

HAI709I - Fondements cryptographiques pour la sécurité

Cours 5 - MAC

Rocco Mora

6 Octobre, 2025

Université de Montpellier – Faculté des Sciences
M1 informatique, parcours Algo, IASD, Imagine

Confidentialité vs intégrité

Scénario 1 : Alice communique avec une banque via Internet. La banque reçoit une demande de transfert de 1000 euros du compte d'Alice vers celui de Bob.

- La demande est-elle **authentique** ? Autrement dit, la demande a-t-elle été envoyée par Alice ou par quelqu'un d'autre (peut-être Bob) ?
- Si la demande a été envoyée par Alice, est-elle correcte ou le montant du virement a-t-il été **modifié** ?

Scénario 2 : Un utilisateur et un serveur client (d'un magasin) communiquent via le protocole HTTP. Un **cookie** stocke les paires (article, prix) du panier de l'utilisateur, où le prix dépend de l'utilisateur (par exemple en raison de remises/codes promotionnels).

- Comment le magasin peut-il s'assurer que l'utilisateur ne modifie pas le cookie pour altérer les prix ?

Il faut garantir **l'intégrité/l'authentification** des messages.

Chiffrement à l'aide de chiffrements par flot et par bloc

- Considérons un texte chiffré obtenu en effectuant une opération XOR entre un masque pseudo-aléatoire et un message. Inverser n'importe quel bit dans le texte chiffré entraîne l'inversion du bit correspondant dans le message déchiffré.

Exemple : Le montant du transfert se trouve dans le message $m_1, m_2, \dots, m_{11}$. Supposons que l'adversaire sait que le montant du transfert est inférieur à 1000€. Alors $m_{11} = 0$. Étant donné $c_1, c_2, \dots, c_{11}$, l'adversaire génère le texte chiffré $c_1, c_2, \dots, c_{10}, c_{11} \oplus 1$. Son déchiffrement donne alors $m_1, m_2, \dots, m_{10}, m_{11} \oplus 1$ c'est-à-dire que $2^{10} = 1024\text{€}$ ont été ajoutés.

⚠ Il marche aussi pour l'OTP !

- En mode ECB, inverser un bit dans le i -ème bloc du texte chiffré n'affecte que le i -ème bloc du texte en clair
- En mode CBC, inverser le j -ème bit de IV ne modifie que le j -ème bit du premier bloc de message m_1 . Le premier bloc peut donc être modifié arbitrairement.

Pour les chiffrements par flot et par blocs, la confidentialité n'implique pas l'intégrité

Code d'authentification de message (MAC)

Un **code d'authentification de message** [🇬🇧 Message authentication code (MAC)] est un triplet d'algorithmes PPT (Gen, Mac, Vrfy) tel que :

- L'algorithme de génération de clé Gen prend en entrée le paramètre de sécurité 1^n et produit une clé k avec $|k| \geq n$.
- L'**algorithme de génération d'étiquette** [🇬🇧 Tag-generation algorithm] Mac prend en entrée une clé k et un message $m \in \{0, 1\}^*$ et produit une **étiquette** [🇬🇧 tag] $t \leftarrow \text{Mac}_k(m)$.
- L'**algorithme de vérification** déterministe Vrfy prend en entrée une clé k , un message m et une étiquette t . Il produit un bit $b := \text{Vrfy}_k(m, t)$. $b = 1$ signifie **valide**, $b = 0$ signifie **non valide**.

Pour chaque n , chaque clé k produite par $\text{Gen}(1^n)$ et chaque $m \in \{0, 1\}^*$, on a

$$\text{Vrfy}_k(m, \text{Mac}_k(m)) = 1.$$

- Mac **peut être aléatoire**.
- Si Mac_k n'est défini que pour $m \in \{0, 1\}^{\ell(n)}$, on dit que Mac a **longueur fixe** $\ell(n)$.

