Crypto à clé secrète

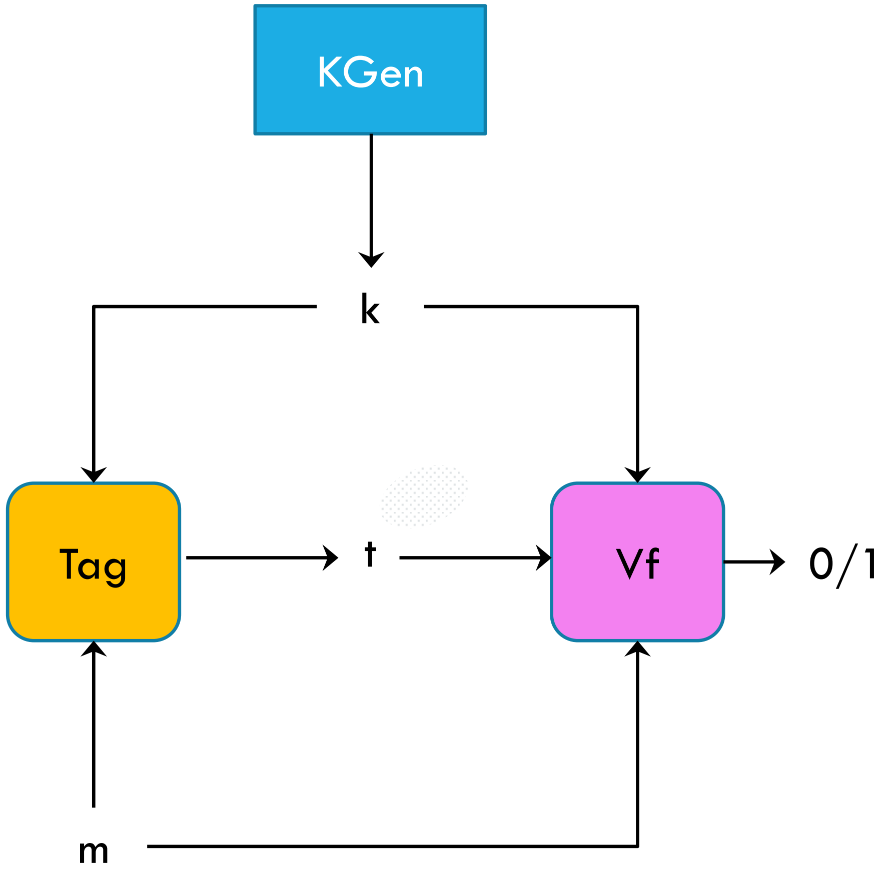
L'authentification de messages

Cristina Onete
cristina.onete@gmail.com

L'AUTHENTIFICATION DE MESSAGES : MA

- Le prouveur et le vérificateur partagent une clé symétrique
  - soit partagée hors connexion
  - soit le résultat d'un protocole d'échange de clé authentifié

- Syntaxe : un triplet d'algorithmes $(\mathrm{KGen}, \mathrm{Tag}, \mathrm{Vf})$ tel que :
  - $\mathrm{KGen}(1^\lambda) \to k$
  - $\mathrm{Tag}(k; m) \to t$
  - $\mathrm{Vf}(k; m, t) \to b \in \{0,1\}$: 1 = accept; 0 = reject



LA SÉCURITÉ DANS L'AUTHENTIFICATION

- Un attaquant ne peut pas authentifier un message "frais"
  - C'est à dire un message qui n'a pas été authentifié en préalable par une des parties honnêtes

- Modèle de l'attaquant :
  - L'adversaire a accès à un oracle de generation de Tag : $\mathrm{oTag}(\cdot)$ qui exécute Tag à boîte noire et retourne t
  - Ceci modélise l'accès que l'attaquant pourrait avoir à des tuples (m, t)

LA SÉCURITÉ DES SCHÉMAS DE MA

- Notion appellée Existential Unforgeability against Chosen Message Attacks (EUF-CMA)
- Le jeu de sécurité commence quand le challenger génère la clé
- Ensuite, l'adversaire peut utiliser son oracle oTag et finit par sortir un tuple (m, t)

$$k \leftarrow \mathrm{KGen}(1^\lambda)$$
$$(m, t) \leftarrow A^{\mathrm{oTag}(\cdot)}(\lambda)$$

A gagne ssi. :

$$\mathrm{Vf}(k; m, t) = 1$$

et m non-envoyé à oTag

- L'adversaire gagne si t vérifie pour m et k, et si m est frais

LE SCHÉMAS DE MAC

- Une instanciation particulière des schémas d'authentification de messages, dans laquelle :
  - l'algorithme $\mathrm{Tag}(\cdot\,;\cdot)$ est remplacé par un algorithme $\mathrm{Mac}(\cdot\,;\cdot)$
  - la vérification $\mathrm{Vf}(k; m, t)$ exécute $t^* \leftarrow \mathrm{Mac}(k; m)$ et vérifie si $t^* = t$

- Dans le jeu EUF-CMA l'attaquant aura accès par l'oracle oTag à :
  - l'algorithme de generation de tags et
  - l'algorithme de vérification

