Securing multimedia content delivery

Secure distribution

gaetan.leguelvouit@b-com.com

b com

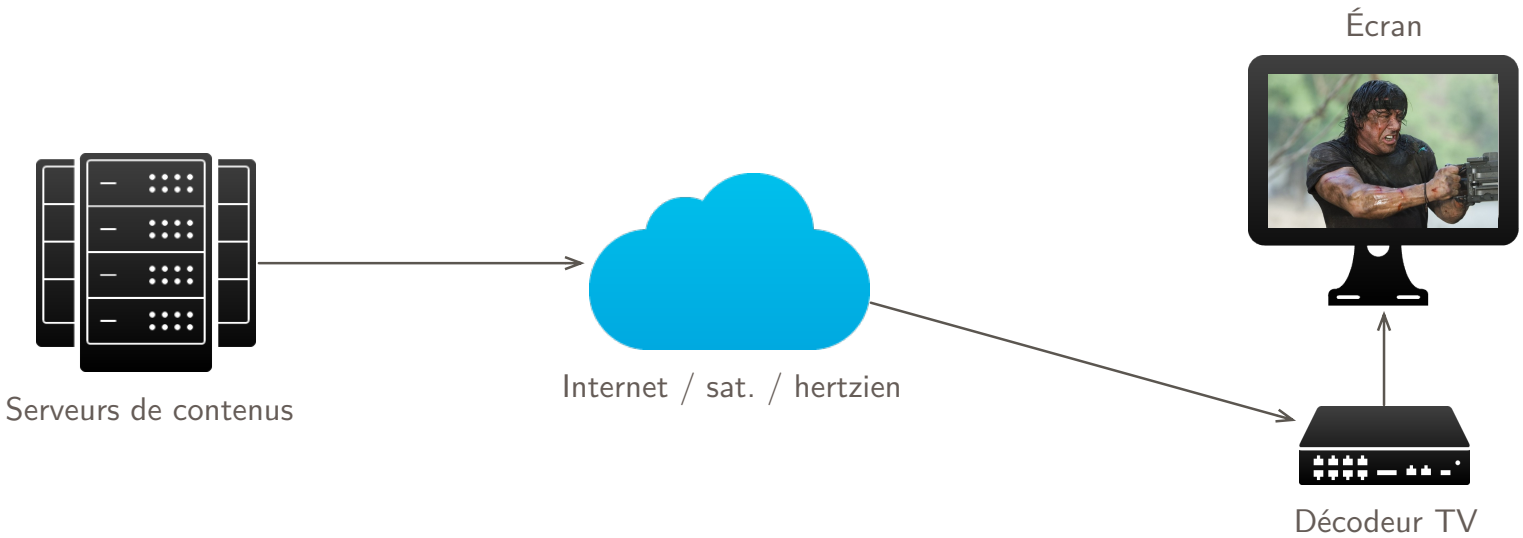
Systèmes d'accès conditionnel (CAS)

Traçage de traitre

Mises en œuvre

Chaine de distribution

- Serveur de contenus : fichiers multimédia
- Canal de transmission : broadcast ou unicast
- Terminal (décodeur, STB) : transforme le fichiers en une suite d'images et de son, matériel ou logiciel
- Écran : TV, moniteur, projecteur



Macrovision

Protection des sorties vidéo analogiques.

- Modification de la composante Y pour leurrer le circuit de control de gain
- Image délavée, décalée
- Utilisé sur les magnétoscopes (il y a longtemps)

HDCP

High-Bandwidth Digital Content Protection. Protection des sorties vidéo numériques : DVI, HDMI, DP.

- Chiffrement du flux vidéo : stream cipher symétrique 56 bits
- Authentification des terminaux et échange de clefs : un ensemble de clefs privées et une clef publique par modèle de terminal
- Révocation : liste des clefs publiques révoquées dans les contenus

Clef maitre perdue dans la nature en 2010 : tout le monde peut se faire passer pour un terminal valide.

Digital rights management

Le contenu est chiffré avec une clef symétrique $C_{k_c}(c)$ et contient un ID et une URL de *callback*. Le terminal possède un ID et une clef k_t uniques.

1. Le terminal communique via l'URL son ID et l'ID du contenu qu'il veut lire
2. Le serveur vérifie l'abonnement ou le paiement
3. La clef k_c est envoyée chiffrée par k_t au terminal, avec un ensemble de droits
4. Le terminal vérifie les droits, déchiffre la clef puis le contenu

Cas broadcast

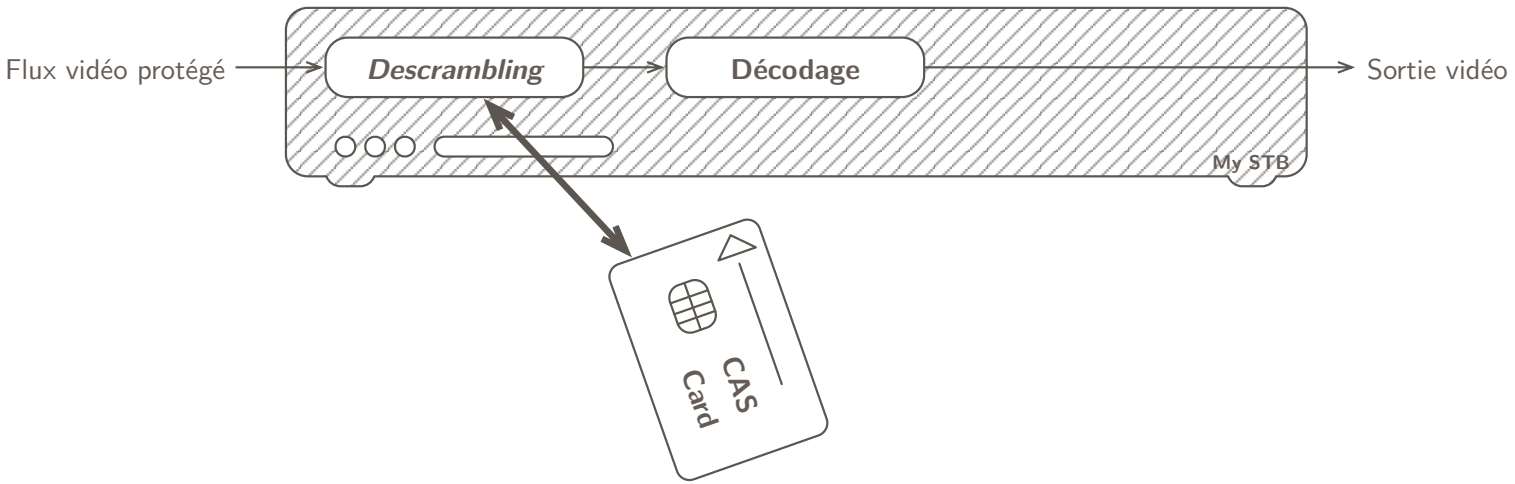
Pas de voie montante ! Le contenu est chiffré avec une suite de clefs symétriques $C_{k_i}(c)$ (*control words*). Le terminal possède une clef k_t unique.