L'expérience d'authentification de message $\text{Mac-forge}_{\mathcal{A}, \Pi}(n)$

1. Une clé k est générée en exécutant $\text{Gen}(1^n)$.
2. L'adversaire $\mathcal{A}$ reçoit l'entrée 1^n et **accès à l'oracle** Mac_k . L'adversaire produit (m, t) . Soit $\mathcal{Q}$ l'ensemble des requêtes m' soumises à l'oracle par $\mathcal{A}$.
3. Le résultat de l'expérience est 1 (c'est-à-dire que $\mathcal{A}$ réussit) si et seulement si $\text{Vrfy}_k(m, t) = 1 \wedge m \notin \mathcal{Q}$.

Infalsifiabilité existentielle sous une attaque adaptative par message choisi

Un code d'authentification de message $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$ est **infalsifiable (de manière existentielle sous une attaque adaptative par message choisi)**, ou **sûr** [🇬🇧 existentially unforgeable under an adaptive chosen-message attack/secure], si pour tout adversaire PPT $\mathcal{A}$, il existe une fonction négligeable negl telle que

$$\Pr(\text{Mac-forge}_{\mathcal{A}, \Pi}(n) = 1) \leq \text{negl}(n).$$

Une définition trop stricte ?

Avoir **accès à un oracle** est une **hypothèse réaliste** : prenons par exemple un système qui stocke des fichiers avec leur MAC pour en vérifier l'intégrité. Si un adversaire peut télécharger des fichiers (avec les étiquettes correspondantes), il a effectivement accès à un oracle. Il ne devrait pas être en mesure de créer un MAC pour un fichier modifié.

Toutefois, l'adversaire pour la notion de sécurité MAC est très puissant :

- $\mathcal{A}$ peut interroger le MAC pour n'importe quel message
- il suffit de produire une étiquette valide pour n'importe quel message

Dans la réalité : généralement, seuls les messages "**significatifs**" sont authentifiés

→ une falsification n'est préjudiciable que si elle cible un message significatif (par exemple, des messages écrits en français)

Néanmoins, cette définition stricte du MAC peut être appliquée à un **large éventail d'objectifs**, par exemple pour authentifier des bases de données ou des données brutes. 5 / 12

Attaques par rejeu

Les MAC ne protègent pas contre les **attaques par rejeu** [🇬🇧 replay attacks] : l'attaquant renvoie simplement un message précédemment authentifié avec son tag valide.

Exemple : Alice envoie une demande de virement de 1000 euros à la banque pour Bob. Elle a calculé le tag et la banque vérifie que la demande est authentique.

Bob peut alors renvoyer le même message et la même étiquette pour obtenir à nouveau 1000 euros d'Alice.

Les attaques par rejeu doivent être gérées par une application de niveau supérieur

Une stratégie de prévention possible : l'expéditeur ajoute un **horodatage** [🇬🇧 timestamp] T . Le destinataire vérifie que t correspond à $m||T$ et que l'heure actuelle T' est proche de T . Cela nécessite que les deux parties restent bien **synchronisées**.

Infalsifiabilité forte

Que se passe-t-il si $\mathcal{A}$ génère une **étiquette différente** $t' \neq t$ pour un message précédemment authentifié ? $\rightarrow$ **Nouvelle expérience** $\text{Mac-sforge}_{\mathcal{A}, \Pi}$ comme $\text{Mac-forge}_{\mathcal{A}, \Pi}$, mais $\mathcal{Q}$ contient des paires de requêtes et de étiquettes associées.

Infalsifiabilité forte

Un code d'authentification de message $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$ est **fortement infalsifiable**, ou **fortement sûr** [🇬🇧 strongly unforgeable/strongly secure] si, pour tout adversaire PPT $\mathcal{A}$, il existe une fonction négligeable negl telle que

$$\Pr(\text{Mac-sforge}_{\mathcal{A}, \Pi}(n) = 1) \leq \text{negl}(n).$$

Vérification canonique : Si un MAC est déterministe, alors Vrfy peut simplement vérifier si $t = \text{Mac}_k(m)$.