HMAC : MAC UTILISANT LE HACHAGE

- Nous allons prendre une function de hachage $H_x : D \rightarrow \{0,1\}^\ell$, pour une clé connue x
- Pour simplifier les choses nous allons omettre la clé x

- La construction de HMAC peut utiliser des clés k d'un grand espace de clés K
  - Attention : les clés de la fonction HMAC ne sont pas les clés de la fonction de hachage !
  - L'algorithme KGen choisit une clé aléatoirement de l'espace K (la taille dépend de 1^λ)
  - L'algorithme $\mathrm{HMAC}(k; m)$ est défini par :

$$\mathrm{HMAC}(k; m) = H\left(\hat{k} \oplus ipad \mid H\left((\hat{k} \oplus opad) \middle| m\right)\right)$$

  - Si $k \geq 2^\ell$, alors $\hat{k} = H(k)$; sinon $\hat{k} = k$
  - ipad et opad sont des constantes connues

LA SÉCURITÉ DE HMAC

- Un cas particulier d'un autre schéma : NMAC, pour lequel :
$$NMAC(k; m) = H(K_1|H(K_2|m))$$
- **<u>Théorème</u>** : HMAC est EUF-CMA si la fonction de hachage est un oracle aléatoire.
- **<u>Preuve</u>** : Nous allons remplacer la fonction de hachage H par un oracle aléatoire. Toute partie (honnête ou malhonnête) devra ensuite faire une requête à cet oracle pour faire ses calcules de hachage.

Stratégie de preuve : dans son jeu, l'attaquant devra pouvoir accéder à son oracle $\mathrm{oMAC}(\cdot)$ -- qui sert pour générer des tags **et** pour vérifier. Dans le ROM, oMAC fait 2 requêtes au RO : $\mathrm{r}_1 \leftarrow RO\big((\hat{k} \oplus opad)\big|m\big)$ et $\mathrm{r}_2 \leftarrow RO\big((\hat{k} \oplus ipad)\big|r_1\big)$.

Nous allons faire notre preuve par une suite de jeux, dans laquelle on va restreindre la capacité de l'adversaire peu à peu, tout en montrant que nos restrictions n'affectent pas (trop) sa capacité de gagner.

INTERMEZZO : PREUVES DE SÉCURITÉ

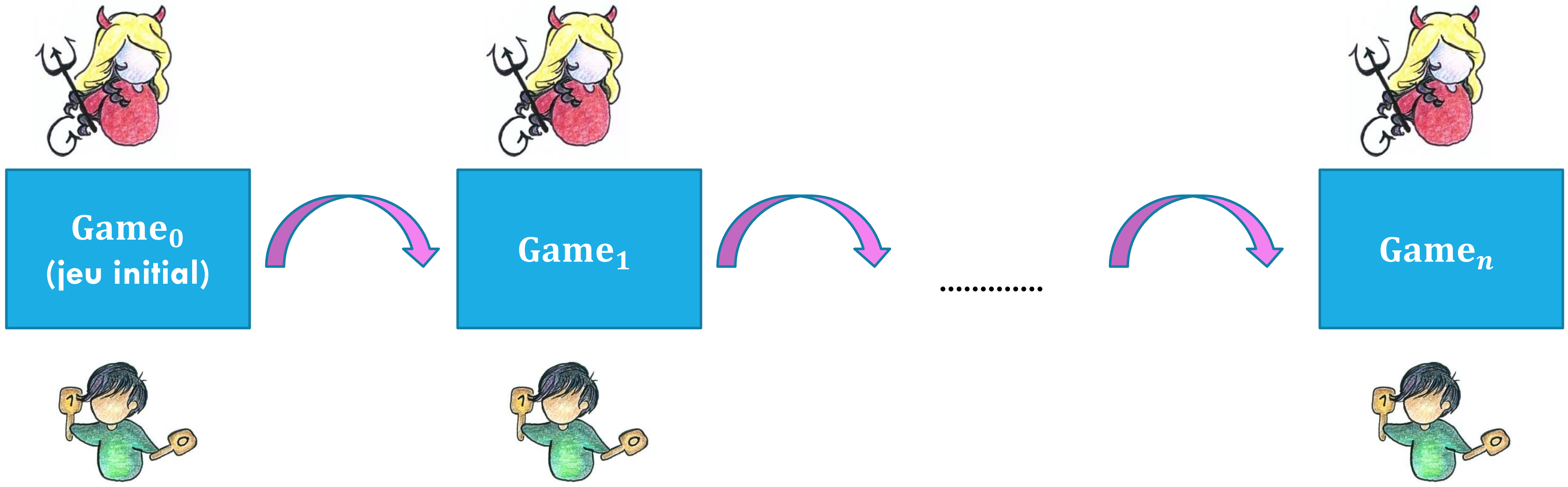
LES PREUVES DANS LA SÉCURITÉ PROUVABLE

- Dans la sécurité prouvable nous devons **prouver** la sécurité des primitives/protocoles
  - Par rapport à un **modèle de sécurité** : un jeu de sécurité, un attaquant, une syntaxe…