1. Le serveur connait l'ensemble des clefs de terminaux des abonnés
2. Le contenu chiffré est diffusé
3. En parallèle, un flux de clefs chiffrées $C_{k_m}(k_i)$ est diffusé
4. Régulièrement, un flux composé de $C_{k_t}(k_m)$ est diffusé
5. Le terminal peut donc déchiffrer la clef maitre
6. Puis ensuite déchiffrer les CW et le contenu

Décodeur

La fonction de déchiffrement (et la clef) est implémentée sur une **carte à puce**. Pendant longtemps, l'attaque la plus populaire était d'intercepter les CW en sortie de carte :

- redistribution parallèle de flux de CW
- à destination de terminaux adaptés
- avec rémunération sous forme d'abonnement



Affichage d'un fingerprint

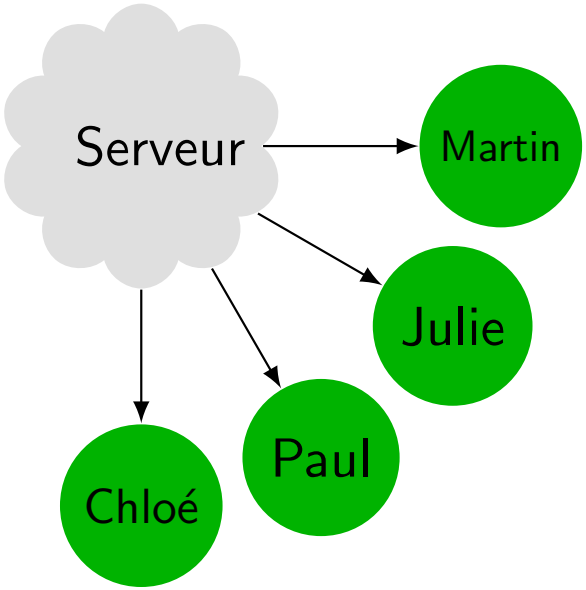
- ↑ débit montant = streaming direct du flux
- mais on peut activer à distance l'affichage de l'ID terminal
- et donc désactiver les terminaux compromis

→ 1ère attaque par collusion !

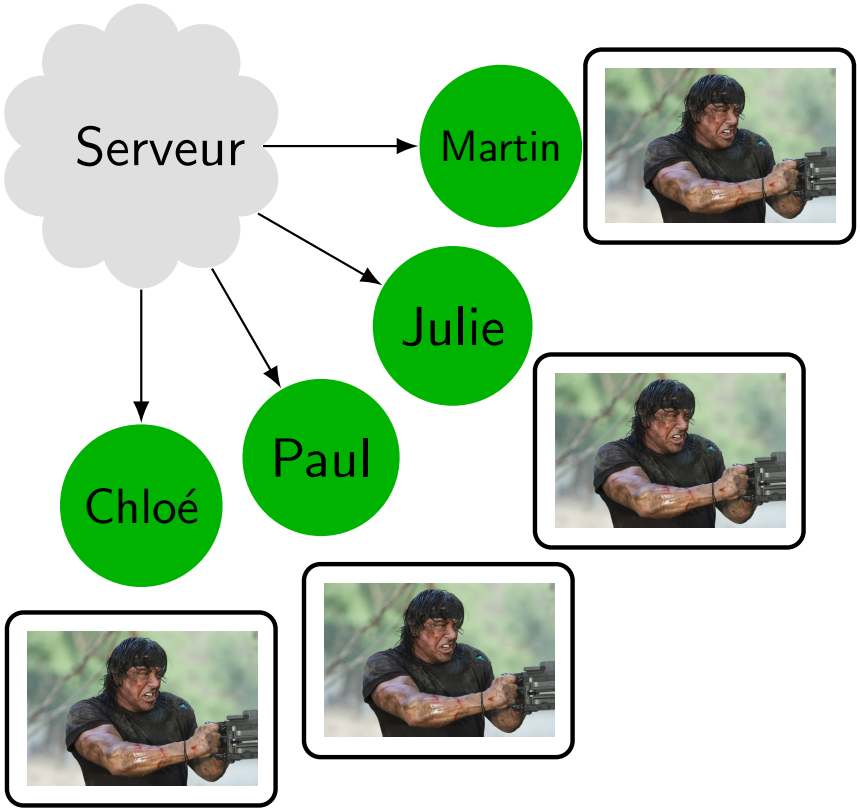
Quelques faits

- Les valeurs de tables de logarithmes (début $20^{\text{ème}}$ siècle)
- Les fuites de documents du cabinet de Mrs. Tatcher (1980)
- Redistribution d'un film destiné aux membres de l'académie des Oscars (2004 et 2015)
- Protection AACS des disques Blu-Ray

