M1 MEEF NSI — Sécurité des communications informatiques

Chiffrement passif

Pierre Karpman
pierre.karpman@univ-grenoble-alpes.fr
https://membres-ljk.imag.fr/Pierre.Karpman/tea.html

2024–01–26 — 2024–02–09

Objectif : confidentialité

Contexte du moment :

- Deux personnes $\mathscr{A}$ & $\mathscr{B}$ souhaitent communiquer sur un canal fiable
  - Sans perte de généralité, on suppose une com. unidirectionnelle
  - Sans perte de généralité, les messages font exactement 128 bits
- Adversaires passifs

$\rightsquigarrow$ système de chiffrement à évaluer v. sécurité IND-CPA

Comment faire si $\mathscr{A}$ & $\mathscr{B}$:

- Possèdent un grand *secret partagé* ?
- — petit — ?
- Ne possèdent aucun — ?

Chiffrement passif ; grand secret partagé

Chiffrement passif ; petit secret partagé

Chiffrement passif ; aucun secret partagé

Grand secret partagé

Hypothèses :

- $\mathscr{A}$ & $\mathscr{B}$ peuvent convenir d'une « grande » valeur K (par ex. $K \in \{0,1\}^{128\times 2^{128}}$) qu'elles sont les seules à « connaître » a priori
  $\rightsquigarrow$ Cryptographie *symétrique / à clef secrète*
- $\mathscr{A}$ possède un mécanisme réutilisable à volonté permettant de tirer *uniformément* un bit aléatoire
  - Et donc — une chaîne binaire de longueur arbitraire

Objectif :

- Utiliser ces capacités pour construire un système de chiffrement avec un bon niveau de sécurité IND-CPA

Un premier système de chiffrement

(Une instanciation du) *one-time pad*, OTP128 :

1. $\mathscr{A}$ tire uniformément une chaîne $K \in \{0,1\}^{128\times 2^{128}}$ et la partage avec $\mathscr{B}$
2. $\mathscr{A}$ initialise un compteur i à 0
3. Chaque fois que $\mathscr{A}$ souhaite envoyer un message $m \in \{0,1\}^{128}$, elle sélectionne les bits de K_i de K d'indices $i \cdots i+127$ et envoie $(i, m \oplus K_i)$ à $\mathscr{B}$, puis incrémente i de 128. Si $i = 2^{128}-1$, $\mathscr{A}$ ne peut plus envoyer de messages avec ce système.

Remarque : Système probabiliste ; plusieurs chiffrés peuvent correspondre à un même clair

(La même chose avec des fonctions)

1. $\mathscr{A}$ tire uniformément une fonction $\mathsf{K} \in \{0,1\}^{128} \to \{0,1\}^{128}$ et la partage avec $\mathscr{B}$
2. $\mathscr{A}$ initialise un compteur i à 0
3. Chaque fois que $\mathscr{A}$ souhaite envoyer un message $m \in \{0,1\}^{128}$, elle envoie $(i, m \oplus \mathsf{K}(i))$ à $\mathscr{B}$, puis incrémente i de 128. Si $i = 2^{128} - 1$, $\mathscr{A}$ ne peut plus envoyer de messages avec ce système.

Sécurité IND-CPA du OTP

L'analyse de la sécurité IND-CPA du OTP se base sur le résultat fondamental suivant :

Lemme ($\mathcal{U} \oplus * \approx \mathcal{U}$)

Soit X une variables aléatoire sur $\{0,1\}$ suivant une distribution uniforme, Y une variable aléatoire indépendante sur $\{0,1\}$ suivant une distribution quelconque, alors $Z := X \oplus Y$ est uniforme et indépendante de Y.

Preuve $\rightsquigarrow$ TD

Remarque : Ce résultat et ses (multiples) généralisations joue un rôle essentiel dans de très nombreux systèmes cryptographiques