- Une preuve simple peut être réalisée dans une seule étape…
  - Mais la plupart de preuves ont plusieurs étapes
  - Chaque étape s'appelle un "game hop" [Shoup06]
  - Un game hop : la transition d'un jeu initial G_i vers un jeu très similaire G_{i+1}, mais dans lequel on limite les possibilités d'attaque d'un adversaire
  - Pour faire la preuve nous devons montrer que, du point de vue de l'adversaire, les deux jeux sont "équivalents"

Bib : *Victor Shoup : "Sequences of Games : a tool for taming complexity in security proofs", 2006*

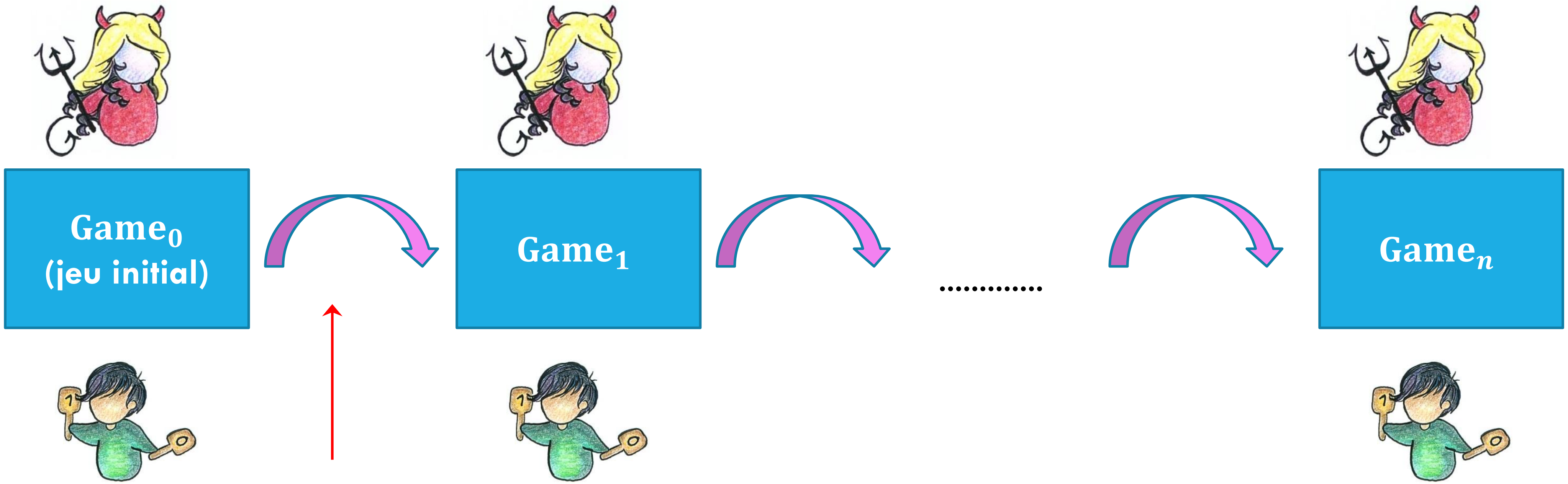
GAME HOPPING EN PERSPECTIVE



Je ne sais rien sur la probabilité de gagner de l'adversaire

Je peux estimer la probabilité de gagner de l'adversaire

GAME HOPPING EN PERSPECTIVE



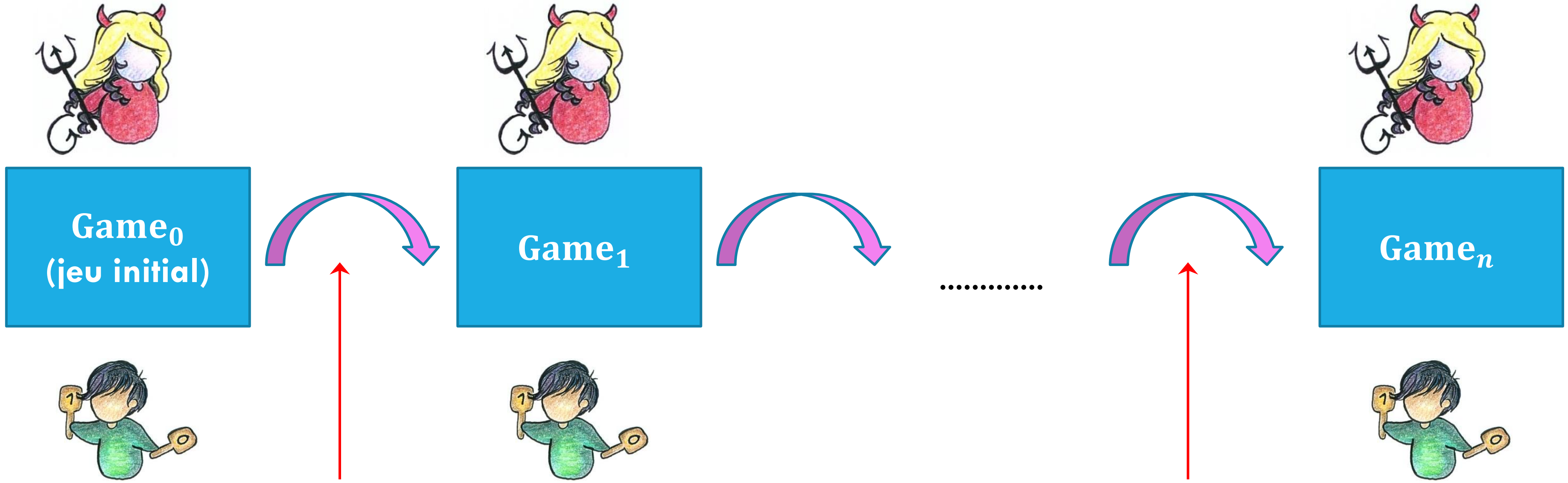
J'estime la différence dans la proba de gagner