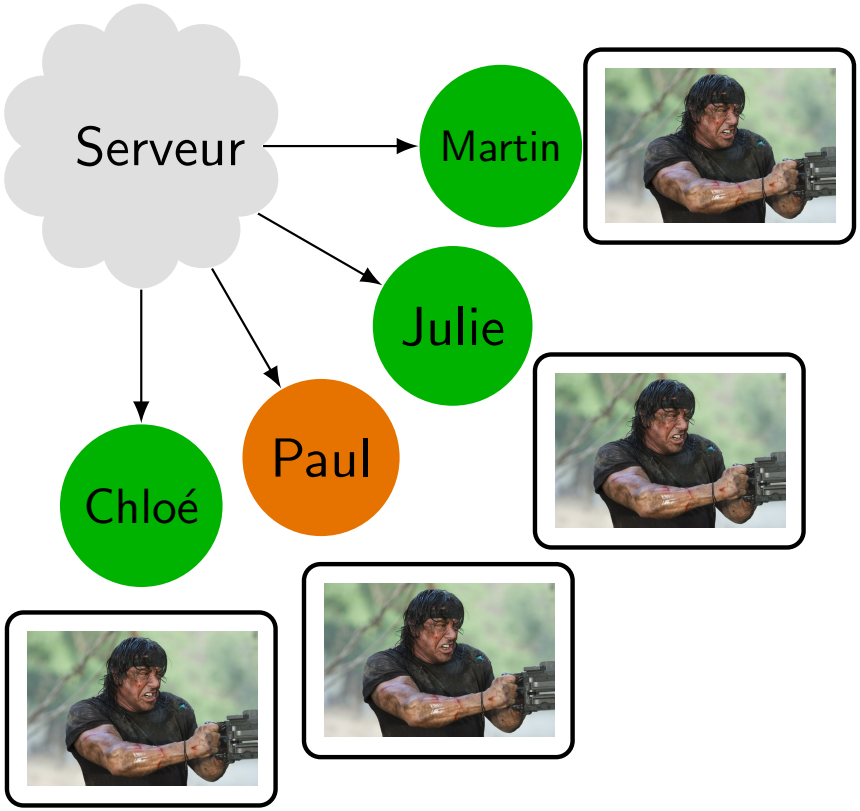
Comment empêcher la redistribution illégale ?



Comment empêcher la redistribution illégale ?

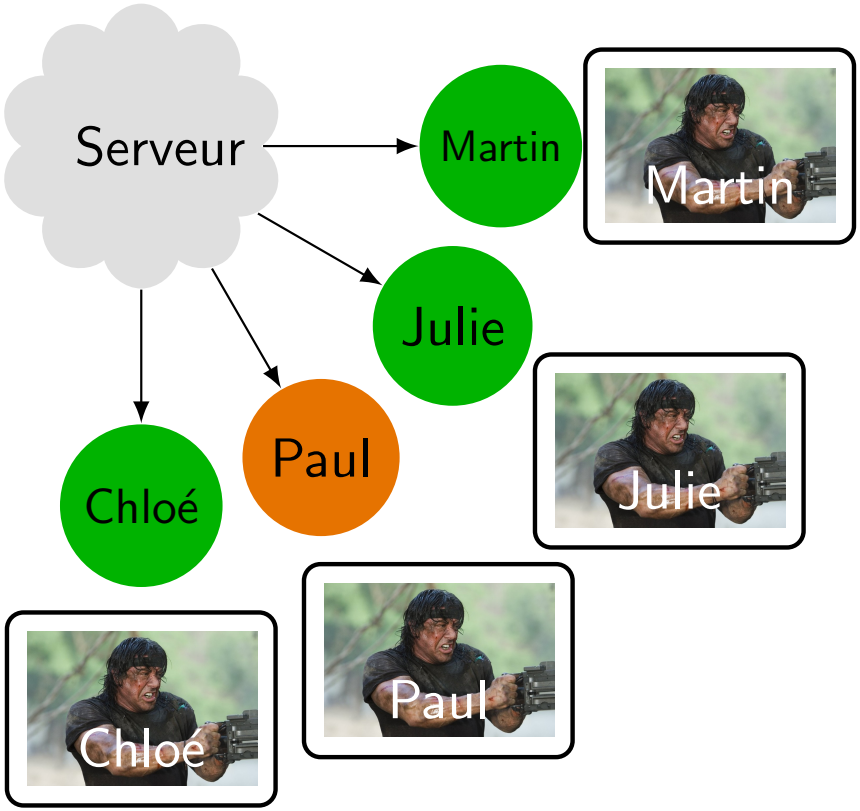


Comment empêcher la redistribution illégale ?



