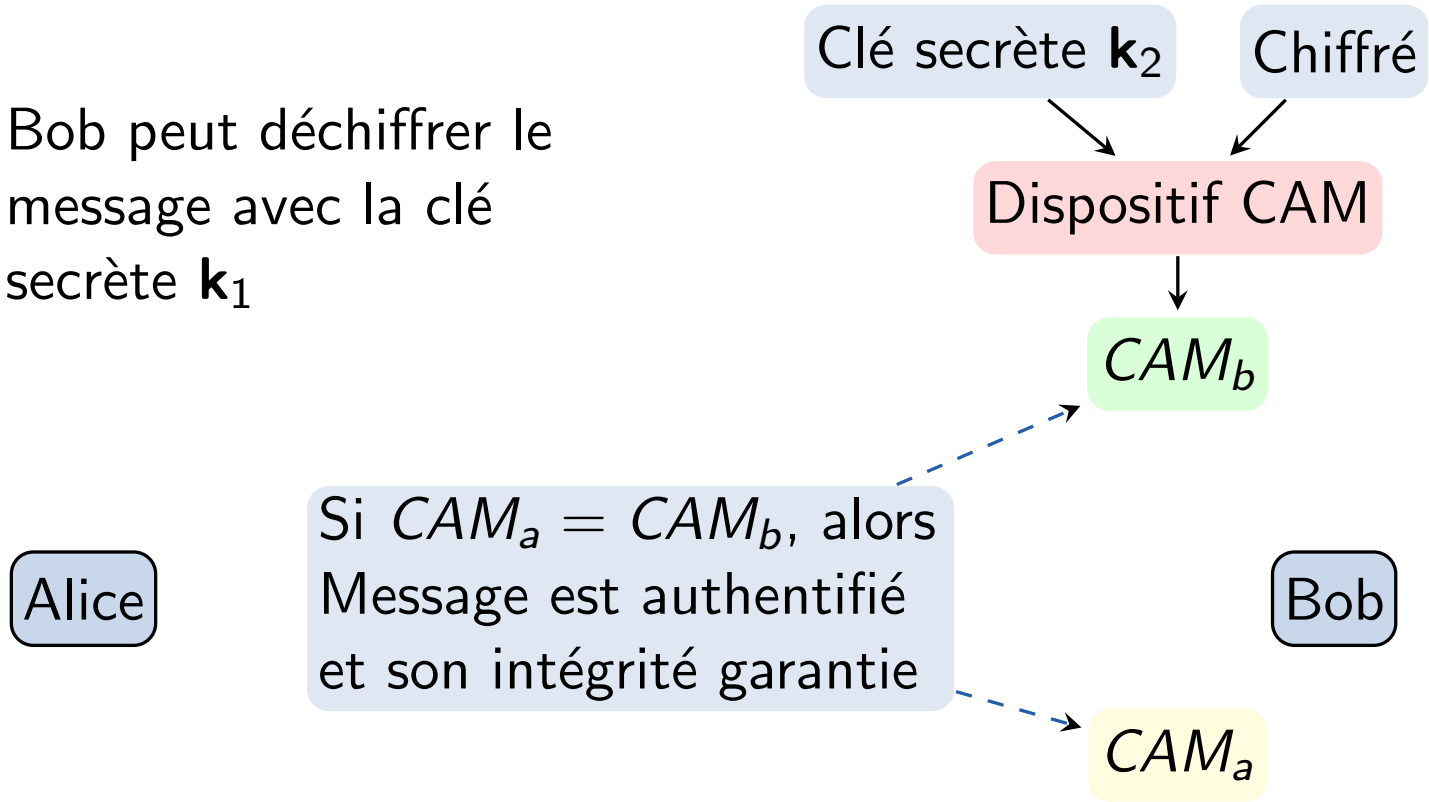


Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration

Alice

Bob

Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration

Message

Alice

Bob

Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration

Message | Clé secrète $\mathbf{k}_2$

Alice

Bob

Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration

Message | Clé secrète $\mathbf{k}_2$

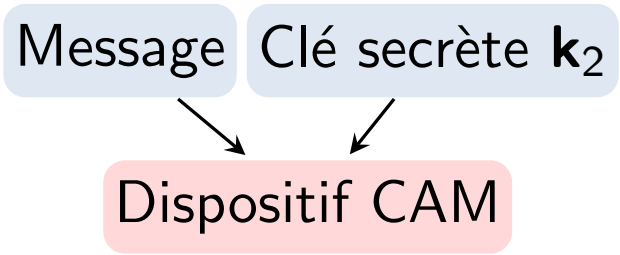
Dispositif CAM

Alice

Bob

Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration

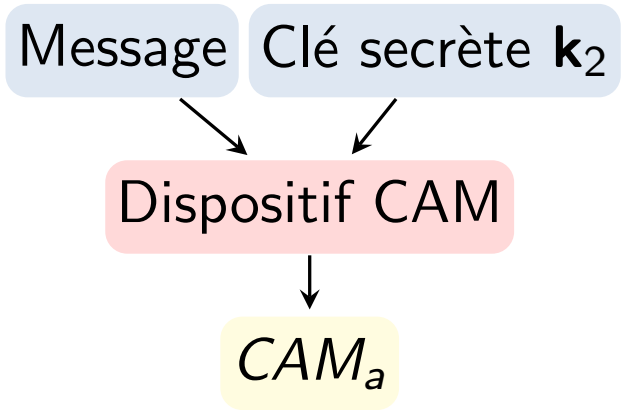


Alice

Bob

Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration



Alice

Bob

Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration

$$\text{Chiffré} = Enc_{\mathbf{k}_1}(\text{Message} \,||\, CAM_a)$$

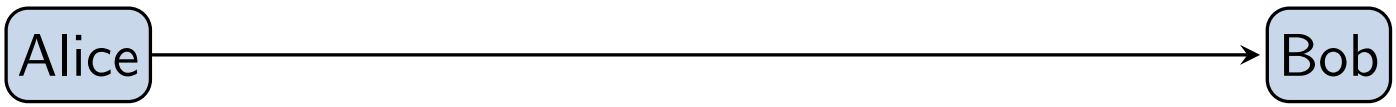
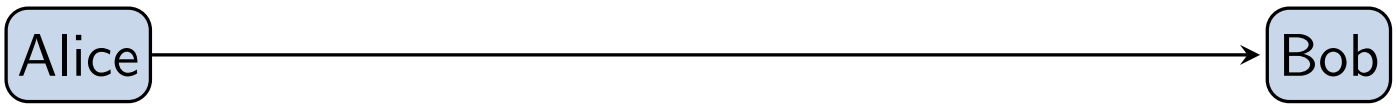


Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration

Message

$$\text{Chiffré} = Enc_{\mathbf{k}_1}(\text{Message} \,||\, CAM_a)$$



$$Dec_{\mathbf{k}_1}(\text{Chiffré}) = \text{Message} \,||\, CAM_a \qquad CAM_a$$

Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration

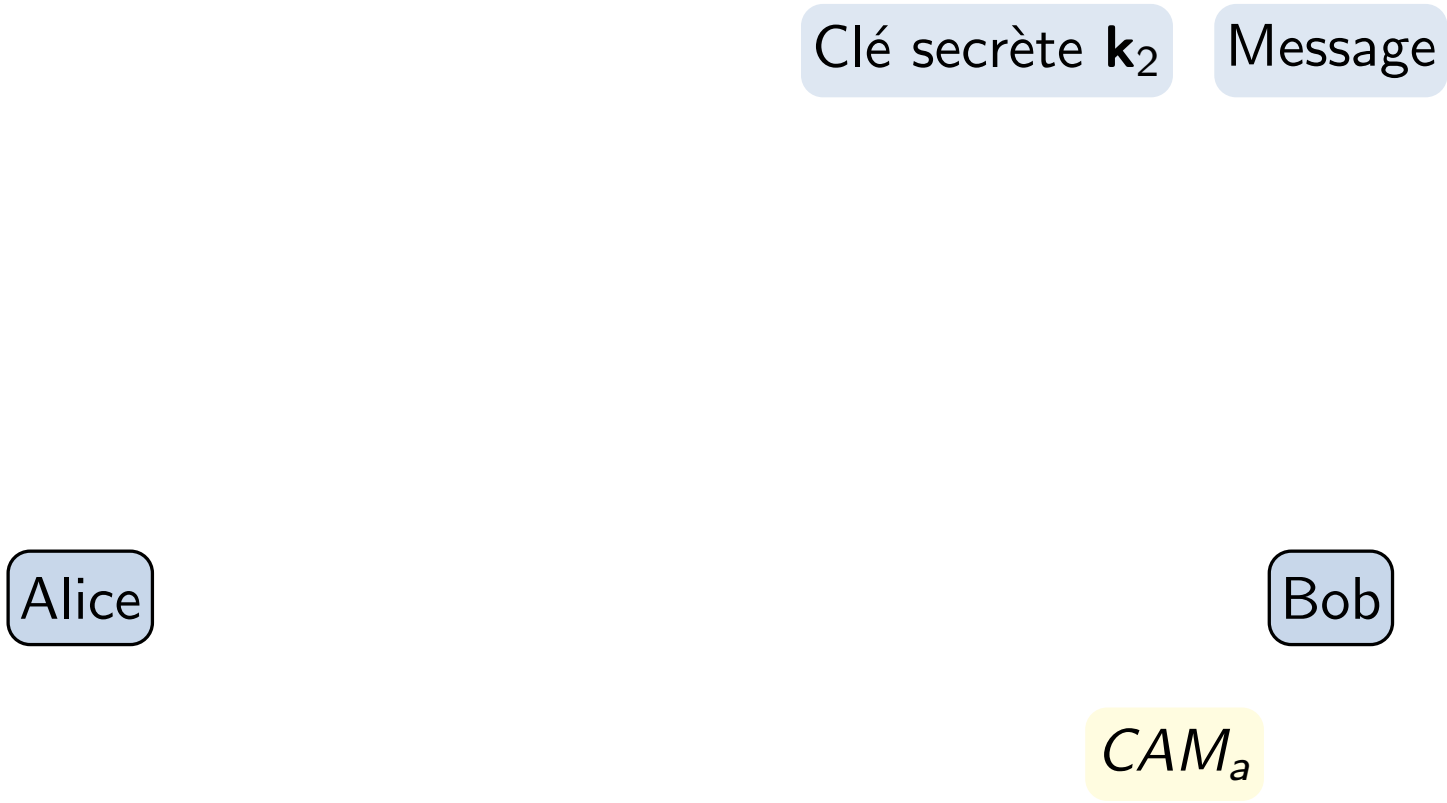


Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration

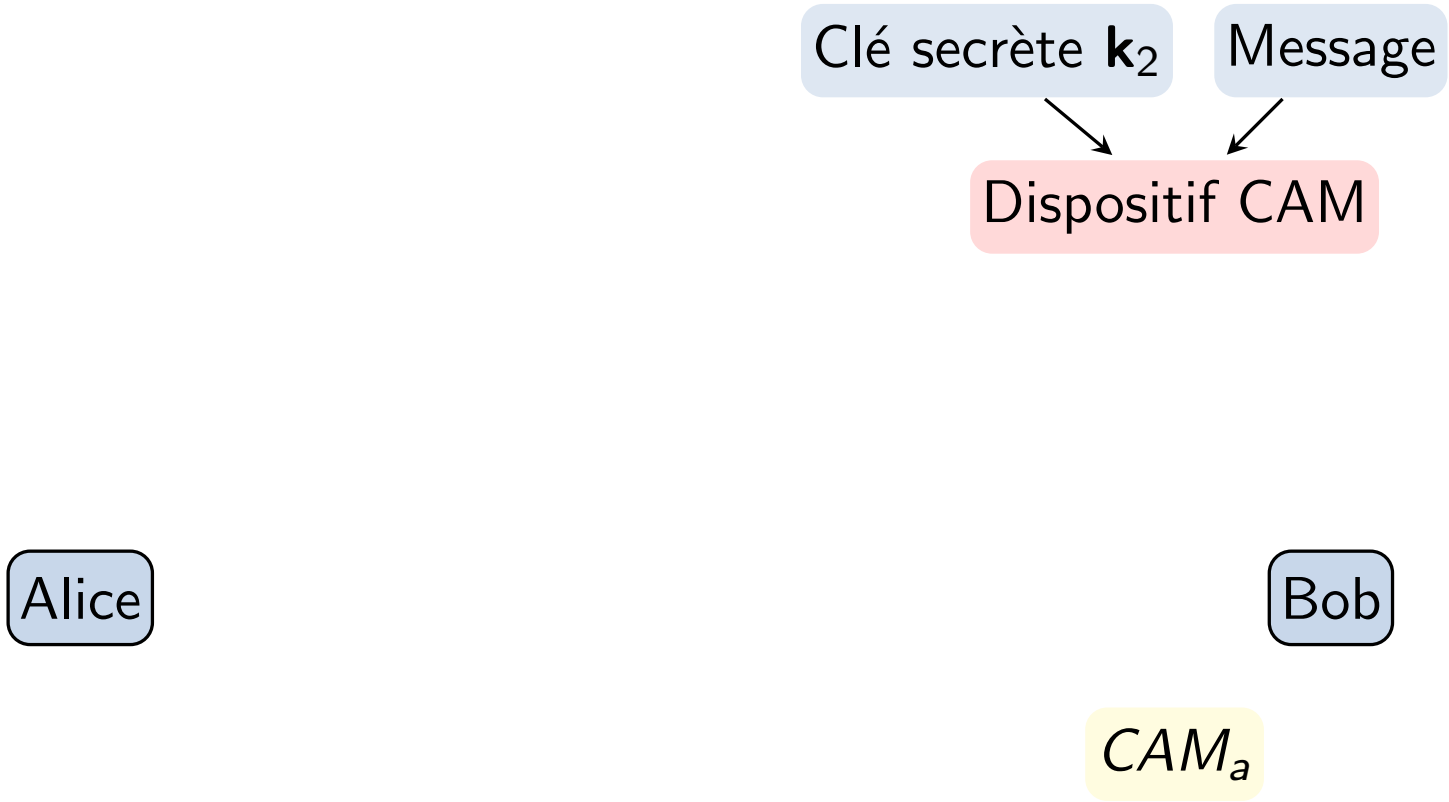


Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration

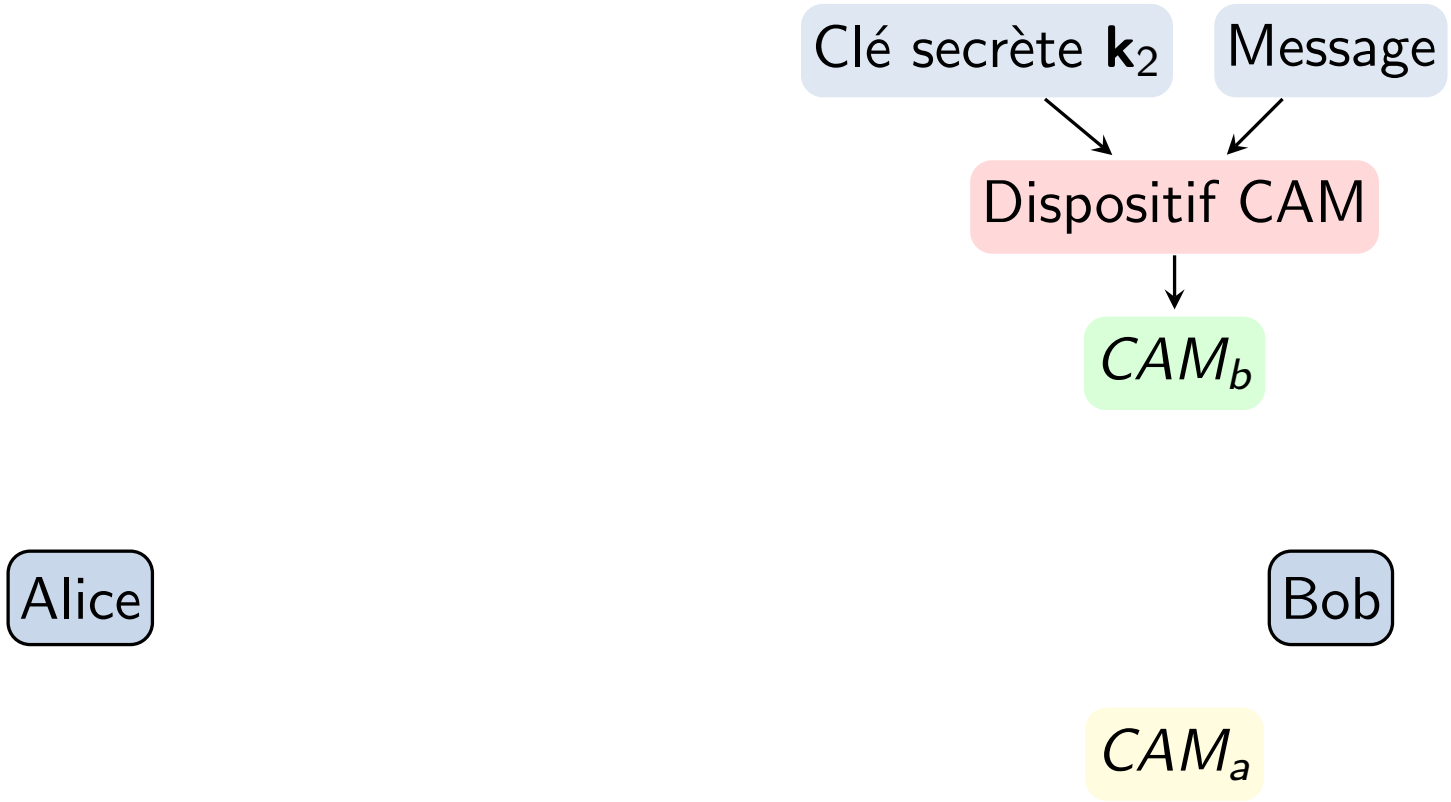
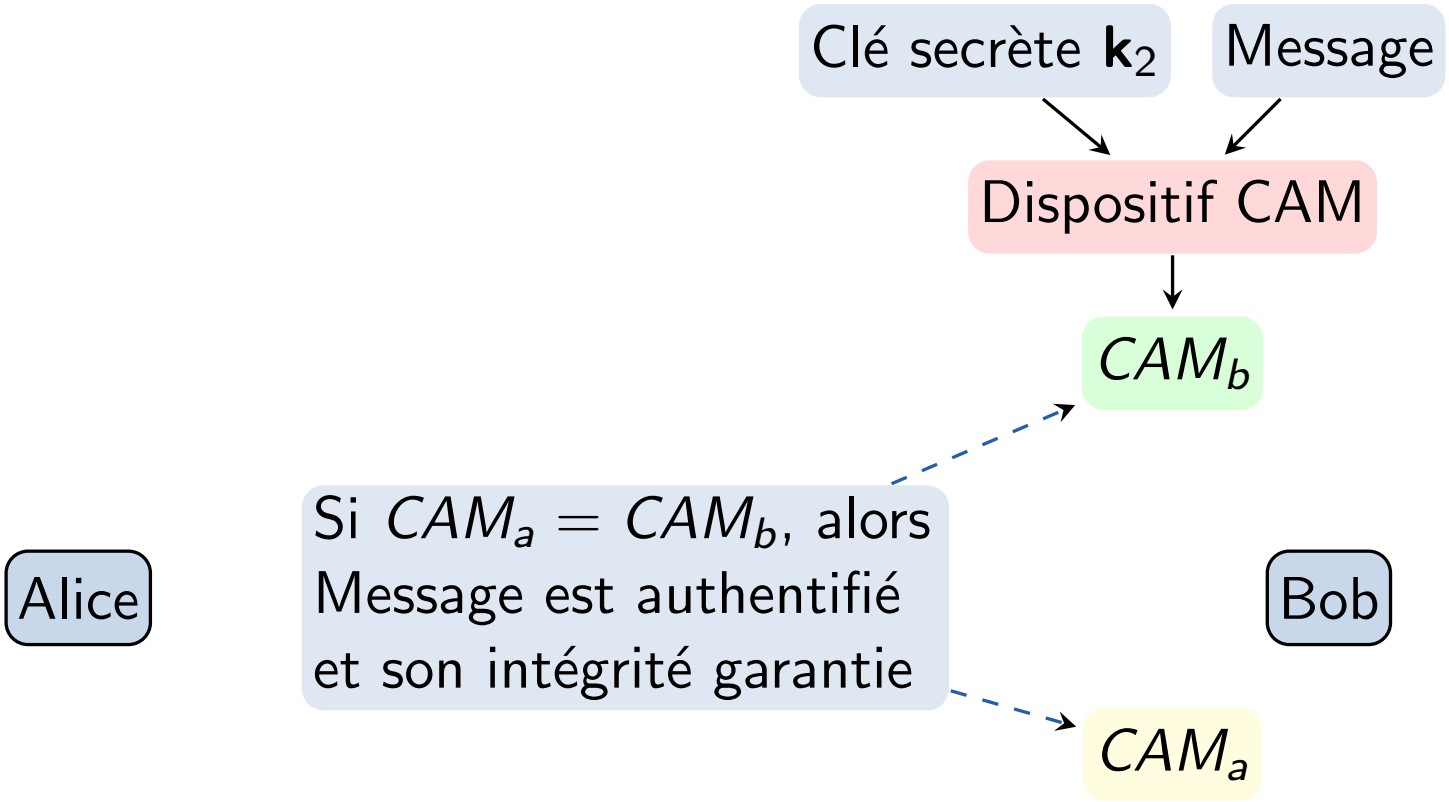