$$\Pr[A\ gagne\ G_0] \leq \Pr[A\ gagne\ G_1] + \epsilon$$

ou

$$\Pr[A\ gagne\ G_0] \leq \epsilon \cdot \Pr[A\ gagne\ G_1]$$

GAME HOPPING : A LA FIN



$$\Pr[A\ gagne\ G_0] \leq \Pr[A\ gagne\ G_1] + \epsilon_{0,1}$$

$$\Pr[A\ gagne\ G_{n-1}] \leq \epsilon_{n-1,n} \cdot \Pr[A\ gagne\ G_n]$$

Cela donne $\boxed{\Pr[A\ gagne\ G_0] \leq \epsilon_{n-1,n} \cdot \Pr[A\ gagne\ G_n] + \ \ldots + \epsilon_{0,1}}$

HOW TO GAME HOP

Game$_{i+1}$

Game$_i$

Stratégie : Limiter l'adversaire :

- Idéaliser les réponses aux certaines réquêtes
- Limiter les réquêtes de l'adversaire
- Enlever des évennements "failure" causés par A

Stratégie : Renforcer le challenger

- Enlever des évennements "failure" causes par les algorithems honnêts
- Permettre au challenger à gagner une information sur les entrées/sorties de l'adversaire (on rappele : l'adversaire fonctionne "à boîte noire")

Game$_{i+1}$

PREUVE DE HMAC

PREUVE HMAC, PREMIÈRE ÉTAPE

- Jeu initial G_0:
  - L'adversaire a accès à la function de hachage (librement), et à un oracle oTag

$$k \leftarrow \mathrm{KGen}(1^\lambda)$$
$$(m, t) \leftarrow A^{\mathrm{oTag}(\cdot)}(\lambda)$$

A gagne ssi. :

$\mathrm{Vf}(k; m, t) = 1$

et m non-envoyé à oTag

- Le schéma de HMAC dans le modèle de l'oracle aléatoire :

$$r_1 \leftarrow RO\big((\hat{k} \oplus opad)\big|m\big)$$
$$\mathrm{HMAC}(k; m) = RO\big(\hat{k} \oplus ipad \,\big|\, r_1\big)$$

- Premier hop G_1:
  - L'attaquant a accès à oTag qui appelle le RO pour calculer le HMAC
  - L'attaquant a un accès independent par oracle à RO

Avertissement :

La transition Hash → RO n'est pas un jeu standard (mais on l'inclut ici pour la clarté !

DIFFÉRENCE ENTRE RO ET H

- En G_0 on utilise H, en G_1 on utilise RO

$$\boxed{\begin{array}{c} r_1 \leftarrow H\big((\hat{k} \oplus opad)\big|m\big) \\ \mathrm{HMAC}(k;m) = H\big(\hat{k} \oplus ipad \mid r_1\big) \end{array}}$$

$$\boxed{\begin{array}{c} r_1 \leftarrow RO\big((\hat{k} \oplus opad)\big|m\big) \\ \mathrm{HMAC}(k;m) = RO\big(\hat{k} \oplus ipad \mid r_1\big) \end{array}}$$

- Quelle est la différence ?
  - En absolu, aucune fonction de hachage n'est pas un RO
  - … mais dans nos jeux, cette différence se mesure par rapport à la probabilité de l'attaquant de gagner
  - C'est à dire, la différence est la capacité de l'adversaire à se rendre compte qu'on a passé de H à RO
  - On dénote "l'avantage de l'adversaire" par ϵ^{RO}
  - Pour une "bonne" fonction de hachage on va supposer que $\epsilon^{RO} \to 0$ pour une taille de paramètre convenable

PREUVE HMAC, ÉTAPE 2

- G_0: jeu initial, l'attaquant a accès à oTag et H
- G_1: l'attaquant a accès à oTag et RO
- G_2 : l'adversaire n'a plus le droit de faire une requête RO $\left(\left(\hat{k} \oplus opad\right) \middle| \, m\right)$

$$r_1 \leftarrow RO\left((\hat{k} \oplus opad) \middle| m\right)$$
$$\mathrm{HMAC}(k; m) = RO\left(\hat{k} \oplus ipad \,\middle|\, r_1\right)$$

$k \leftarrow \mathrm{KGen}(1^\lambda)$ $(m, t) \leftarrow A^{\mathrm{oTag}(\cdot)}(\lambda)$
A gagne ssi. : $\mathrm{Vf}(k; m, t) = 1$ et m non-envoyé à oTag

- **Proposition** : $\Pr[A\ gagne\ G_1] \leq \Pr[A\ gagne\ G_2] + \frac{1}{2^{|\hat{k}|}}$

- **Preuve** : Pour connaître l'entrée, l'attaquant doit connaître au minimum $\left(\hat{k} \oplus opad\right)$. Comme le RO ne fuit aucune information sur les entrées à partir des sorties, impossible qu'il apprenne cette valeur et doit la deviner.