Sécurité IND-CPA de OTP128

- On suppose que l'adversaire A est déterministe (on peut montrer que cela est sans perte de généralité) et fait $q < 2^{128}$ requêtes d'entraînement
- Déf. $\mathcal{O} := \{c_b : \mathsf{A}(\{(x_i, y_i)_{1 \leq i \leq q}\}, m_0, m_1, c_b) = 1\}$
- La probabilité de succès se calcule sur le tirage de b et de K
- $\Pr[\hat{b} = b] = \Pr[\mathsf{A}(\cdots) = 0 \wedge b = 0] + \Pr[\mathsf{A}(\cdots) = 1 \wedge b = 1]$
  $= 1/2 \times (\Pr[\mathsf{A}(\cdots) = 0 : b = 0] + \Pr[\mathsf{A}(\cdots) = 1 : b = 1])$
- $p_1 := \Pr[\mathsf{A}(\cdots) = 1] = \Pr[c_b \in \mathcal{O}] = \#\mathcal{O}/2^{128}$
  $p_0 := \Pr[\mathsf{A}(\cdots) = 0] = \Pr[c_b \in \overline{\mathcal{O}}] = 1 - p_1$
- $\Pr[c_b \in \mathcal{O} : b = 0] = \Pr[c_b \in \mathcal{O} : b = 1] = p_0$
- $\Pr[c_b \in \overline{\mathcal{O}} : b = 0] = \Pr[c_b \in \overline{\mathcal{O}} : b = 1] = p_1$
- $\Pr[\hat{b} = b] = 1/2 \rightsquigarrow \mathbf{Adv}^{\text{IND-CPA}}_{\text{OTP128}}(< 2^{128}, \infty) = 0$

Sécurité IND-CPA de OTP128 (bis)

Remarques :

- $\mathbf{Adv}^{\text{IND-CPA}}_{\text{OTP128}}(< 2^{128}, \infty) = 0 \rightsquigarrow$ le mieux qu'on puisse espérer : quelque soit la puissance de calcul de l'adversaire, son avantage est zéro $\rightsquigarrow$ niveau de sécurité ∞ (quelque soit la définition)
- Parfois appelé sécurité parfaite (avantage zéro) au sens théorie de l'information (puissance de calcul non bornée)
  - **ATTENTION** : nous avons fait des hypothèses sur nos capacités et le modèle d'adversaire
- On obtient aussi un avantage zéro pour des variantes de définitions un peu plus fortes qu'IND-CPA
  - On peut exactement *simuler* OTP128 sans connaître les messages

Chiffrement passif ; grand secret partagé

Chiffrement passif ; petit secret partagé

Chiffrement passif ; aucun secret partagé

Réduire la taille du secret

- Une instanciation d'OTP offre la meilleure sécurité IND-CPA possible
- Mais si on veut échanger beaucoup de données, les hypothèses sur la taille du secret sont déraisonnables
- Mais ∞ bits de sécurité ne sont pas nécessaires ; 128 serait (souvent) suffisant
- Objectif : réduire la taille du secret en gardant un bon niveau sécurité

Petit secret partagé

Hypothèses :

- $\mathscr{A}$ & $\mathscr{B}$ peuvent convenir d'une « petite » valeur K (par ex. $K \in \{0,1\}^{128}$) qu'elles sont les seules à « connaître » a priori
- $\mathscr{A}$ possède un mécanisme réutilisable à volonté permettant de tirer *uniformément* un bit aléatoire

Objectif :

- Utiliser ces capacités pour construire un système de chiffrement avec un bon niveau de sécurité IND-CPA, *possiblement conditionné à des hypothèses supplémentaires à définir*

Les primitives à la rescousse

Idées :

- On sait qu'ajouter de l'aléa uniforme donne une sécurité IND-CPA infinie
- Mais on n'a pas assez d'aléa uniforme pour faire ça pour chaque message
- $\rightsquigarrow$ « étendre » notre petit aléa uniforme en un grand aléa *presque uniforme* et utiliser ce dernier ?
- $\rightsquigarrow$ utiliser pour cela une bonne (famille de) *fonction pseudo-aléatoire* (une *primitive*)
- $\rightsquigarrow$ si la fonction est « bonne » (pour un sens **bien défini**), alors on obtient une « bonne » sécurité IND-CPA

Famille de fonction pseudo-aléatoire : syntaxe

- On considère généralement des fonctions $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$: une famille de fonction paramétrée par un paramètre (généralement appelé la *clef*) : pour tout $k \in \{0,1\}^\kappa$, $\mathsf{F}(k,\cdot)$ est une fonction $\{0,1\}^n \to \{0,1\}^n$
- (Mais on peut parfois souhaiter avoir des entrées ou des sorties de taille variable)

Chiffrement à base de fonction pseudo-aléatoire

1. $\mathscr{A}$ et $\mathscr{B}$ se mettent *publiquement* d'accord sur une famille de fonction $\mathsf{F} : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$
2. $\mathscr{A}$ tire uniformément une clef $K \in \{0,1\}^{128}$ et la partage avec $\mathscr{B}$
3. $\mathscr{A}$ initialise un compteur i à 0
4. Chaque fois que $\mathscr{A}$ souhaite envoyer un message $m \in \{0,1\}^{128}$, elle envoie $(i, m \oplus \mathsf{F}(K,i))$ à $\mathscr{B}$, puis incrémente i de 128. Si $i = 2^{128} - 1$, $\mathscr{A}$ ne peut plus envoyer de messages avec ce système.