Proposition

Un MAC déterministe sûr avec **vérification canonique** est fortement sûr.

MAC sûr à partir d'une PRF

Construction

Soit F une fonction pseudo-aléatoire préservant la longueur. Définissons un MAC de longueur fixe pour les messages de longueur n comme suit.

- Mac : en entrée une clé $k \in \{0, 1\}^n$ et un message $m \in \{0, 1\}^n$, en sortie l'étiquette $t := F_k(m)$.
- Vrfy : en entrée une clé $k \in \{0, 1\}^n$, un message $m \in \{0, 1\}^n$ et une étiquette $t \in \{0, 1\}^n$, renvoie 1 si et seulement si $t = F_k(m)$.

Théorème

Si F est une PRF, alors la construction ci-dessus est un **MAC sûr à longueur fixe pour les messages de longueur n** .

- + construction simple
- ne traite que les messages courts de longueur fixe et est inefficace → **pas utilisé en pratique**

MACs pas sûr pour les messages de longueur arbitraire

MAC pas sûr de longueur arbitraire à partir du $(\text{Mac}', \text{Vrfy}')$ sûr de longueur fixe n .

MACs pas sûr pour les messages de longueur arbitraire

MAC pas sûr de longueur arbitraire à partir du $(\text{Mac}', \text{Vrfy}')$ sûr de longueur fixe n .

Idée 1 Décomposer m en séquence de blocs de n bits $m_1, \dots, m_d$ et authentifier chaque bloc séparément, c.-à-d. que l'étiquette est $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'_k(m_i)$.

MACs pas sûr pour les messages de longueur arbitraire

MAC pas sûr de longueur arbitraire à partir du $(\text{Mac}', \text{Vrfy}')$ sûr de longueur fixe n .

Idée 1 Décomposer m en séquence de blocs de n bits $m_1, \dots, m_d$ et authentifier chaque bloc séparément, c.-à-d. que l'étiquette est $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'_k(m_i)$.

Attaque par réorganisation des blocs : sachant que m_1, m_2 ont l'étiquette (t_1, t_2) , $\mathcal{A}$ peut authentifier m_2, m_1 avec (t_2, t_1) .

MACs pas sûr pour les messages de longueur arbitraire

MAC pas sûr de longueur arbitraire à partir du $(\text{Mac}', \text{Vrfy}')$ sûr de longueur fixe n .

Idée 1 Décomposer m en séquence de blocs de n bits $m_1, \dots, m_d$ et authentifier chaque bloc séparément, c.-à-d. que l'étiquette est $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'_k(m_i)$.

Attaque par réorganisation des blocs : sachant que m_1, m_2 ont l'étiquette (t_1, t_2) , $\mathcal{A}$ peut authentifier m_2, m_1 avec (t_2, t_1) .

Idée 2 Ajouter un indice, c.-à-d. $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'(i || m_i)$.

MACs pas sûr pour les messages de longueur arbitraire

MAC pas sûr de longueur arbitraire à partir du $(\text{Mac}', \text{Vrfy}')$ sûr de longueur fixe n .

Idée 1 Décomposer m en séquence de blocs de n bits $m_1, \dots, m_d$ et authentifier chaque bloc séparément, c.-à-d. que l'étiquette est $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'_k(m_i)$.

Attaque par réorganisation des blocs : sachant que m_1, m_2 ont l'étiquette (t_1, t_2) , $\mathcal{A}$ peut authentifier m_2, m_1 avec (t_2, t_1) .

Idée 2 Ajouter un indice, c.-à-d. $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'(i || m_i)$.

Attaque par troncature : sachant que m_1, m_2 ont l'étiquette (t_1, t_2) , $\mathcal{A}$ peut authentifier m_1 avec t_1 , c'est-à-dire que $\mathcal{A}$ peut supprimer les derniers blocs.

