Introduction aux protocoles cryptographiques

Thomas Genet

genet@irisa.fr

http://www.irisa.fr/celtique/genet/Crypt/

Plan

- Bibliographie
- Partie I : protocoles crypto, mécanismes et propriétés
  - Trois mots de cryptographie
  - Propriétés de sécurité
    - ★ associées aux fonctions cryptographiques
    - ★ associées aux protocoles
  - Principes de base des protocoles
  - Protocoles d'échange de clés
  - Protocoles d'authentification
  - Protocole d'accord non-répudiable
  - Paiement électronique
    - ★ SET/CSET de VISA et MasterCard
    - ★ ≪ Monnaie électronique ≫ : Bitcoin

Plan (suite)

- Partie II : protocoles crypto, attaques et vérification
  - Mises en garde
  - Le paiement avec une carte bancaire à puce
  - Les failles
  - Construction des hypothèses de vérification des protocoles cryptographiques
  - Le modèle de Dolev-Yao
  - Deux mots de vérification

Bibliographie

- S. Goldwasser and M. Bellare. Lecture Notes on Cryptography.
  `http://www.cs.ucsd.edu/users/mihir/papers/gb.html`
- J. Clark and J. Jacob. *A survey of Authentication Protocol Litterature*. 1997.

`http://www.cs.york.ac.uk/%7Ejac/PublishedPapers/reviewV1_1997.pdf`

- Jean-Paul Delahaye. *Bitcoin, la cryptomonnaie*.
  `http://www.lifl.fr/~delahaye/pls/2013/241.pdf`. 2013.
- Véronique Cortier. *Vérification automatique des protocoles cryptographiques*. Thèse de doctorat de l'École Normale Supérieure de Cachan 2003.
  `http://www.loria.fr/~cortier/Papiers/Cortier-these.pdf`

Trois mots de cryptographie

Soient :

- m, m_1, m_2 des messages
- K une clé
- A et B des agents

Trois mots de cryptographie

Soient :

- m, m_1, m_2 des messages
- K une clé
- A et B des agents

On note :

- m_1, m_2 le message constitué de m_1 et m_2

Trois mots de cryptographie

Soient :

- m, m_1, m_2 des messages
- K une clé
- A et B des agents

On note :

- m_1, m_2 le message constitué de m_1 et m_2
- $\{m\}_K$ le message m chiffré avec K.

Trois mots de cryptographie

Soient :

- m, m_1, m_2 des messages
- K une clé
- A et B des agents

On note :

- m_1, m_2 le message constitué de m_1 et m_2
- $\{m\}_K$ le message m chiffré avec K.
- $A \hookrightarrow B : m$ l'envoi par A d'un message m à B.

Trois mots de cryptographie

Soient :

- m, m_1, m_2 des messages
- K une clé
- A et B des agents

On note :

- m_1, m_2 le message constitué de m_1 et m_2
- $\{m\}_K$ le message m chiffré avec K.
- $A \hookrightarrow B : m$ l'envoi par A d'un message m à B.
- $I(A) \hookrightarrow B : m$ l'envoi d'un message m par I se faisant passer pour A

Trois mots de cryptographie

Trois mots de cryptographie

- Chiffrement à clé asymétrique (Ex. RSA, PGP, . . .) :

Trois mots de cryptographie

- Chiffrement à clé asymétrique (Ex. RSA, PGP, ...) :
  - la clé publique (K_A) donnée à tous les acteurs

Trois mots de cryptographie

- Chiffrement à clé asymétrique (Ex. RSA, PGP, ...) :
  - la clé publique (K_A) donnée à tous les acteurs
  - la clé privée (K_A^{-1}) connue de A seul

Trois mots de cryptographie

- Chiffrement à clé asymétrique (Ex. RSA, PGP, ...) :
  - la clé publique (K_A) donnée à tous les acteurs
  - la clé privée (K_A^{-1}) connue de A seul
  - $\{\{m\}_{K_A}\}_{K_A^{-1}} = m = \{\{m\}_{K_A^{-1}}\}_{K_A}$