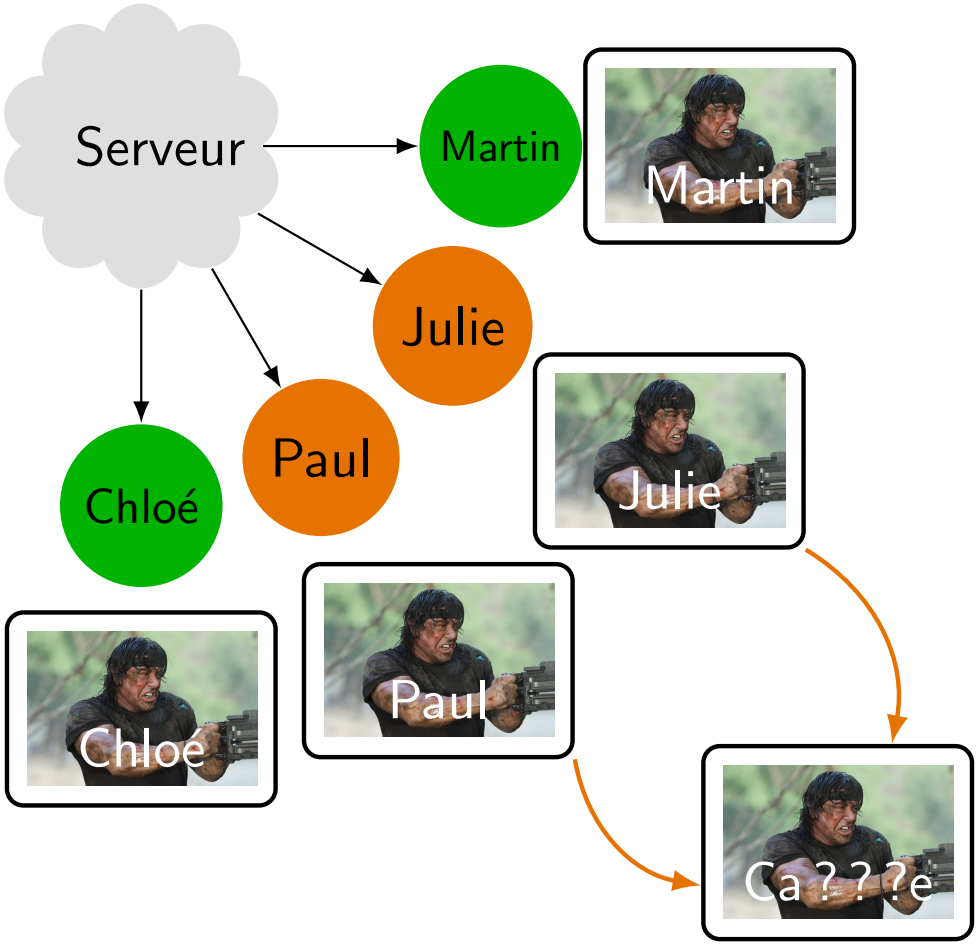
- La cryptographie ne suffit pas

Comment empêcher la redistribution illégale ?



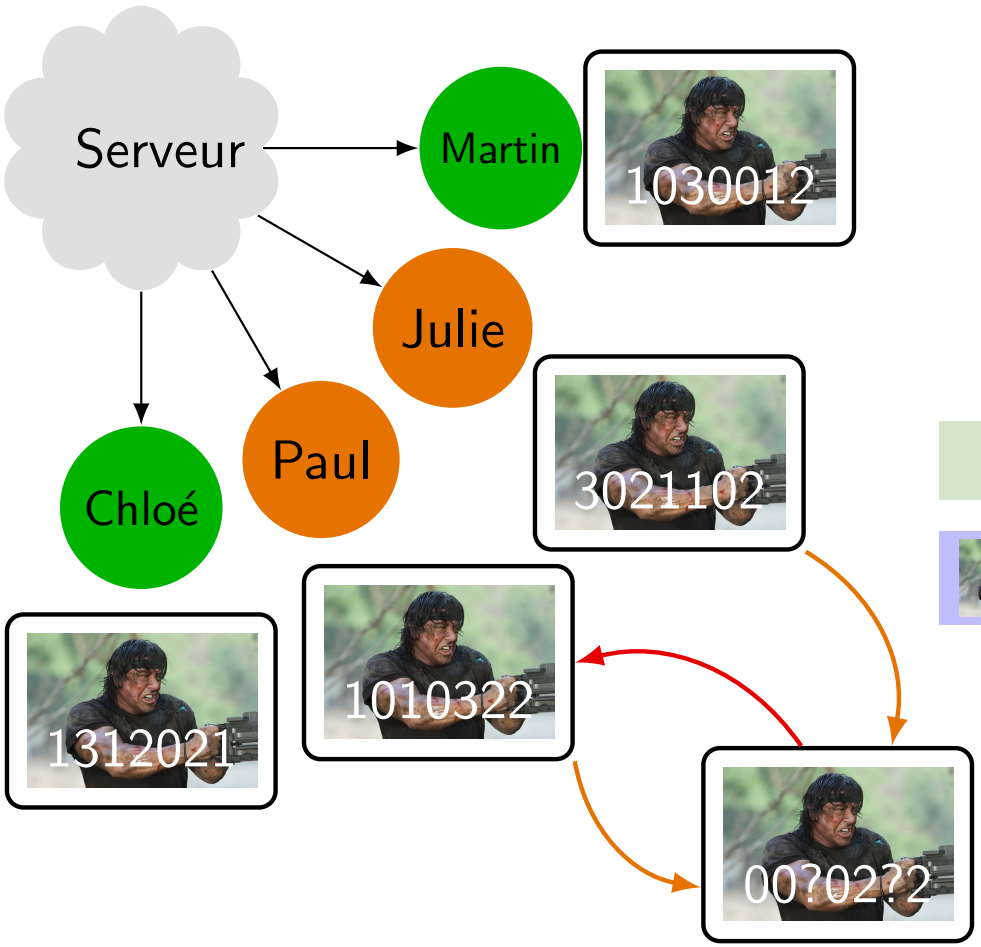
- La cryptographie ne suffit pas
- Besoin de tatouage

Comment empêcher la redistribution illégale ?

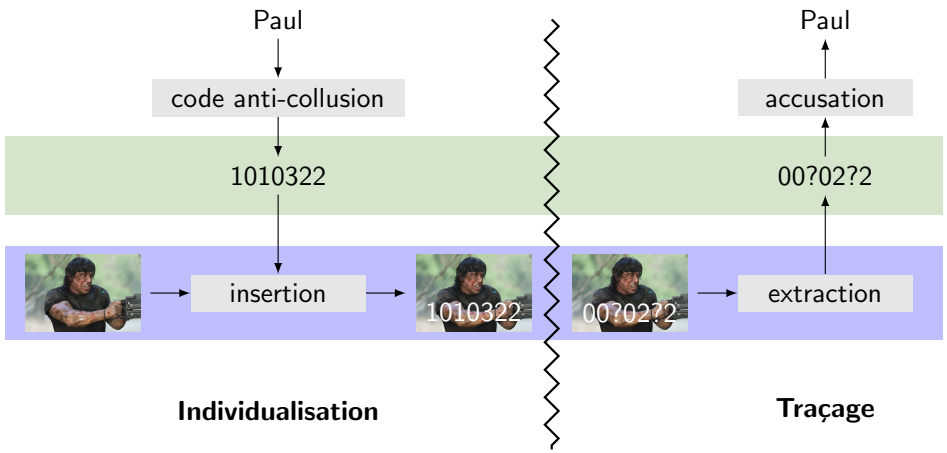


- La cryptographie ne suffit pas
- Besoin de tatouage

Comment empêcher la redistribution illégale ?