  Si KGen est bien, l'attaquant a une proba de $\frac{1}{2^{|\hat{k}|}}$ de deviner la clé.

  La "distance" entre les deux jeux est :

$$|\Pr[A\ gagne\ G_1] - \Pr[A\ gagne\ G_2]| \leq \frac{1}{2^{|\hat{k}|}}$$

PREUVE HMAC, ÉTAPE 3

$$r_1 \leftarrow RO\big((\hat{k} \oplus opad)\big|m\big)$$
$$\mathrm{HMAC}(k;m) = RO\big(\hat{k} \oplus ipad \,\big|\, r_1\big)$$

- G_0: jeu initial, l'attaquant a accès à oTag et H
- G_1: l'attaquant a accès à oTag et RO
- G_2 : l'adversaire n'a plus le droit de faire une requête RO $\big((\hat{k} \oplus opad)\big|\, m)$
- G_3 : pas le droit de faire une requête $RO\big(\hat{k} \oplus ipad\, \big|RO((\hat{k} \oplus opad)\big|\, m)\big)$

$k \leftarrow \mathrm{KGen}(1^\lambda)$
$(m,t) \leftarrow A^{\mathrm{oTag}(\cdot)}(\lambda)$

A gagne ssi. :
$\mathrm{Vf}(k;m,t) = 1$
et m non-envoyé à oTag

- **Proposition** : $\Pr[A\ gagne\ G_2] \leq \Pr[A\ gagne\ G_3] + \frac{1}{2^{|\hat{k}|+\ell}}$
- **Preuve** : Suite au jeu G_2, l'attaquant n'a aucune information sur la deuxième partie de l'entrée, donc il doit deviner les deux parties de l'entrée. L'attaquant a une proba de $\frac{1}{2^{|\hat{k}|+\ell}}$ de deviner les entrées.

  La distance entre G_2, G_3 est : $|\Pr[A\ gagne\ G_2] - \Pr[A\ gagne\ G_3]| \leq \frac{1}{2^{|\hat{k}|+\ell}}$

PREUVE HMAC, ANALYSE G3

$$r_1 \leftarrow RO\big((\hat{k} \oplus opad)|m\big)$$
$$\mathrm{HMAC}(k;m) = RO\big(\hat{k} \oplus ipad \mid r_1\big)$$

- G_0: jeu initial, l'attaquant a accès à oTag et H
- G_1: l'attaquant a accès à oTag et RO
- G_2 : l'adversaire n'a plus le droit de faire une requête RO $\big((\hat{k} \oplus opad\big)|\, m)$
- G_3 : pas le droit de faire une requête $RO\big(\hat{k} \oplus ipad \,|RO((\hat{k} \oplus opad\big)|\, m)\big)$

$k \leftarrow \mathrm{KGen}(1^\lambda)$ $(m,t) \leftarrow A^{\mathrm{oTag}(\cdot)}(\lambda)$
A gagne ssi. : $\mathrm{Vf}(k;m,t) = 1$ et m non-envoyé à oTag

- **Proposition** : $\Pr[A\ gagne\ G_3] = \frac{1}{2^\ell}$
- **Preuve** : Dans le jeu G_3, l'attaquant ne peut pas utiliser son oracle aléatoire pour calculer la valeur t, qui correspond au tag juste pour le message m. De plus, comme nous utilisons un RO, toute autre requête à oTag ou RO n'aide pas à trouver des infos sur le bon tag.

  L'attaquant a une proba de $\frac{1}{2^\ell}$ de deviner t.

PREUVE HMAC, FIN

- G_0: jeu initial, l'attaquant a accès à oTag et H
- G_1: l'attaquant a accès à oTag et RO

$$\Pr[A\ gagne\ G_0] \leq \Pr[A\ gagne\ G_1] + \epsilon^{RO}$$

- G_2 : l'adversaire n'a plus le droit de faire une requête RO ($\left(\hat{k} \oplus opad\right)| \, m$)

$$\Pr[A\ gagne\ G_1] \leq \Pr[A\ gagne\ G_2] + \frac{1}{2^{|\hat{k}|}}$$

- G_3 : pas le droit de faire une requête $RO\left(\hat{k} \oplus ipad\ \middle| RO(\left(\hat{k} \oplus opad\right)| \, m)\right)$

$$\Pr[A\ gagne\ G_2] \leq \Pr[A\ gagne\ G_3] + \frac{1}{2^{|\hat{k}|+\ell}}$$

$$\Pr[A\ gagne\ G_3] = \frac{1}{2^{\ell}}$$

- Donc, en mettant les éléments ensemble :

$$\Pr[A\ gagne\ G_0] \leq \frac{1}{2^{|\hat{k}|}} + \frac{1}{2^{|\hat{k}|+\ell}} + \frac{1}{2^{\ell}} + \epsilon^{RO}$$