MACs pas sûr pour les messages de longueur arbitraire

MAC pas sûr de longueur arbitraire à partir du $(\text{Mac}', \text{Vrfy}')$ sûr de longueur fixe n .

Idée 1 Décomposer m en séquence de blocs de n bits $m_1, \dots, m_d$ et authentifier chaque bloc séparément, c.-à-d. que l'étiquette est $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'_k(m_i)$.

Attaque par réorganisation des blocs : sachant que m_1, m_2 ont l'étiquette (t_1, t_2) , $\mathcal{A}$ peut authentifier m_2, m_1 avec (t_2, t_1) .

Idée 2 Ajouter un indice, c.-à-d. $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'(i || m_i)$.

Attaque par troncature : sachant que m_1, m_2 ont l'étiquette (t_1, t_2) , $\mathcal{A}$ peut authentifier m_1 avec t_1 , c'est-à-dire que $\mathcal{A}$ peut supprimer les derniers blocs.

Idée 3 Ajouter la longueur du message, c.-à-d. $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'(\ell || i || m_i)$.

MACs pas sûr pour les messages de longueur arbitraire

MAC pas sûr de longueur arbitraire à partir du $(\text{Mac}', \text{Vrfy}')$ sûr de longueur fixe n .

Idée 1 Décomposer m en séquence de blocs de n bits $m_1, \dots, m_d$ et authentifier chaque bloc séparément, c.-à-d. que l'étiquette est $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'_k(m_i)$.

Attaque par réorganisation des blocs : sachant que m_1, m_2 ont l'étiquette (t_1, t_2) , $\mathcal{A}$ peut authentifier m_2, m_1 avec (t_2, t_1) .

Idée 2 Ajouter un indice, c.-à-d. $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'(i || m_i)$.

Attaque par troncature : sachant que m_1, m_2 ont l'étiquette (t_1, t_2) , $\mathcal{A}$ peut authentifier m_1 avec t_1 , c'est-à-dire que $\mathcal{A}$ peut supprimer les derniers blocs.

Idée 3 Ajouter la longueur du message, c.-à-d. $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'(\ell || i || m_i)$.

Attaque "mix-and-match" : sachant que $m_1, \dots, m_d$ a l'étiquette $(t_1, \dots, t_d)$ et que $m'_1, \dots, m'_d$ a l'étiquette $(t'_1, \dots, t'_d)$, $\mathcal{A}$ peut authentifier $m_1, m'_2, m_3, m'_4, \dots$ avec $(t_1, t'_2, t_3, t'_4, \dots)$.

MACs pas sûr pour les messages de longueur arbitraire

MAC pas sûr de longueur arbitraire à partir du $(\text{Mac}', \text{Vrfy}')$ sûr de longueur fixe n .

Idée 1 Décomposer m en séquence de blocs de n bits $m_1, \dots, m_d$ et **authentifier chaque bloc séparément**, c.-à-d. que l'étiquette est $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'_k(m_i)$.

Attaque par réorganisation des blocs : sachant que m_1, m_2 ont l'étiquette (t_1, t_2) , $\mathcal{A}$ peut authentifier m_2, m_1 avec (t_2, t_1) .

Idée 2 Ajouter un indice, c.-à-d. $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'(i || m_i)$.

Attaque par troncature : sachant que m_1, m_2 ont l'étiquette (t_1, t_2) , $\mathcal{A}$ peut authentifier m_1 avec t_1 , c'est-à-dire que $\mathcal{A}$ peut supprimer les derniers blocs.

Idée 3 Ajouter la longueur du message, c.-à-d. $t = (t_1, \dots, t_d)$ avec $t_i := \text{Mac}'(\ell || i || m_i)$.

Attaque "mix-and-match" : sachant que $m_1, \dots, m_d$ a l'étiquette $(t_1, \dots, t_d)$ et que $m'_1, \dots, m'_d$ a l'étiquette $(t'_1, \dots, t'_d)$, $\mathcal{A}$ peut authentifier $m_1, m'_2, m_3, m'_4, \dots$ avec $(t_1, t'_2, t_3, t'_4, \dots)$.