- La cryptographie ne suffit pas
- Besoin de tatouage
- Besoin de code anti-collusion avec du traçage



Généralités

Chaque copie distribuée est individualisée

- par le serveur (unicast, iTunes, VOD)
- par le terminal chez le client (multicast, Bluray)

Chaque copie contient un message identifiant $\mathbf{x}_j$ (un "fingerprint" de l'utilisateur j), inséré sous la forme de tatouage.

Mêmes techniques d'insertion que pour le tatouage robuste
$\Rightarrow$ mêmes attaques et mêmes exigences de sécurité.

Mais nous devons y ajouter les attaques par collusion et des contraintes de sécurité supplémentaires.

Quelques questions préliminaires

Collusion = plusieurs utilisateurs malhonnêtes se regroupent et utilisent leurs copies pour forger une copie pirate.

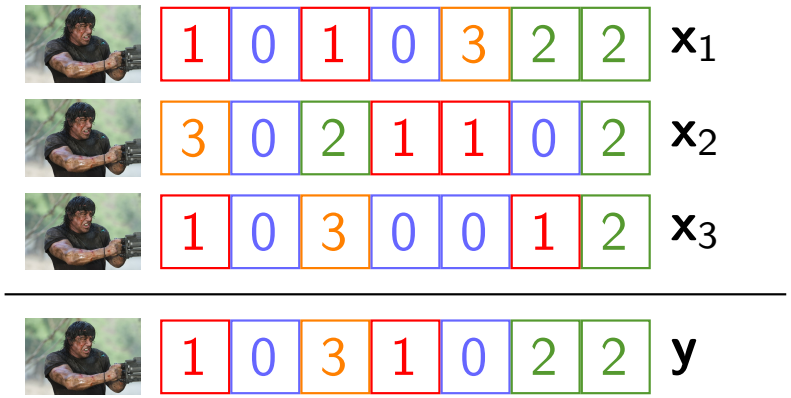
- Combien de colluders ?
- Quelles actions ?
- Un innocent peut-il être accusé ?
- Un participant à la collusion peut-il échapper à la détection ?

1ère stratégie : échange de blocs (copy/paste)

Le bloc forgé est un des blocs originaux.

Exemples :

- choisir au hasard avec une loi uniforme,
- à la majorité, minorité, etc.



D'un point de vue code, c'est comme si les colluders échangeaient les symboles de leurs mots de code.

La protection est assurée par le code anti-collusion.

2nde stratégie : fusion de blocs

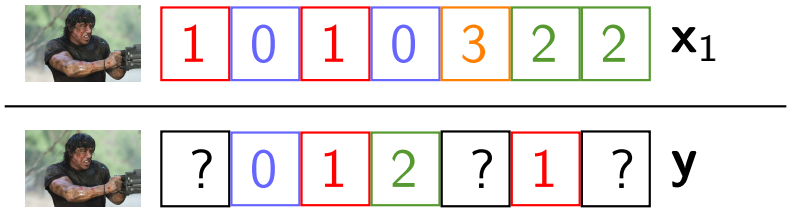
Le bloc forgé est issu de la fusion des blocs des colluders.
Exemples :

- moyenne, min, max, median,
- découpages plus ou moin fins,
- pixels, coefficients DCT/ondelettes

On constate des erreurs et des effacements. La protection repose sur la robustesse du tatouage, puis sur le code anti-collusion.

3ème stratégie : dégradation

- Processing : compression, débruitage, ...
- Un pirate suffit
- Génère des erreurs et des effacement.



La protection repose sur la robustesse du tatouage, puis sur le code anti-collusion.

Attaques et hypothèses : résumé



1 0 1 0 3 2 2 $\mathbf{x}_1$

3 0 2 1 1 0 2 $\mathbf{x}_{...}$

1 0 3 0 0 1 2 $\mathbf{x}_c$

Collusion (c utilisateurs parmi n)

1 0 3 1 0 2 2 $\mathbf{y}$

Copy/paste des blocs (*e.g.* hasard, maj., etc)

0 0 ? 0 2 ? 2 $\mathbf{y}$

Fusion de blocs (*e.g.* moyenne)

? 0 1 2 ? 1 ? $\mathbf{y}$

Traitements individuels (*e.g.* compression)

Hypothèses et notations : résumé

- Document découpé en blocs, 1 symbole caché par bloc. Principe de Kerkhoffs : le découpage est public.

- Il y a n utilisateurs.
- Il y a donc n codewords $\mathbf{x}_j$, $1 \leq j \leq n$, de longueur m. $\mathbf{x}_j = (\mathbf{x}_j[1], \ldots, \mathbf{x}_j[m])$ identifie l'utilisateur j.
- Une collusion implique au plus c utilisateurs.
- La construction d'une copie pirate $\mathbf{y}$ se fait bloc à bloc : le $i^{\text{ème}}$ bloc forgé se base sur les $i^{\text{ème}}$ blocs des colluders.

Modélisations

- La Marking Assumption : Boneh & Shaw 1995-1998 : $\text{desc}(\mathcal{C}) = \{Y | Y_i \in \{X_{j_1 i}, \ldots, X_{j_c i}\}\}$. Rares extensions vers tout symbole de l'alphabet (Barg 2003), en autorisant des effacements (Boneh & Shaw 1998, Guth & Pfitzmann (1999), Barg 2003). Echange de blocs (beaucoup étudié), fusion.