$k \leftarrow \mathrm{KGen}(1^{\lambda})$ $(m,t) \leftarrow A^{\mathrm{oTag}(\cdot)}(\lambda)$
A gagne ssi. : $\mathrm{Vf}(k; m, t) = 1$ et m non-envoyé à oTag

DIFFÉRENCE ENTRE RO ET H

- En G_0 on utilise H, en G_1 on utilise RO

$$\boxed{\begin{array}{c} r_1 \leftarrow H\big((\hat{k} \oplus opad)\big|m\big) \\ \mathrm{HMAC}(k;m) = H\big(\hat{k} \oplus ipad \,\big|\, r_1\big) \end{array}} \qquad \boxed{\begin{array}{c} r_1 \leftarrow RO\big((\hat{k} \oplus opad)\big|m\big) \\ \mathrm{HMAC}(k;m) = RO\big(\hat{k} \oplus ipad \,\big|\, r_1\big) \end{array}}$$

- Quelle est la différence ?
  - En absolu, au… …'est pas un RO
  - Mais dans ???????? …ure par rapport à la probabilité de l'attaquant de gagner
  - C'est à dire, la… …e de l'adversaire à se rendre compte qu'on a passé de H à RO
  - On dénote "l'avantage de l'adversaire" par ϵ^{RO}
  - Pour une "bonne" fonction de hachage on va supposer que $\epsilon^{RO} \to 0$ pour une taille de paramètre convenable

INTERMEZZO : AVANTAGE

LES JEUX DE INDISTINGUABILITÉ

- Fonctions de hachage : l'attaquant doit trouver une valeur longue (préimage, collision...)
- Si la taille de paramètre est large, alors la probabilité de déviner approche 0
- Un autre type de jeu est la famille de jeux d'indistinguabilité
  - Retrouvable dans la sécurité des schémas de chiffrement, les générateurs pseudo-alétoires, les fonctions PRF...

- Les jeux d'indistinguabilité dépendent typiquement d'un bit b
  - Ce bit est un artifice du jeu, qui nous permet d'exprimer la propriété de sécurité désirée
- Le but de l'adversaire est de pouvoir deviner la valeur de b à la suite de son attaque

EXEMPLE : HASH VS RO

- Voici le jeu suivant :
  - La valeur x représente une partie "imprévisible" de l'entrée
  - Intuition pour ce jeu : l'attaquant n'arrive pas à distinguer la vraie function de hachage d'un oracle aléatoire

- Analyse :
  - La taille de X joue un rôle essentiel (x doit rester imprévisible)
  - La fonction de hachage doit être "bonne"

- La probabilité $\Pr[A \text{ gagne}]$:
  - Même pour un x de grande taille…
  - Même pour une bonne function de hachage…
  - … La probabilité est d'au moins ½ !

$k \xleftarrow{\$} K$
$x \xleftarrow{\$} X$
$d \leftarrow A^{\mathrm{oHash}_b(\cdot)}(k)$

A gagne ssi. : $d = b$

$\underline{\mathrm{oHash}_0(m)}$

$h \leftarrow H_k(x|m)$
return h

$\underline{\mathrm{oHash}_1(m)}$

Nouveau m :
$h \xleftarrow{\$} \{0,1\}^\ell$
m vu avant :
reprise de h
return h

L'AVANTAGE DE A

- Pour les jeux d'indistinguabilité, la probabilité de gagner n'est pas une bonne mesure

- C'est pourquoi d'habitude nous allons mesurer le succès d'un adversaire par son *avatange* :
  - Notamment son avantage par rapport à une attaque par force brute

$$Adv[A] = \Pr[A\ gagne] - \Pr[\text{brute force pour } \lambda \text{ large}]$$

- Jeux d'indistinguabilité : $Adv[A] = \Pr[A\ gagne] - 1/2$
- Jeux où A doit produire une valeur : $Adv[A] = \Pr[A\ gagne]$

APPLICATION À NOTRE PREUVE

- En G_0 on utilise H, en G_1 on utilise RO

$$\boxed{\begin{array}{c} r_1 \leftarrow H\left((\hat{k} \oplus opad)\middle|m\right) \\ \mathrm{HMAC}(k;m) = H\left(\hat{k} \oplus ipad \mid r_1\right) \end{array}} \qquad \boxed{\begin{array}{c} r_1 \leftarrow RO\left((\hat{k} \oplus opad)\middle|m\right) \\ \mathrm{HMAC}(k;m) = RO\left(\hat{k} \oplus ipad \mid r_1\right) \end{array}}$$

- Ceci veut dire que G_0 correspond à l'utilisation de oHash_0 et G_1 à l'utilisation de oHash_1
- L'adversaire distingue entre les deux jeux avec l'avantage de son jeu d'indistinguabilité