Trois mots de cryptographie

- Chiffrement à clé asymétrique (Ex. RSA, PGP, ...) :
  - la clé publique (K_A) donnée à tous les acteurs
  - la clé privée (K_A^{-1}) connue de A seul
  - $\{\{m\}_{K_A}\}_{K_A^{-1}} = m = \{\{m\}_{K_A^{-1}}\}_{K_A}$
  - inutilisable pour chiffrer de gros volumes de données

Trois mots de cryptographie

- Chiffrement à clé asymétrique (Ex. RSA, PGP, ...) :
  - la clé publique (K_A) donnée à tous les acteurs
  - la clé privée (K_A^{-1}) connue de A seul
  - $\{\{m\}_{K_A}\}_{K_A^{-1}} = m = \{\{m\}_{K_A^{-1}}\}_{K_A}$
  - inutilisable pour chiffrer de gros volumes de données
  - deux clés par acteur

Trois mots de cryptographie

- Chiffrement à clé asymétrique (Ex. RSA, PGP, . . .) :
  - la clé publique (K_A) donnée à tous les acteurs
  - la clé privée (K_A^{-1}) connue de A seul
  - $\{\{m\}_{K_A}\}_{K_A^{-1}} = m = \{\{m\}_{K_A^{-1}}\}_{K_A}$
  - inutilisable pour chiffrer de gros volumes de données
  - deux clés par acteur
- Chiffrement à clé symétrique (Ex. DES) :

Trois mots de cryptographie

- Chiffrement à clé asymétrique (Ex. RSA, PGP, ...) :
  - la clé publique (K_A) donnée à tous les acteurs
  - la clé privée (K_A^{-1}) connue de A seul
  - $\{\{m\}_{K_A}\}_{K_A^{-1}} = m = \{\{m\}_{K_A^{-1}}\}_{K_A}$
  - inutilisable pour chiffrer de gros volumes de données
  - deux clés par acteur
- Chiffrement à clé symétrique (Ex. DES) :
  - $K_{AB} \equiv K_{AB}^{-1}$

Trois mots de cryptographie

- Chiffrement à clé asymétrique (Ex. RSA, PGP, . . .) :
  - la clé publique (K_A) donnée à tous les acteurs
  - la clé privée (K_A^{-1}) connue de A seul
  - $\{\{m\}_{K_A}\}_{K_A^{-1}} = m = \{\{m\}_{K_A^{-1}}\}_{K_A}$
  - inutilisable pour chiffrer de gros volumes de données
  - deux clés par acteur

- Chiffrement à clé symétrique (Ex. DES) :
  - $K_{AB} \equiv K_{AB}^{-1}$
  - bon rapport $\dfrac{\text{volume de données à chiffrer}}{\text{temps de chiffrement}}$

Trois mots de cryptographie

- Chiffrement à clé asymétrique (Ex. RSA, PGP, ...) :
  - la clé publique (K_A) donnée à tous les acteurs
  - la clé privée (K_A^{-1}) connue de A seul
  - $\{\{m\}_{K_A}\}_{K_A^{-1}} = m = \{\{m\}_{K_A^{-1}}\}_{K_A}$
  - inutilisable pour chiffrer de gros volumes de données
  - deux clés par acteur

- Chiffrement à clé symétrique (Ex. DES) :
  - $K_{AB} \equiv K_{AB}^{-1}$
  - bon rapport $\dfrac{\text{volume de données à chiffrer}}{\text{temps de chiffrement}}$
  - une clé par couple d'acteurs (C_n^2 clés)

Trois mots de cryptographie

Trois mots de cryptographie

- Fonctions de hachage (Ex. SHA, MD4, MD5, . . .)