- Le modèle de Somekh-Baruch et Merhav (2005), ou modèle de la distorsion. On contraint la distorsion lors de l'insertion, et lors de l'attaque. Très peu étudié.

- Le modèle signal, issu d'une vision "signal" (2001-2003) : signaux modélisés par des gaussiennes, attaques au niveau du signal lui-même. Fusion, lessivage. Considéré dès qu'il y a une insertion.

Approches possibles

1. Le code anti-collusion code comme une couche supplémentaire :
   - L'approche "crypto/codes" (1998-) basée sur la Marking Assumption ;
   - L'approche statistique initiée par Tardos (2003-), uniquement basée sur la Marking Assumption ;
   - L'approche "crypto/codes" (2005-2006) basée sur les modèles de Somekh-Baruch et Merhav.
2. L'approche signal.
3. Adopter une vision globale des attaques, et combiner le plus judicieusement possible un code anti-collusion et une technique d'insertion.

Marking Assumption

Traçabilité forte : $\mathbb{P}(\textbf{accuser un innocent}) = 0$

- ⊙ code correcteur
- ☹ $n \geq 3, c \geq 2$: uniquement les attaque copy/paste
- ☹ codes trop longs, sur des alphabets volumineux [HvLLT98]

Traçabilité faible : $\mathbb{P}(\textbf{accuser un innocent}) < \varepsilon$

- ⊙ codes correcteurs + codes probabilistes
- ✓ copy/paste + fusion
- ⊙ limite de Peikert [PSS03][Tar03,Tar08] : $m \geq \mathcal{O}(c^2 \ln(1/\varepsilon))$
- ✓ premiers codes à atteindre la limite : codes de Tardos

Codes de Tardos : la révolution

Tardos 2003 : une construction simplissime, avec

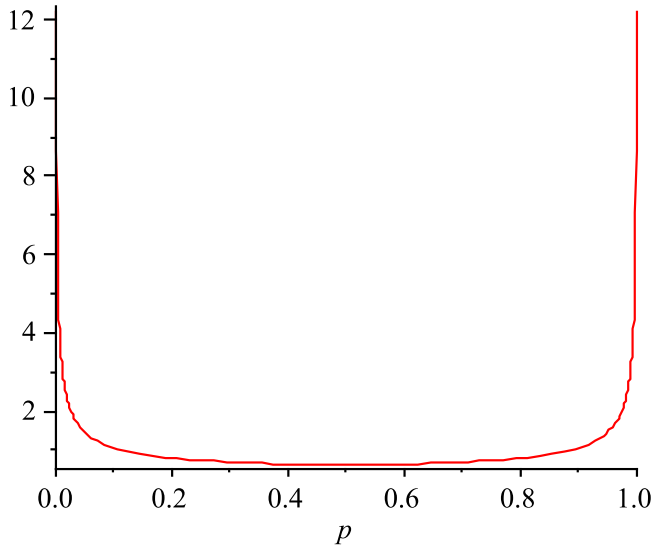
$$m = 100c^2 \left\lceil \ln\left(\frac{1}{\varepsilon_1}\right)\right\rceil \quad Z = 2\pi c\lceil \ln(1/\varepsilon_1)\rceil$$

$$\mathbb{P}(\text{accuser un innocent}) < \varepsilon_1 \quad (\varepsilon_1 = \varepsilon)$$
$$\mathbb{P}(\text{n'accuser personne}) < \varepsilon_2 \quad (\varepsilon_2 = \varepsilon^{c/4})$$

Skoric et al. 2007 : construction symétriques plus efficace pour $1 << c$, $\varepsilon_1 << \varepsilon_2$, $m = 2\pi^2 c^2 \lceil \ln(1/\varepsilon_1)\rceil$ dans le cas binaire + extension q-aire

Tardos version binaire symétrique [SKC08]+[SVCT08]

1. Initialisation
   - Générer un vecteur $\mathbf{p}$ de dimension m dont les éléments suivent la distribution f
   - Le vecteur $\mathbf{p}$ est la clef secrète du code
2. Construction
   - Pour chaque utilisateur, générer un vecteur binaire $\mathbf{x}_j$ aléatoire tel que $\mathbb{P}(\mathbf{x}_j[i] = 1) = \mathbf{p}[i]$
3. Accusation
   - Pour chaque utilisateur, comparer la séquence $\mathbf{y}$ avec son vecteur $\mathbf{x}_j$ et sommer ses scores en fonction de $\mathbf{p}$