$\rightsquigarrow$ une instance du *mode compteur* (CTR) pour la (famille de) fonction F (notation : CTR[F])

Famille de fonction pseudo-aléatoire : sécurité

Idées :

- La seule différence entre OTP128 = CTR[K] et CTR[F] est qu'on a remplacé $K \twoheadleftarrow \text{Funcs}(\{0,1\}^{128})$ par $F(K,\cdot), K \twoheadleftarrow \{0,1\}^{128}$
  - Notation : Pour un ensemble fini $\mathcal{S}$, $\text{Funcs}(\mathcal{S})$ dénote l'ensemble des fonctions $\mathcal{S} \to \mathcal{S}$
  - Notation : Pour un ensemble fini $\mathcal{S}$, $X \twoheadleftarrow \mathcal{S}$ dénote le fait que X est un élément tiré uniformément dans $\mathcal{S}$, a priori indépendamment de tout autre tirage
- Si on suppose qu'il est difficile pour tout adversaire de distinguer K et F, alors il sera difficile de distinguer CTR[F] de OTP128, qui a une sécurité infinie
  - Preuve par *réduction* : on réduit la sécurité IND-CPA de CTR[F] à la sécurité *PRF* de F

Sécurité PRF

On définit la sécurité PRF (pour *pseudorandom function*) d'une famille de fonction F par l'intermédiaire de la fonction d'avantage suivante :

$\mathbf{Adv}^{\mathrm{PRF}}$

$$\mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{F}}(q,t) = \max_{A_{q,t}} \Big| \Pr[A^{\mathbb{O}}_{q,t}() = 1 : \mathbb{O} \twoheadleftarrow \mathsf{Funcs}(\{0,1\}^n)] - \Pr[A^{\mathbb{O}}_{q,t}() = 1 : \mathbb{O} = \mathsf{F}(K,\cdot), K \twoheadleftarrow \{0,1\}^\kappa] \Big|$$

Remarque : Par abus de langage, on dit souvent que F **est** une PRF pour dire qu'elle a une « bonne » sécurité PRF. (Pareil pour « Enc un chiffrement IND-CPA »)

Sécurité PRF : remarques (bis)

- Pour tout $\kappa \lessapprox n2^n$, quelque soit F on peut gagner $\mathbf{Adv}_{\mathsf{F}}^{\mathsf{PRF}}$ avec un avantage constant en utilisant suffisamment de ressources q et t
- Par exemple pour $\kappa = n$, $\mathbf{Adv}_{\mathsf{F}}^{\mathsf{PRF}}(2, 2^n) \approx 1$ (Exercice : le montrer)
- On parle d'attaque *générique* : ce sont « les paramètres » qui sont attaqués, pas la fonction spécifiquement

$\rightsquigarrow$ Il faut faire **ATTENTION** à choisir des paramètres non vulnérables aux attaques génériques, cf. par ex. les ordres de grandeur de calcul/avantage déjà donnés

Chiffrement à base de fonction pseudo-aléatoire : sécurité

- On peut montrer que $\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTR[F]}}(q,t) \approx \mathbf{Adv}^{\text{PRF}}_{\text{F}}(q,t)$, pour $q < 2^n$
- $\rightsquigarrow$ Il suffit d'une bonne PRF pour pouvoir construire un bon chiffrement IND-CPA
- « N'importe quelle » bonne PRF $\rightsquigarrow$ modularité
- De façon générale, les systèmes de chiffrement se construisent comme des *modes opératoires* pour des fonctions ou des *chiffres par bloc* (cf. ci-dessous)

Famille de fonction pseudo-aléatoire : construction

- On sait plus facilement construire des familles de *permutations* pseudo-aléatoires que de fonctions arbitraires : des chiffres (ou chiffrement) par bloc (*block cipher*)

Définition : permutation

Une permutation est une fonction bijective d'un ensemble fini vers lui-même. Il y a $N!$ permutations distinctes sur un ensemble de N éléments.