Trois mots de cryptographie

- Fonctions de hachage (Ex. SHA, MD4, MD5, . . .)
  - $| hash(d) | << | d |$

Trois mots de cryptographie

- Fonctions de hachage (Ex. SHA, MD4, MD5, . . .)
  - $| \, hash(d) \, | << | \, d \, |$

et **statistiquement :**

  - $\forall d_1, d_2 : \; hash(d_1) \neq hash(d_2) \quad$ si $\quad d_1 \neq d_2$

Trois mots de cryptographie

- Fonctions de hachage (Ex. SHA, MD4, MD5, . . .)
  - $| \, hash(d) \, | << | \, d \, |$

et **statistiquement :**

  - $\forall d_1, d_2 : \; hash(d_1) \neq hash(d_2) \quad$ si $\quad d_1 \neq d_2$
  - non inversible : $hash^{-1}$ difficile à calculer

Interlude : RSA pour les nuls

- $K_B = (e = 1367, n = 8633)$
- $K_B^{-1} = (e^{-1} = 5735)$
- $m = 8000$

Interlude : RSA pour les nuls

- $K_B = (e = 1367, n = 8633)$
- $K_B^{-1} = (e^{-1} = 5735)$
- $m = 8000$

Chiffrement

$$\begin{array}{rcl} \{m\}_{K_B} & = & m^e \ mod \ n \\ & = & 8000^{1367} \ mod \ 8633 \\ & = & 7633 \end{array}$$

Interlude : RSA pour les nuls

- $K_B = (e = 1367, n = 8633)$
- $K_B^{-1} = (e^{-1} = 5735)$
- $m = 8000$

Chiffrement

$$\begin{aligned} \{m\}_{K_B} &= m^e \bmod n \\ &= 8000^{1367} \bmod 8633 \\ &= 7633 \end{aligned}$$

Déchiffrement

$$\begin{aligned} \{\{m\}_{K_B}\}_{K_B^{-1}} &= (m^e)^{e^{-1}} \bmod n \\ &= 7633^{5735} \bmod 8633 \\ &= 8000 \end{aligned}$$

Propriétés associées à la cryptographie

Propriétés associées à la cryptographie

- Confidentialité
  - $\{\text{“4976 0974 2373 7788”}\}_{K_B}$
  - $\{\texttt{recette\_teurgoule.ps}\}_{K_{AB}}$

Propriétés associées à la cryptographie

- Confidentialité
  - $\{$ “4976 0974 2373 7788” $\}_{K_B}$
  - $\{$`recette_teurgoule.ps`$\}_{K_{AB}}$
- Authentification de message (Signature électronique)
  - “genet@irisa.fr”, $\{$ “genet@irisa.fr” $\}_{K_{genet}^{-1}}$
  - `toto.gif`, $\{$ *hash*(`toto.gif`) $\}_{K_{genet}^{-1}}$

Propriétés associées à la cryptographie

- Confidentialité
  - $\{$ “4976 0974 2373 7788” $\}_{K_B}$
  - $\{\texttt{recette\_teurgoule.ps}\}_{K_{AB}}$
- Authentification de message (Signature électronique)
  - “genet@irisa.fr”, $\{$ “genet@irisa.fr” $\}_{K_{genet}^{-1}}$
  - $\texttt{toto.gif}$, $\{\ hash(\texttt{toto.gif})\ \}_{K_{genet}^{-1}}$
- Intégrité
  - Le contenu m d'un message chiffré $\{m\}_K$ ne peut être modifié sans K

Propriétés des protocoles crypto

Les principales propriétés sont

- Secret
- Authentification d'un message (ou d'une entité)
- Fraîcheur/anti-rejeu
- Accord non répudiable
- Équité
- Anonymat

Applications des protocoles crypto

Quelques applications :

Applications des protocoles crypto

Quelques applications :

- Communication secrète (Ex : SSL de https, SSH, GSM)

Applications des protocoles crypto

Quelques applications :