Schéma de fonctionnement - Solution 2

MAC-then-Encrypt configuration



Comment utiliser un CAM

	Solution 1	Solution 2
Avantages	Pas besoin de déchiffrer le message si on n'a pas pu authentifier → **potentiellement moins couteux pour Bob**	Un seul message est envoyé par Alice à Bob → **moins couteux pour le réseau**
Inconvéniants	Deux messages sont envoyés par Alice à Bob → **plus couteux pour le réseau**	Systématiquement, Bob doit déchiffrer le message et calculer le CAM → **plus couteux pour Bob**

CAM en pratique : CBC-MAC

CAM en pratique : CAM itératif

Sur le même principe que la fonction de hachage cryptographique itérative pour gérer la taille aléatoire du message

Soit $\mathbf{m} \in \{0,1\}^*$ le message à authentifier.
Soit $f_{\mathbf{k}} : \{0,1\}^n \times \{0,1\}^{n'} \to \{0,1\}^n$ une **fonction de compression** où la **sortie dépend de la clé secrète** (on a souvent $n = n'$).

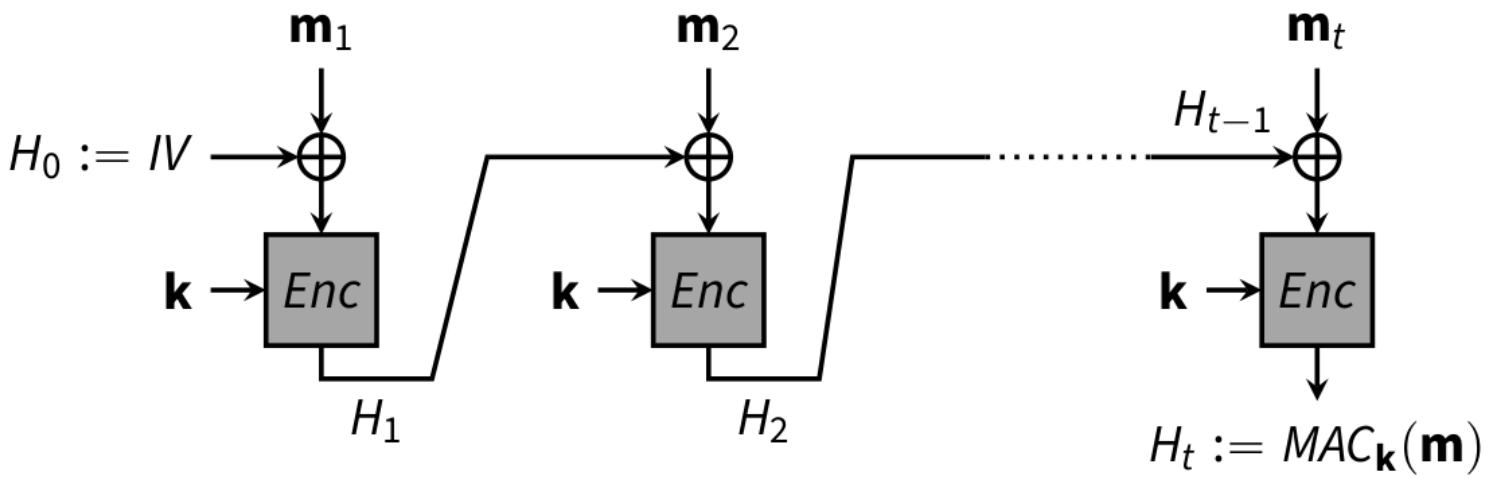
1. bourrage et découpage : découper le message $\mathbf{m}$ (avec bourrage pour être multiple de n') en t messages de taille n'.
2. $H_0 = IV$ *(vecteur d'initialisation)*
3. pour tout $1 \leq i \leq t$, on calcule récursivement $H_i = f_{\mathbf{k}}(H_{i-1}, \mathbf{m}_i)$
4. retourne H_t