Définition : chiffre par bloc

Un chiffre par bloc est une famille de permutations : une fonction $\mathsf{E} : \{0,1\}^\kappa \times \mathcal{M} \to \mathcal{M}$ t.q. $\forall k \in \{0,1\}^\kappa$ on a que $\mathsf{E}(k, \cdot)$ est une permutation

Remarque : en général, $\mathcal{M} = \{0,1\}^n$ pour $n \in \{64, 128, 256\}$

Sécurité PRP

On définit la sécurité PRP (pour *pseudorandom permutation*) d'un chiffre par bloc E par l'intermédiaire de la fonction d'avantage suivante :

$\mathbf{Adv}^{\mathrm{PRP}}$

$$\mathbf{Adv}^{\mathrm{PRP}}_{\mathsf{E}}(q,t) = \max_{A_{q,t}} \Big| \Pr[A^{\mathbb{O}}_{q,t}() = 1 : \mathbb{O} \twoheadleftarrow \mathrm{Perms}(\{0,1\}^n)] - \Pr[A^{\mathbb{O}}_{q,t}() = 1 : \mathbb{O} = \mathsf{E}(K,\cdot), K \twoheadleftarrow \{0,1\}^\kappa] \Big|$$

- $\mathrm{Perms}(\mathcal{S})$: l'ensemble des permutations sur $\mathcal{S}$

PRP/PRF switching

- Si l'on souhaite remplacer une PRF par une PRP, il faut qu'une bonne PRP soit aussi une bonne PRF, ce qui est bien le cas :

Lemme (PRP/PRF switching)

Soit E une famille de permutations sur N éléments, on a :

$$\mathbf{Adv}_{\mathsf{E}}^{\mathsf{PRF}}(q,t) \leq \mathbf{Adv}_{\mathsf{E}}^{\mathsf{PRP}}(q,t) + \frac{q(q-1)}{2N}$$

Remarques :

- Le terme $q(q-1)/2N$ est *générique* (ne dépend pas de E)
- C'est un terme en « anniversaires » (cf. le « paradoxe des anniversaires »), et l'inégalité devient vide de sens « à la borne des anniversaires », c-à-d pour $q \approx \sqrt{N}$
- La borne est *tight* (pour $q \leq \sqrt{2N}$, $t \propto q$, minoration par $q(q-1)/4N$)

Sécurité du mode compteur avec chiffres par blocs

- Des résultats précédents, on obtient :

$$\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTR[E]}}(q,t) \approx \mathbf{Adv}^{\text{PRF}}_{\text{E}}(q,t) \lessapprox \mathbf{Adv}^{\text{PRP}}_{\text{E}}(q,t) + \frac{q(q-1)}{2N}$$

- $\rightsquigarrow$ Il suffit de (n'importe quelle) bonne PRP pour pouvoir construire un bon chiffrement IND-CPA
- Minoration aussi possible (cf. ci-dessus) : la sécurité s'effondre à la borne des anniversaires
- $\rightsquigarrow$ La sécurité (IND-CPA) du mode compteur dépend de la sécurité (PRP) du chiffre par bloc, **MAIS AUSSI** de la quantité de données chiffrées
  - $\rightsquigarrow$ Il faut arrêter de communiquer / changer de clef *bien avant* d'atteindre $q \approx \sqrt{N}$

Sécurité aux anniversaires : impact

Limitation aux anniversaires : application numérique

- E avec des blocs de 64 bits
  - $\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTR[E]}}(2^{10}, 2^{10}) \gtrapprox 2^{-46}$ (64 Kb de chiffrés)
  - $\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTR[E]}}(2^{20}, 2^{20}) \gtrapprox 2^{-26}$ (64 Mb de chiffrés)
  - $\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTR[E]}}(2^{30}, 2^{30}) \gtrapprox 2^{-6}$ (64 Gb de chiffrés)
- — 128 bits
  - $\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTR[E]}}(2^{30}, 2^{30}) \gtrapprox 2^{-70}$ (128 Gb de chiffrés)
  - $\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTR[E]}}(2^{60}, 2^{60}) \gtrapprox 2^{-10}$ (128 Eb de chiffrés)
- — 256 bits
  - $\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTR[E]}}(2^{60}, 2^{60}) \gtrapprox 2^{-138}$ (128 Eb de chiffrés)
  - $\mathbf{Adv}^{\text{IND-CPA}}_{\text{CTR[E]}}(2^{80}, 2^{80}) \gtrapprox 2^{-98}$ (128 Yb de chiffrés)

$\rightsquigarrow$ Pas de soucis avec de gros blocs, mais attention aux petits ! ! !