- Communication secrète (Ex : SSL de https, SSH, GSM)
- Authentification d'agents (Ex : `login`, CAS/SSO, GSM)

Applications des protocoles crypto

Quelques applications :

- Communication secrète (Ex : SSL de https, SSH, GSM)
- Authentification d'agents (Ex : `login`, CAS/SSO, GSM)
- Signature de contrats électroniques

Applications des protocoles crypto

Quelques applications :

- Communication secrète (Ex : SSL de https, SSH, GSM)
- Authentification d'agents (Ex : `login`, CAS/SSO, GSM)
- Signature de contrats électroniques
- Paiement bancaire par carte à puce

Applications des protocoles crypto

Quelques applications :

- Communication secrète (Ex : SSL de https, SSH, GSM)
- Authentification d'agents (Ex : `login`, CAS/SSO, GSM)
- Signature de contrats électroniques
- Paiement bancaire par carte à puce
- Paiement bancaire en ligne (transactions électroniques)

Applications des protocoles crypto

Quelques applications :

- Communication secrète (Ex : SSL de https, SSH, GSM)
- Authentification d'agents (Ex : `login`, CAS/SSO, GSM)
- Signature de contrats électroniques
- Paiement bancaire par carte à puce
- Paiement bancaire en ligne (transactions électroniques)
- Paiement bancaire hors ligne (monnaie électronique)

Applications des protocoles crypto

Quelques applications :

- Communication secrète (Ex : SSL de https, SSH, GSM)
- Authentification d'agents (Ex : `login`, CAS/SSO, GSM)
- Signature de contrats électroniques
- Paiement bancaire par carte à puce
- Paiement bancaire en ligne (transactions électroniques)
- Paiement bancaire hors ligne (monnaie électronique)
- Chaînes de télévision payantes
- Vote électronique
- ...

Définitions

Définitions

Définition *(Secret)* Un protocole assure le *secret* d'une donnée s si un intrus ne peut pas *déduire* s.

(s'il peut choisir s ça n'est pas une attaque sur le secret)

Définitions

Définition *(Secret)* Un protocole assure le *secret* d'une donnée s si un intrus ne peut pas *déduire* s.

(s'il peut choisir s ça n'est pas une attaque sur le secret)

Définition *(Authentification de message)* Un protocole permet à un agent A d'*authentifier un message* m si A peut connaître de façon sûre l'émetteur de m.

Définitions

Définition *(Secret)* Un protocole assure le *secret* d'une donnée s si un intrus ne peut pas *déduire* s.

(s'il peut choisir s ça n'est pas une attaque sur le secret)

Définition *(Authentification de message)* Un protocole permet à un agent A d'*authentifier un message m* si A peut connaître de façon sûre l'émetteur de m.

Définition *(Authentification d'entité)* Un protocole permet à un agent A d'*authentifier un agent B* si à la fin d'une session réussie, A a la garantie qu'il a bien réalisé le protocole avec B.

Définitions

Définition *(Secret)* Un protocole assure le *secret* d'une donnée s si un intrus ne peut pas *déduire* s.

(s'il peut choisir s ça n'est pas une attaque sur le secret)

Définition *(Authentification de message)* Un protocole permet à un agent A d'*authentifier un message m* si A peut connaître de façon sûre l'émetteur de m.

Définition *(Authentification d'entité)* Un protocole permet à un agent A d'*authentifier un agent B* si à la fin d'une session réussie, A a la garantie qu'il a bien réalisé le protocole avec B.

Définition *(Fraîcheur)* Pendant une session de protocole, une donnée est *fraîche* si l'on peut garantir qu'elle a été émise spécifiquement pour cette session par un des acteurs.

Principes de base des protocoles

Envoi d'un secret de B à A

Principes de base des protocoles

Envoi d'un secret de B à A

- Premier essai

  1. $B \hookrightarrow A : \{s\}_{K_A}$

  si la clé K_A et $\{\}_{K_A}$ sont robustes, seul A peut lire s.