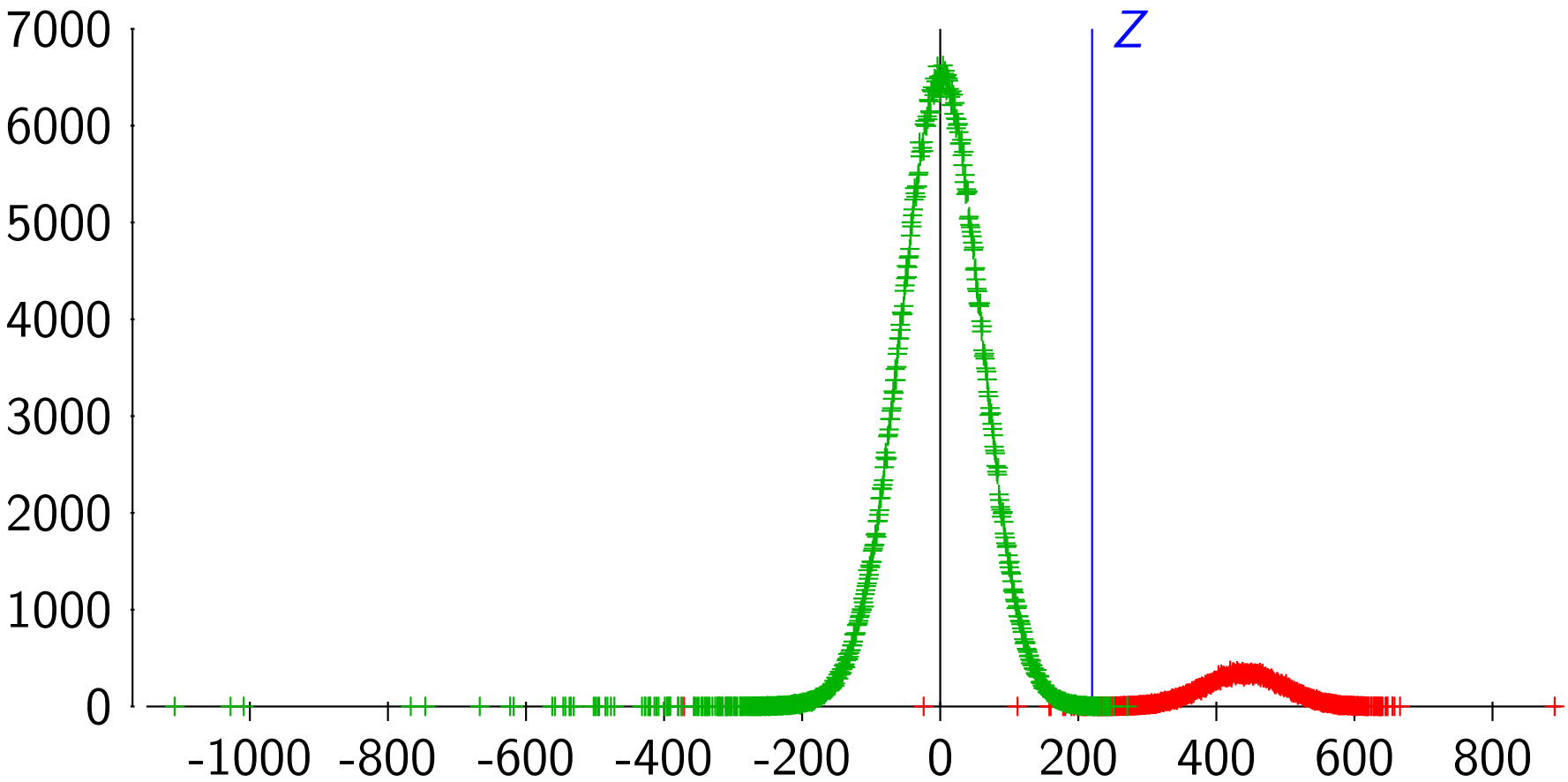
Fonction d'accusation

$$S_j = \sum_{i=1}^{m} g(\mathbf{y}[i], \mathbf{x}_j[i], \mathbf{p}[i]) >^? Z$$

$$g(1,1,p) = g(0,0,1-p) = \sqrt{(1-p)/p}$$

$$g(1,0,p) = g(0,1,1-p) = -\sqrt{p/(1-p)}$$

Innocents/accusés



Analyse rapide

La longueur "minimale" est indépendante du nombre d'utilisateurs (n peut augmenter après coup), facile à implémenter, gestion de c plus grands (> 2).

Accusation : faisceau de preuves (Cluedo), accusation utilisateur par utilisateur, indépendante de la stratégie, bonne tolérance aux effacements (sauf positions verrouillées par la Marking Assumption), qui "raccourcissent" le code.

Ordres de grandeur (1/2)

n	$1 << c$	$\varepsilon_1 (<< \varepsilon_2)$	$m = \lceil (2\pi^2 c^2 \lceil \ln(1/\varepsilon_1) \rceil) \rceil$	$Z = 2\pi c \lceil \ln(1/\varepsilon_1) \rceil$
10^2	5	10^{-3}	3455	219.91
10^3	5	10^{-4}	4935	314.16
	5	10^{-7}	8390	534.07
	7	10^{-4}	9673	439.82
	7	10^{-7}	16443	747.70
	10	10^{-4}	19740	628.32
	10	10^{-7}	33557	1068.14
	20	10^{-4}	78957	1256.64
10^4	5	10^{-5}	5922	376.99
	7	10^{-5}	11687	527.79
	10	10^{-5}	23688	753.98
	20	10^{-5}	94749	1507.96

Ordres de grandeur (2/2)

n	$1 << c$	$\varepsilon_1 (<< \varepsilon_2)$	$m = \lceil (2\pi^2 c^2 \lceil \ln(1/\varepsilon_1) \rceil) \rceil$	$Z = 2\pi c \lceil \ln(1/\varepsilon_1) \rceil$
10^4	5	10^{-5}	5922	376.99
	7	10^{-5}	11687	527.79
	10	10^{-5}	23688	753.98
	20	10^{-5}	94749	1507.96
10^5	5	10^{-6}	6909	439.82
	7	10^{-6}	13542	615.75
	10	10^{-6}	27635	879.65
	20	10^{-6}	110540	1759.29
10^6	5	10^{-7}	8390	534.07
	7	10^{-7}	16443	747.70
	10	10^{-7}	33557	1068.14
	20	10^{-7}	134227	2136.28

Bibliographie

Tardos 2003 : limité théorique sur m, preuve des limites d'erreur, pas de justification de la construction (choix pour f et g). Même efficacité quelles que soient les attaques.

Skoric et al. 2007 : longueur plus courte, c important, extension au cas q-aire.

Xie et al. 2008* : schèma complet avec tatouage, basé sur Tardos et Broken Arrows, extension de Tardos pour détecter plusieurs symboles.

Skoric et al. 2009+11 détection évoluée de plusieurs symboles.

Billet et al. 2008 : application au traçage de traitre classique (crypto).

Bibliographie

Furon et al. 2008 : estimation d'événements rares, application au calcul de ε_1.

Furon et al. 2008 : proof des choix de Tardos.

Charpentier et al. 2009* : estimation dynamique de l'attaque, et optimisation en conséquence de la fonction d'accusation.

Furon et al. 2009 : améliorations + pire attaque.