$\rightsquigarrow$ Peut mener à des attaques concrètes, cf. par ex.

`https://sweet32.info/`

Quelques exemples de chiffres par bloc

- AES (“Advanced Encryption Standard”) : blocs de 128 bits ; clefs de 128, 192, 256 bits
  - Standard du NIST (USA) : FIPS 197 (2001)
  - Versatile, bonne performance, très analysé, pas de vulnérabilités connues (en contexte normal d’utilisation)
- PRESENT : blocs de 64 bits ; clefs de 80 ou 128 bits
  - Un exemple de *lightweight block cipher* : peu coûteux à implémenter en circuit
- SPECK : blocs de 48 à 128 bits ; clefs de 96 à 128 bits
  - Autre exemple de *lightweight block cipher* : peu coûteux à implémenter en logiciel (sur petites architectures). Attention aux très petits blocs !
- SHACAL-2 : blocs de 256 bits ; clefs de 512 bits
  - Un exemple de chiffre par bloc avec de gros blocs et une très grosse clef

Quelques remarques de conclusion intermédiaire

- Chiffrement symétrique (IND-CPA) à base de chiffres par blocs (PRP) grâce à des modes opératoires : une approche (très) courante, mais pas la seule !
  - Exemple d'alternative : chiffrement à base de permutation, par ex. ASCON (en cours de standardisation par le NIST)
- Définitions de sécurité pour fonctions et chiffres par blocs : PRF et PRP mais pas que !
  - D'autres définitions & modèles existent, par ex. l'*unpredictability* (UP, cf. plus bas) ou des modèles idéaux (cf. OTP, version fonction)
- Rappel : la sécurité IND-CPA considère seulement des adversaires *passifs*, relativement faibles
  - Mais un chiffrement IND-CPA est un bon point de départ pour résister aussi à des adversaires actifs (mais ce n'est pas le seul !), cf. plus tard

Quelques remarques de conclusion intermédiaire (bis)

En pratique, le chiffrement symétrique est (très) efficace. Quelques ordres de grandeurs (pour les algorithmes appropriés) :

- Sur une architecture puissante : à peine quelques cycles CPU pour chiffrer un octet
- Sur une architecture embarqué : à peine quelques centaines d'octets pour implémenter le chiffrement et quelques dizaines de cycles par octet (Attention : sans protection contre les canaux auxiliaires)
- Sur un circuit : à peine quelques milliers de portes pour implémenter le chiffrement (Attention : sans protection contre les canaux auxiliaires)

Chiffrement passif ; grand secret partagé

Chiffrement passif ; petit secret partagé

Chiffrement passif ; aucun secret partagé

Réduire la taille du secret

- L'ensemble des techniques de chiffrement symétrique donnent une solution satisfaisante
- Mais cela présuppose la connaissance d'un secret partagé
  - Envisageable dans certains cas, par ex. pour des communications personnelles dans un cadre restreint
  - Mais pas à large échelle (cf. tous les contextes d'utilisation de cryptographie)
- Objectif : réduire la taille du secret partagé à zéro !

Aucun secret partagé

Hypothèses :

- $\mathscr{A}$ et $\mathscr{B}$ possèdent un mécanisme réutilisable à volonté permettant de tirer *uniformément* un bit aléatoire

Objectif :

- Utiliser cette capacité pour construire un système de chiffrement avec un bon niveau de sécurité IND-CPA, *possiblement conditionné à des hypothèses supplémentaires à définir*

Les primitives & protocoles à la rescousse

Idées :

- Les techniques précédentes sont suffisantes à condition de connaître un secret partagé
- Il suffit donc pour $\mathscr{A}$ et $\mathscr{B}$ d'être capable d'« apprendre » un secret partagé

$\rightsquigarrow$ Protocoles d'échange de clefs (*key exchange*) ou d'*encapsulation de clef* (*key encapsulation mechanism* : KEM)

$\rightsquigarrow$ cryptographie *asymétrique / à clef publique*

Vocabulaire (informellement) : protocole : algorithme avec interaction entre différents participants

Protocole d'échange de clef

Informellement :