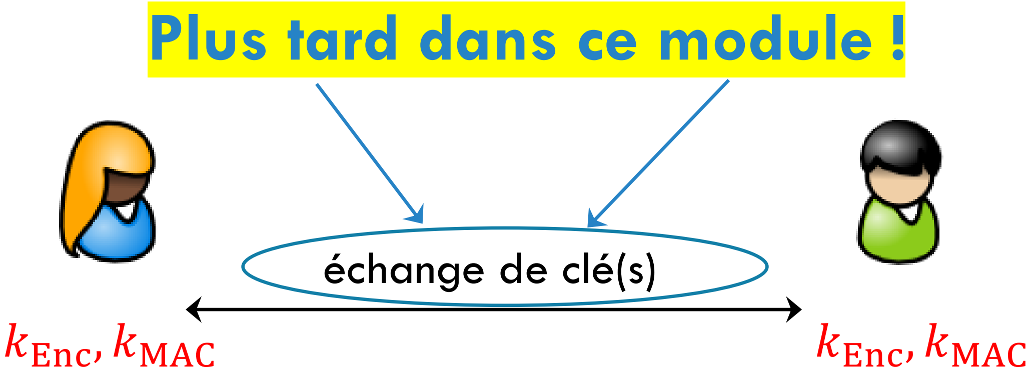
CHIFFREMENT ET MAC

CANAUX SÉCURISÉS

- Un des objectifs les plus importants dans la cryptographie est la communication sécurisée
- Communication sécurisée = 3 propriétés principales :
  - Confidentialité : l'attaquant n'a aucune information sur le contenu des messages échangés
  - Authenticité : l'attaquant ne peut pas insérer des messages "frais" dans la conversation
  - Intégrité : toute modification d'un message par un attaquant est détectée

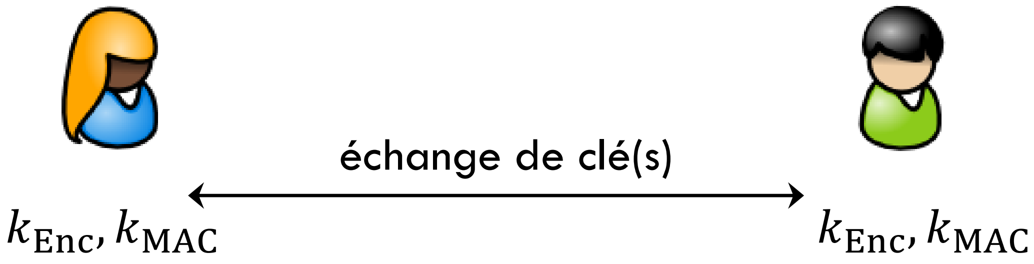
- En pratique un canal sécurisé implique du chiffrement et des schémas d'authentification
  - Principalement à clé symétrique (Pourquoi ???)
  - On peut faire le chiffrement et le MAC séparément ou ensemble (AEAD)

ETABLISSEMENT D'UN CANAL SÉCURISÉ



- MAC then Encrypt :
  - $t \leftarrow \text{MAC}_{k_{\text{MAC}}}(m)$ et $c \leftarrow \text{Enc}_{k_{\text{Enc}}}(m, t)$ et ensuite envoyer c
- MAC and Encrypt :
  - $t \leftarrow \text{MAC}_{k_{\text{MAC}}}(m)$ et $c \leftarrow \text{Enc}_{k_{\text{Enc}}}(m)$ et ensuite envoyer c, t
- Encrypt then MAC :
  - $c \leftarrow \text{Enc}_{k_{\text{Enc}}}(m)$et $t \leftarrow \text{MAC}_{k_{\text{MAC}}}(c)$ et ensuite envoyer c, t

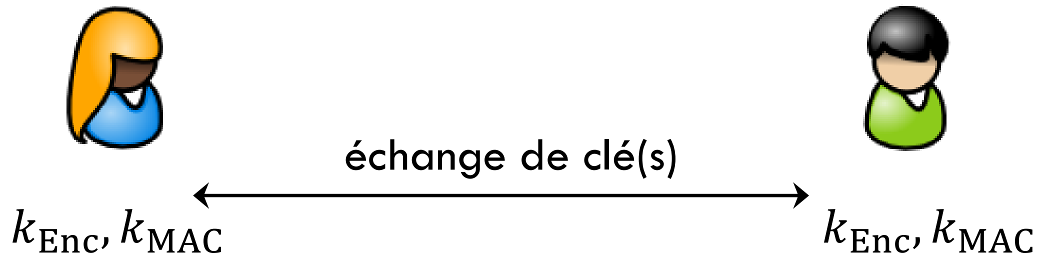
ETABLISSEMENT D'UN CANAL SÉCURISÉ



- MAC then Encrypt :
  - $t \leftarrow \text{MAC}_{k_{\text{MAC}}}(m)$ et $c \leftarrow \text{Enc}_{k_{\text{Enc}}}(m, t)$ et ensuite envoyer c

- De l'autre côte il faut déchiffrer c avant de s'assurer que t est correct
  - Ceci peut mettre le réceveur de messages en danger