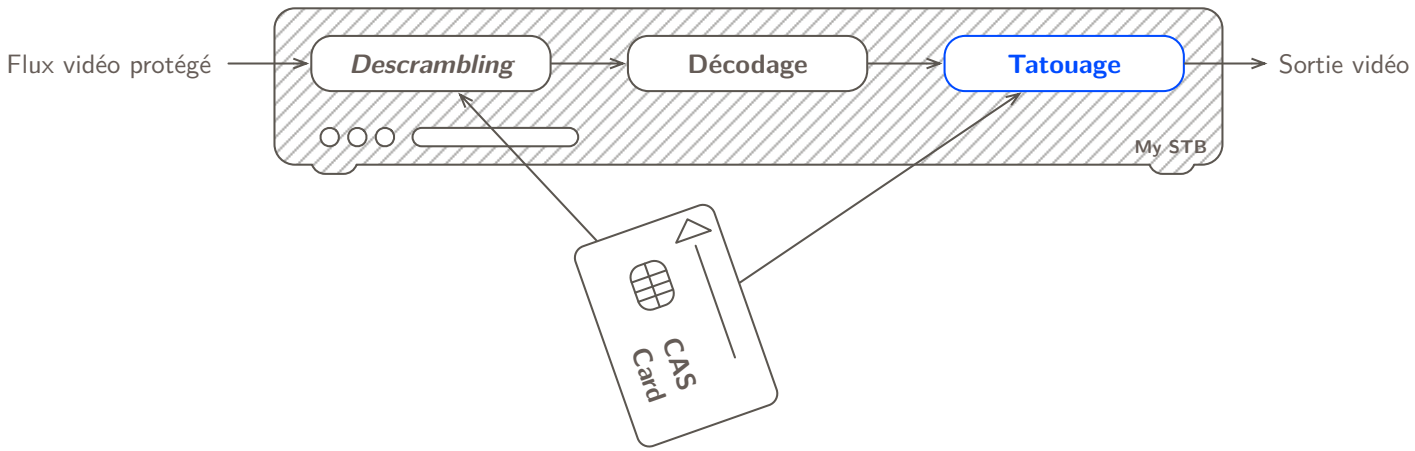
Nuida et al. 2007-09 : longueur réduite et implémentation plus efficace.

Mathon et al. 2010 : contraintes de robustesse et de sécurité pour la couche tatouage avec utilisation des codes de Tardos.

Tatouage côté client

Pour être compatible avec tous les formats vidéo, la fonction de tatouage est placé après le décodage.
Dans une set top box, elle utilise les informations uniques de la carte à puce comme identifiant.
Contraintes :

1. capacité de traitement suffisante pour tatouage en temps réel
2. impératif de protection du flux entre la fonction de descrambling (déchiffrement) et le tatouage



Tatouage côté serveur

La version serveur est à privilégier

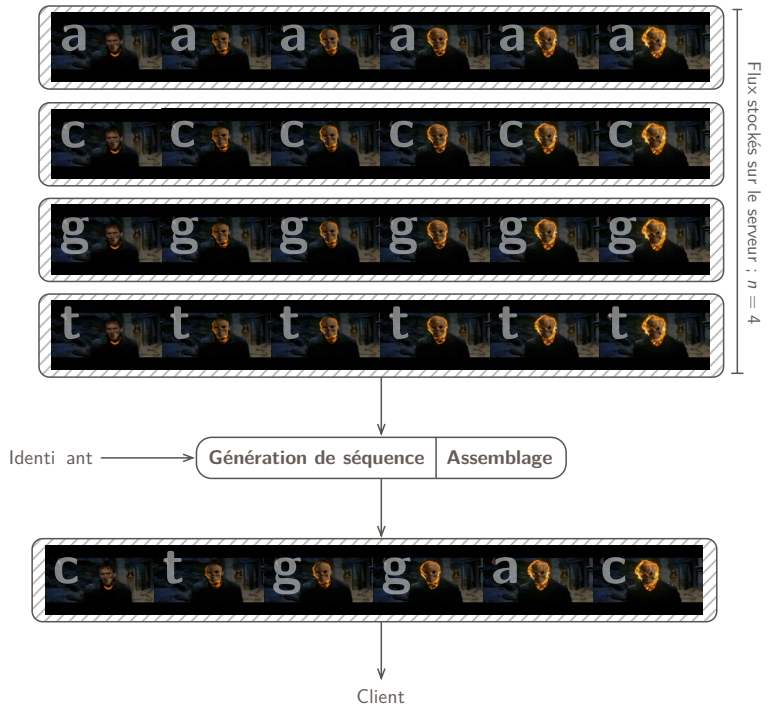
- aucun secret déporté dans les terminaux clients
- aucune contrainte sur la configuration cliente

Mais on se heurte à des problèmes de latence : comme la fonction de tatouage prend en entrée des flux baseband, à chaque transaction

1. le fichier d'origine est décodé
2. tatoué avec l'identifiant client
3. et re-encodé (env. 1h30)

Tatouage par assemblage

- Plusieurs versions du même film sont marquées offline et stockées
- Lors de ma transaction, une séquence est générée à partir de l'identifiant client
- Une nouvelle vidéo est créée par assemblage de blocs issus des n versions
- L'assemblage est une simple manipulation de fichiers



Tatouage par assemblage

- Rapide : plus de 100 Mb/s
- Économique : marquage par film et non par transaction
- Fonctionne avec toute technique de tatouage et tout format vidéo (même fermé ou protégé)
- Le codeur/décodeur de séquences peut avoir un pouvoir de corrections, ce qui augmente la robustesse du tatouage sous-jacent
- Remplit les conditions pour faire du traçage de traitres à la Tardos

Pour conclure

- Tatouage = signal + communications numériques + crypto
- Des avancées qui ont trouvé d'autres applications : codage informé, Tardos
- 2019 sera l'année du tatouage (j'entends ça depuis 10 ans)

- Démonstration et discussion