1. $\mathscr{A}$ (resp. $\mathscr{B}$) génère un paramètre $k_{\mathscr{A}}$ (resp. $k_{\mathscr{B}}$) connu d'elle seule
2. $\mathscr{A}$ et $\mathscr{B}$ échangent publiquement des messages $\mathcal{M} := \{m_i\}$
3. $\mathscr{A}$ (resp. $\mathscr{B}$) caclule le secret partagé $k_{\mathscr{A}\mathscr{B}}$ comme $\mathsf{K}(k_{\mathscr{A}}, \mathcal{M})$ (resp. $\mathsf{K}(k_{\mathscr{B}}, \mathcal{M})$) (pour une certaine fonction publique K)

Vocabulaire : $\mathcal{M}$ forme le *transcript* d'une exécution du protocole

Vocabulaire : le protocole est *non-interactif* (NIKE) si $\mathscr{A}$ et $\mathscr{B}$ envoient un seul message « en parallèle »

Objectif de fonctionnalité : $\mathscr{A}$ et $\mathscr{B}$ obtiennent le même secret à l'issue du protocole

Objectifs de sécurité : ??

Sécurité passive d'un échange de clef

Modèle d'adversaire : adversaire passif (pour l'instant)

- Connaît les messages échangés et rien de plus $\rightsquigarrow$ *eavesdropper* $\rightsquigarrow$ "EAV"
- (L'adversaire peut aussi faire tourner le protocole avec $\mathscr{A}$ ou $\mathscr{B}$, ce qui permet parfois des attaques)
- (Mais plus d'accès à un oracle de chiffrement comme en CPA : protocole « fixe » ne dépendant d'aucun environnement autre que celui accessible comme ci-dessus, donc pas pertinent)

Succès de l'adversaire : plusieurs options, par ex :

- UP (*unpredictability*) : L'adversaire est incapable de prédire la valeur prise par $k_{\mathscr{A}\mathscr{B}}$
- IND (*indistinguishability*) : L'adversaire est incapable de distinguer $k_{\mathscr{A}\mathscr{B}}$ d'un aléa uniforme de même taille

Remarque : IND « plus fort » que UP, mais UP suffisant ici

Jeu de sécurité UP-EAV

Pour un protocole (publique) d'échange de clef KE :

Jeu UP-EAV

1. L'adversaire peut acquérir de l'information en exécutant KE avec $\mathscr{A}$ ou $\mathscr{B}$
2. Une fois cet entraînement terminé, $\mathscr{A}$ et $\mathscr{B}$ exécutent KE en produisant un transcript $\mathcal{M}$ donné à l'adversaire, et obtiennent un secret partagé $k_{\mathscr{A}\mathscr{B}}$
3. L'adversaire essaye de deviner $k_{\mathscr{A}\mathscr{B}}$: il renvoie $\hat{k}$ et gagne le jeu ssi. $\hat{k} = k_{\mathscr{A}\mathscr{B}}$

Remarque : l'adversaire peut (comme toujours) être probabiliste, ce qui est utile s'il « hésite » entre plusieurs valeurs

On définit ensuite l'*insecurity* d'un protocole relativement à ce jeu

Fonction $\mathbf{InSec}^{\text{UP-EAV}}$

L'insécurité UP-EAV d'un protocole d'échange de clef KE peut se définir comme :

$\mathbf{InSec}^{\text{UP-EAV}}(q, t)$

$$\mathbf{InSec}^{\text{UP-EAV}}_{\text{KE}}(q, t) := \max_{A_{q,t}} \Pr[A_{q,t} \text{ gagne le jeu UP-EAV contre KE}]$$

$A_{q,t}$: un adversaire qui fait q exécutions d'entraînement et qui tourne en temps t

Remarques :

- Attaque générique : si KE est tel que $k_{\mathscr{A}\mathscr{B}}$ appartient à un ensemble de taille N, on a trivialement $\mathbf{InSec}^{\text{UP-EAV}}_{\text{KE}}(0, 1) \geq 1/N$
- (Fonctionne quelque soit la façon dont le secret à deviner a été construit)

$\mathbf{InSec}^{\text{UP-EAV}}$: remarques (bis)

- Un protocole « parfait » serait tel que $\mathbf{InSec}^{\text{UP-EAV}}(\infty, \infty) = 1/N$
  - Atteignable (même contre des adversaires actifs !) avec des protocoles de "*perfectly-secure message transmission*" (PSMT), qui font cependant des hypothèses fortes sur les adversaires ! !
- De façon générale, suffisant que $\mathbf{InSec}^{\text{UP-EAV}}$ ne diminue pas (trop) l'effort d'un adversaire par rapport au cas où $k_{\mathscr{A}\mathscr{B}}$ est un secret partagé initial (cf. ci-dessus)
- Par exemple, $\mathbf{InSec}^{\text{UP-EAV}}(\infty, t) \lessapprox t/N$ est « raisonnable »

Une famille de protocole d'échange de clef : DH

DH ?