CAM en pratique : CBC-MAC

CBC-MAC est un CAM itératif où $n' = n$ et la fonction de compression est :

$$f_{\mathbf{k}}(H_{i-1}, \mathbf{m}_i) = Enc_{\mathbf{k}}(H_{i-1} \oplus \mathbf{m}_i)$$

où $Enc_{\mathbf{k}}$ est un **chiffrement symétrique** comme AES.



CAM en pratique : CBC-MAC

Remarque : Le vecteur d'initialisation n'est pas un secret. Par simplicité, on suppose que $IV = 0$.

Attaque par falsification : trouver une paire valide $(\mathbf{m}, CAM_{\mathbf{k}}(\mathbf{m}))$ sans aucune connaissance sur la clé secrète $\mathbf{k}$.

Il existe deux attaques par falsification sur le CBC-MAC.

CAM en pratique : HMAC

CAM en pratique : basé sur une fonction de hachage

La propriété itérative est caché dans la fonction de hachage cryptographique

Soit H une fonction de hachage cryptographique itérative :

- CAM **préfixé** de la clé : $CAM_{\mathbf{k}}(\mathbf{m}) = H(\mathbf{k}||\mathbf{m})$
- CAM **suffixé** de la clé : $CAM_{\mathbf{k}}(\mathbf{m}) = H(\mathbf{m}||\mathbf{k})$
- CAM **enveloppé** de la clé : $CAM_{\mathbf{k}_1,\mathbf{k}_2}(\mathbf{m}) = H(\mathbf{k}_1||\mathbf{m}||\mathbf{k}_2)$

Si H ne résiste pas aux attaques par collision, alors il y a des attaques par falsification triviales.

CAM en pratique : HMAC

Définition : HMAC

$$HMAC_{\mathbf{k}}(\mathbf{m}) = H((\mathbf{k}_0 \oplus opad)||H((\mathbf{k}_0 \oplus ipad)||m))$$

où :

- $\mathbf{m}$: le message à authentifier
- H : une fonction de hachage cryptographique itérative avec une fonction de compression $h : \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ où $t \geq n$.
- $\mathbf{k}_0 = \begin{cases} (\mathbf{k}||0) & \text{si } len(\mathbf{k}) < t \\ (H(\mathbf{k})||0) & sinon \end{cases}$
  Donc la longueur de la clé secrète $\mathbf{k}_0$ est t.
- $ipad = 0x3636...36$ et $opad = 0x5C5C...5C$ sont tous les deux de taille t.

CAM en pratique : HMAC

- H est souvent SHA-256
- HMAC a été publiée pour la première fois en 1996 par Mihir Bellare, Ran Canetti, et Hugo Krawczyk (qui a écrit la RFC 2104)
- HMAC est utilisé dans IPsec, TLS, ...

Dans la RFC2104, à propos de HMAC il est dit :

- possible d'utiliser des fonctions de hachage cryptographique existante ;
- préserve les performances d'origine des fonctions de hachage ;
- utilise et manipule les clé facilement ;
- remplace facilement la fonction de hachage si celle-ci est compromise ou si de nouvelles versions plus rapide ou sûres seraient disponibles.

Authentification des entités

Authentification des entités

Nous pouvons utiliser la fonction de hachage cryptographique ou CAM pour authentifier les entités au lieu des messages. Attention à l'attaque "par rejeu". Eve intercepte un message et son CAM/Hash qui sont utilisés pour authentifier Alice. Plus tard, Eve renvoie le même message à Bob pour s'authentifier à la place d'Alice.

Pour éviter ce type d'attaque, Bob envoie d'abord un défi à Alice, qui est simplement un *nonce*. Alice intègre ce *nonce* à son message. Par la suite, Eve ne pourra pas s'authentifier à la place d'Alice car le CAM qu'elle a précédemment intercepté concernait un *nonce* particulier qui est modifié par Bob à chaque demande d'authentification.