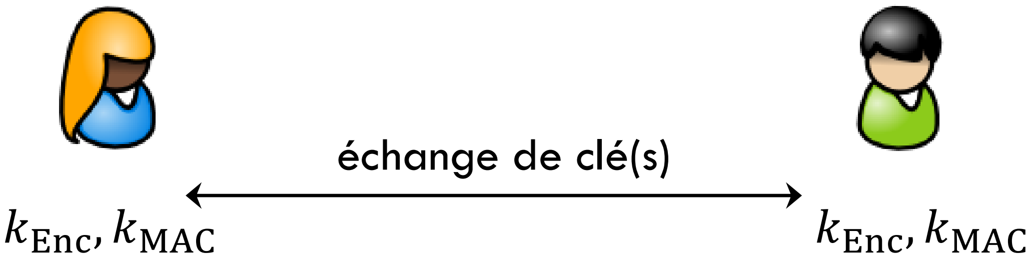
ETABLISSEMENT D'UN CANAL SÉCURISÉ



- MAC and Encrypt :
  - $t \leftarrow \mathrm{MAC}_{k_{\mathrm{MAC}}}(m)$ et $c \leftarrow \mathrm{Enc}_{k_{\mathrm{Enc}}}(m)$ et ensuite envoyer c, t

- De l'autre côte : pour verifier t il faut premièrement déchiffrer c
  - Encore une fois, ceci peut mettre le réceveur de messages en danger

ETABLISSEMENT D'UN CANAL SÉCURISÉ



- Encrypt then MAC :
  - $c \leftarrow \mathrm{Enc}_{k_{\mathrm{Enc}}}(m)$ et $t \leftarrow \mathrm{MAC}_{k_{\mathrm{MAC}}}(c)$ et ensuite envoyer c, t

- Premièrement on vérifie la provenance de c avant de le déchiffrer
  - Plus de sécurité pour les deux parties

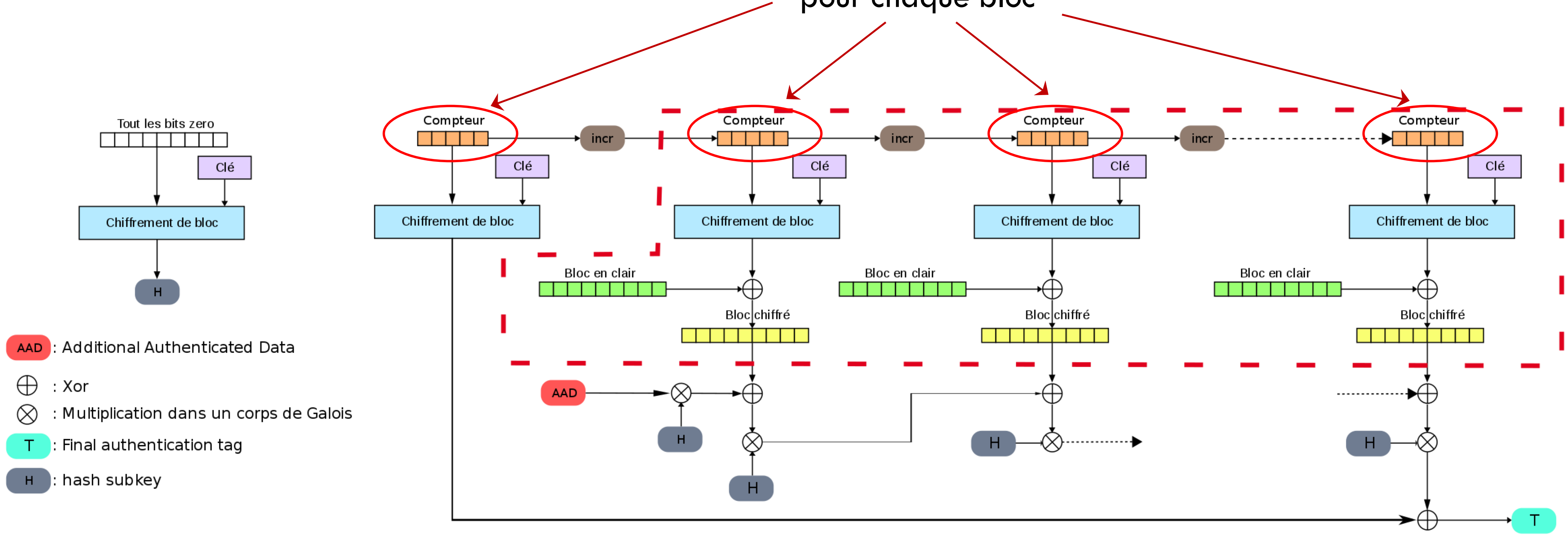
AEAD

- Authenticated Encryption with Additional Data :
  - Une partie du message est chiffrée et authentifiée
  - Une autre partie est authentifiée mais envoyée en clair (AD)

- Une seule clé est utilisée pour les deux opérations
- Deux méthodes très populaires instanciant AEAD :
  - AES-GCM
  - ChaCha avec Poly1305

AES-GCM (GALOIS COUNTER MODE)

compteur : valeur consécutive
pour chaque bloc



AES-GCM (GALOIS COUNTER MODE)