Figure – Photo : JB Liautard

Une famille de protocole d'échange de clef : DH

DH : « Diffie-Hellman »

- Famille de protocoles basée sur l'exponentiation dans un groupe cyclique : chaque tel groupe $\mathbb{G}$ permet une instanciation du protocole
- Condition *nécessaire* de sécurité : le problème du *logarithme discret* doit être « difficile » dans $\mathbb{G}$

Détails ci-dessous...

Groupes cycliques : définitions

Groupe cyclique

Un *groupe cyclique* $\mathbb{G}$ d'*ordre* (ou cardinalité, ou taille) $N \in \mathbb{N} \setminus \{0\}$ est un ensemble $\mathcal{G} := \{1 := g^0, g, \ldots, g^{N-1}\}$ muni d'une opération $\times$ définie pour tout $(g^a, g^b) \in \mathcal{G}^2$ par $g^a \times g^b \mapsto g^{(a+b)\%N}$

% : le « modulo » informatique pour les entiers naturels : $n\%m$ est le reste (positif) de la division de n par m

Vocabulaire : ci-dessus, g est un *générateur* du groupe

Notation : on peut utiliser $\langle g \rangle$ pour dénoter le groupe généré par g comme g^0, g, g^2 etc. Ce groupe est cyclique ssi. il existe un entier N t.q. $g^N = g^0$

Logarithme discret

Soit $\mathbb{G} := \langle g \rangle$ un groupe cyclique, le *logarithme discret* en base g d'un élément $h \in \mathbb{G}$ est l'entier $a \in [\![0, N-1]\!]$ t.q. $h = g^a$

Groupes cycliques : exemples

Deux exemples :

- $(\mathbb{Z}/N\mathbb{Z}, +)$: les entiers modulo N avec l'addition ; logarithme discret $\equiv$ division : pas adapté pour DH
- $(\mathbb{Z}/p\mathbb{Z})^\times$: les entiers (sauf zéro) modulo un *nombre premier* p avec la multiplication ; (a priori) plutôt adapté pour DH

Les instanciations typiques de DH utilisent aussi parfois d'autres groupes, plus difficiles à décrire

Le protocole de NIKE DH

DH : version basique

$\mathbb{G}$, un groupe cyclique d'ordre N et g un de ses générateurs sont connus publiquement

1. $\mathscr{A}$ tire $a \twoheadleftarrow [\![0, N-1]\!]$, calcule g^a, et l'envoie à $\mathscr{B}$
2. $\mathscr{B}$ tire $b \twoheadleftarrow [\![0, N-1]\!]$, calcule g^b, et l'envoie à $\mathscr{A}$
3. $\mathscr{A}$ calcule $k = (g^b)^a = g^{ab}$ et définit $k_{\mathscr{A}\mathscr{B}} = k$
4. $\mathscr{B}$ calcule $k = (g^a)^b = g^{ab}$ —

Correction : Immédiate d'après la définition d'un groupe cyclique

Remarque : La version basique présentée ici correspond à DH « éphémère » : $\mathscr{A}$ (resp. $\mathscr{B}$) tire un nouveau a (resp. b) à chaque exécution du protocole. On peut aussi définir une version « statique » ou « semi-statique » qui a des avantages pour la performance, mais certains inconvénients pour la sécurité

Le protocole DH : sécurité

Soit une exécution du protocole précédent « DH[$\mathbb{G}$] » instancié avec $\mathbb{G}$

- Un adversaire UP-EAV a la connaissance de $\mathbb{G}$, g, g^a, g^b, et cherche à deviner g^{ab}
- Ceci n'*est pas plus dur* que de calculer un logarithme discret quelconque (et une exponentiation) dans $\mathbb{G}$ (Exercice : écrire l'algorithme sous-jacent)
- $\rightsquigarrow$ il est **NÉCESSAIRE** que le problème « DLOG[$\mathbb{G}$] » de calculer un logarithme discret soit « difficile » dans $\mathbb{G}$ (pour obtenir une « bonne » sécurité)
- Mais est-ce suffisant ?