Principes de base des protocoles

Envoi d'un secret de B à A

- Premier essai

  1. $B \hookrightarrow A : \{s\}_{K_A}$

  si la clé K_A et $\{\}_{K_A}$ sont robustes, seul A peut lire s.

  Mais de qui vient le secret s ?

Principes de base des protocoles

Envoi d'un secret de B à A

- Premier essai

  1. $B \hookrightarrow A : \{s\}_{K_A}$

  si la clé K_A et $\{\}_{K_A}$ sont robustes, seul A peut lire s.

  Mais de qui vient le secret s ?

- Deuxième essai (avec tentative d'authentification)
  1. $B \hookrightarrow A : \{B, s\}_{K_A}$

Principes de base des protocoles

Envoi d'un secret de B à A

- Premier essai

  1. $B \hookrightarrow A : \{s\}_{K_A}$

  si la clé K_A et $\{\}_{K_A}$ sont robustes, seul A peut lire s.

  Mais de qui vient le secret s ?

- Deuxième essai (avec tentative d'authentification)

  1. $B \hookrightarrow A : \{B, s\}_{K_A}$

  Facile à détourner par un intrus :

  1. $I(B) \hookrightarrow A : \{B, s_I\}_{K_A}$

  s_I est connu de I et accepté par A comme venant de B !

Principes de base des protocoles

Envoi d'un secret de B à A

Troisième essai (authentification par signature)

Principes de base des protocoles

Envoi d'un secret de B à A

Troisième essai (authentification par signature)

1. $B \hookrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}K_A$

Principes de base des protocoles

Envoi d'un secret de B à A

Troisième essai (authentification par signature)

1. $B \hookrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A}$

Rmq : solution avec des clés symétriques (canal chiffré)

1. $B \hookrightarrow A : \{B, s\}_{K_{AB}}$ si K_{AB} partagée par A et B

Principes de base des protocoles

Envoi d'un secret de B à A

Troisième essai (authentification par signature)

1. $B \hookrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A}$

Rmq : solution avec des clés symétriques (canal chiffré)

1. $B \hookrightarrow A : \{B, s\}_{K_{AB}}$ si K_{AB} partagée par A et B

Propriétés attendues :

- s secret
- A authentifie s comme étant émis par B

Principes de base des protocoles

Envoi d'un secret de B à A

Troisième essai (authentification par signature)

1. $B \hookrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A}$

Rmq : solution avec des clés symétriques (canal chiffré)

1. $B \hookrightarrow A : \{B, s\}_{K_{AB}}$ si K_{AB} partagée par A et B

Propriétés attendues :

- s secret
- A authentifie s comme étant émis par B

Mais I peut faire accepter de nouveau s (par rejeu) :

Clés asymétriques	Clés symétriques
1. $B \hookrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A}$	1. $B \hookrightarrow A : \{B, s\}_{K_{AB}}$
2. $I(B) \hookrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A}$	2. $I(B) \hookrightarrow A : \{B, s\}_{K_{AB}}$

Principes de base des protocoles

Envoi d'un secret de B à A

Quatrième essai (secret, authentification et fraîcheur) :

Principes de base des protocoles

Envoi d'un secret de B à A

Quatrième essai (secret, authentification et fraîcheur) :

On utilise un nombre aléatoire frais : un *nonce* N_A

Principes de base des protocoles

Envoi d'un secret de B à A

Quatrième essai (secret, authentification et fraîcheur) :

On utilise un nombre aléatoire frais : un *nonce* N_A

1. $A \hookrightarrow B : \{A, B, N_A\}_{K_B}$ "Challenge" de A pour B

Principes de base des protocoles

Envoi d'un secret de B à A

Quatrième essai (secret, authentification et fraîcheur) :