La dernière attaque peut être évitée grâce à un **identifiant de message**

Construction

Soit $(\text{Mac}', \text{Vrfy}')$ un MAC de longueur fixe n . Définissons le MAC :

- **Mac** : en entrée une clé $k \in \{0, 1\}^n$ et un message $m \in \{0, 1\}^*$ de longueur $0 < \ell < 2^{n/4}$, décomposer m en d blocs $m_1, \dots, m_d$ de longueur $n/4$ (en ajoutant éventuellement un remplissage au dernier). Choisir un **identifiant de message** uniforme $r \in \{0, 1\}^{n/4}$.

Pour $i = 1, \dots, d$, calculer $t_i \leftarrow \text{Mac}'_k(r || \ell || i || m_i)$, avec i, ℓ codés sous forme de chaînes de $(n/4)$ bits. Afficher $t := (r, t_1, \dots, t_d)$.

- **Vrfy** : en entrée une clé $k \in \{0, 1\}^n$, un message $m \in \{0, 1\}^*$ de longueur $0 < \ell < 2^{n/4}$ et une étiquette $t = (r, t_1, \dots, t_{d'})$, décomposer m en d blocs $m_1, \dots, m_d$, chacun de longueur $n/4$ (en complétant éventuellement le dernier). Sortie 1 si et seulement si $d' = d$ et $\text{Vrfy}'_k(r || \ell || i || m_i, t_i) = 1$ pour $i = 1, \dots, d$.

Théorème

Si Π' est un MAC sûr de longueur fixe, alors la construction ci-dessus est un **MAC sûr de longueur arbitraire**.

CBC-MAC

La construction précédente est très inefficace : l'étiquette d'un message de longueur $nd/4$ fait plus de $nd/4$ bits et nécessite d évaluations d'un chiffrement par blocs.

Construction (CBC-MAC)

Soit F une PRF, et fixons une fonction $\ell = \ell(n) > 0$. Le CBC-MAC de base est :

- Mac : en entrée une clé $k \in \{0, 1\}^n$ et un message de longueur $\ell \cdot n$,
 1. Décomposer $m = m_1, \dots, m_\ell$, avec m_i de longueur n .
 2. Définir $t_0 := 0^n$ et $t_i := F_k(t_{i-1} \oplus m_i)$. L'étiquette est t_ℓ .
- Vrfy : en entrée une clé $k \in \{0, 1\}^n$, un message m et une étiquette t , renvoyer 1 si et seulement si $(m \text{ a une longueur } \ell n) \wedge t = \text{Mac}_k(m)$.

Théorème

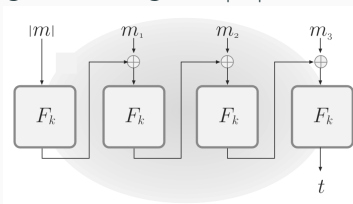
Soit ℓ un polynôme. Si F est une PRF, alors la construction ci-dessus est un MAC sûr pour les messages de longueur $\ell(n) \cdot n$.

Le CBC-MAC de base ne nécessite que ℓ évaluations PRF et l'étiquette a longueur n

CBC-MAC pour les messages de longueur arbitraire

La construction précédente peut être modifiée pour traiter des messages de **longueur arbitraire**, en restant **sûre**. Nous présentons deux possibilités :

1. Ajouter au début du message m sa longueur $|m|$



2. Choisir deux clés uniformes indépendantes $k_1, k_2 \in \{0, 1\}^n$. Calculer d'abord le CBC-MAC de base t de m en utilisant k_1 , puis afficher l'étiquette $t' = F_{k_2}(t)$.

CBC-MAC **standardisé** dans les années 1980 et adopté dans plusieurs normes **Internet**

Pour aller plus loin : **GMAC**, **Poly1305** offrent de bien meilleures performances et des étiquettes plus courtes.