Sécurité de DH : le problème CDH

- On définit le problème *Computational Diffie-Hellman* (CDH) pour un groupe $\mathbb{G}$ comme le jeu UP-EAV pour DH[$\mathbb{G}$] (avec zéro entraînement $\leftarrow$ pas de différence pour le cas éphémère, mais possiblement un impact pour certaines variantes (semi-)statiques du protocole)
- $\rightsquigarrow \mathbf{InSec}^{\text{UP-EAV}}_{\text{DH}[\mathbb{G}]}(q, t) \approx \mathbf{InSec}^{\text{CDH}}_{\mathbb{G}}(t)$
- On peut montrer (Maurer, 1994) que pour certains groupes (ou tous, en fonction du point de vue) $\mathbf{InSec}^{\text{DLOG}}_{\mathbb{G}}(t) \approx \mathbf{InSec}^{\text{CDH}}_{\mathbb{G}} \rightsquigarrow \mathbf{InSec}^{\text{UP-EAV}}_{\text{DH}[\mathbb{G}]}(q, t) \approx \mathbf{InSec}^{\text{DLOG}}_{\mathbb{G}}(t)$: une *réduction* de la sécurité de DH[$\mathbb{G}$] au problème du logarithme discret dans ce groupe

Remarque : On rencontre aussi couramment le problème *Decisional Diffie-Hellman* (DDH) qui s'obtient comme ci-dessus en considérant $\mathbf{Adv}^{\text{IND-EAV}}_{\text{DH}[\mathbb{G}]}$; dans ce cas on sait montrer pour certains groupes que DDH est bien plus facile que DLOG ou CDH

Difficulté de DLOG

- Les meilleurs algorithmes *génériques* permettent de résoudre DLOG (c-à-d trouver le logarithme d'un élément uniforme avec probabilité constante) dans un groupe d'ordre N en $\Theta(\sqrt{N})$ opérations dans le groupe (par ex. *baby step / giant step*, algorithme des kangourous...)
- Mais ce problème est *plus simple* dans certains groupes utilisés en pratique : il faut (actuellement) p d'environ 3000 bits pour que DLOG dans $(\mathbb{Z}/p\mathbb{Z})^\times$ ait une sécurité d'environ 128 bits
  - **ATTENTION** (à nouveau) au choix des paramètres des systèmes utilisés !

Remarque : DLOG est *facile* à résoudre pour n'importe quel groupe si l'on dispose d'un ordinateur quantique suffisamment puissant. Développer des systèmes cryptographiques restant (a priori) sûrs sous cette hyopothèse est le rôle de la cryptographie « post-quantique »

DH : efficacité ?

- L'exponentiation dans $\mathbb{G}$ est « efficace » ssi la loi de groupe l'est
  - L'algorithme naïf d'exponentiation est *terriblement inefficace* (Q : quel est son coût ?), mais on peut facilement utiliser un algorithme « d'exponentiation rapide »
  - Mais l'efficacité de DH dépend néamoins de celle de $\mathbb{G}$ $\rightsquigarrow$ trouver des groupes (a priori) sûrs et « efficaces » est un enjeu pratique très important
- En pratique (comme tout NIKE connu) même dans le meilleur cas DH coûte beaucoup plus cher que du chiffrement symétrique : au moins plusieurs dizaines de milliers de cycles pour un échange
  - *Relativement* cher mais tout à fait raisonnable

NIKE, KEM : alternatives

DH est très utilisé en pratique, mais il existe des alternatives !

- RSA version KEM (l'autre grand système historique)
  - Sécurité *à peu près* basée sur la difficulté de la factorisation
  - A aussi besoin d'un bon « mode opératoire » pour être sûr
- KEM post-quantiques, par ex. Kyber (en cours de standardisation / déploiement)
- NIKE post-quantique, par ex. CSIDH

Conclusion

- Chiffrement symétrique + échange de clef permettent (a priori) de résoudre le problème de communication confidentielle contre des adversaires *passifs*
- $\rightsquigarrow$ La *base* de beaucoup de systèmes utilisés en pratique, notamment TLS
- Mais tout cela s'*écroule* complètement contre des adversaires *actifs*
- Le travail n'est pas fini...