On utilise un nombre aléatoire frais : un *nonce* N_A

1. $A \hookrightarrow B : \{A, B, N_A\}_{K_B}$ "Challenge" de A pour B
2. $B \hookrightarrow A : \{A, B, N_A, s\}_{K_A}$ Réponse de B au challenge

Principes de base des protocoles

Envoi d'un secret de B à A

Quatrième essai (secret, authentification et fraîcheur) :

On utilise un nombre aléatoire frais : un *nonce* N_A

1. $A \hookrightarrow B : \{A, B, N_A\}_{K_B}$ "Challenge" de A pour B
2. $B \hookrightarrow A : \{A, B, N_A, s\}_{K_A}$ Réponse de B au challenge

Propriétés attendues :

Principes de base des protocoles

Envoi d'un secret de B à A

Quatrième essai (secret, authentification et fraîcheur) :

On utilise un nombre aléatoire frais : un *nonce* N_A

1. $A \hookrightarrow B : \{A, B, N_A\}_{K_B}$ "Challenge" de A pour B
2. $B \hookrightarrow A : \{A, B, N_A, s\}_{K_A}$ Réponse de B au challenge

Propriétés attendues :

- s secret

Principes de base des protocoles

Envoi d'un secret de B à A

Quatrième essai (secret, authentification et fraîcheur) :

On utilise un nombre aléatoire frais : un *nonce* N_A

1. $A \hookrightarrow B : \{A, B, N_A\}_{K_B}$ "Challenge" de A pour B
2. $B \hookrightarrow A : \{A, B, N_A, s\}_{K_A}$ Réponse de B au challenge

Propriétés attendues :

- s secret
- N_A frais et donc $\{A, B, N_A, s\}_{K_A}$ frais

Principes de base des protocoles

Envoi d'un secret de B à A

Quatrième essai (secret, authentification et fraîcheur) :

On utilise un nombre aléatoire frais : un *nonce* N_A

1. $A \hookrightarrow B : \{A, B, N_A\}_{K_B}$ "Challenge" de A pour B
2. $B \hookrightarrow A : \{A, B, N_A, s\}_{K_A}$ Réponse de B au challenge

Propriétés attendues :

- s secret
- N_A frais et donc $\{A, B, N_A, s\}_{K_A}$ frais
- A authentifie s comme étant émis par B

Principes de base des protocoles

Envoi d'un secret de B à A

Quatrième essai (secret, authentification et fraîcheur) :

On utilise un nombre aléatoire frais : un *nonce* N_A

1. $A \hookrightarrow B : \{A, B, N_A\}_{K_B}$ "Challenge" de A pour B
2. $B \hookrightarrow A : \{A, B, N_A, s\}_{K_A}$ Réponse de B au challenge

Propriétés attendues :

- s secret
- N_A frais et donc $\{A, B, N_A, s\}_{K_A}$ frais
- A authentifie s comme étant émis par B
- A authentifie B lors de la session.

Principes de base des protocoles

Envoi d'un secret de B à A

Solution similaire avec des clés symétriques :

1. $A \hookrightarrow B : \{A, B, N_A\}_{K_{AB}}$ "Challenge" de A pour B
2. $B \hookrightarrow A : \{A, B, N_A, s\}_{K_{AB}}$ Réponse de B au challenge

Principes de base des protocoles

Envoi d'un secret de B à A

Solution similaire avec des clés symétriques :

1. $A \hookrightarrow B : \{A, B, N_A\}_{K_{AB}}$ "Challenge" de A pour B
2. $B \hookrightarrow A : \{A, B, N_A, s\}_{K_{AB}}$ Réponse de B au challenge

Qu'il s'agisse de clés symétriques (K_{AB}) ou asymétriques (K_A, K_B) :

un échange de clés préalable est nécessaire !

Protocoles d'échange de clés

Echange d'une clé symétrique avec un tiers de confiance S

Protocoles d'échange de clés

Echange d'une clé symétrique avec un tiers de confiance S

(Extrait de Needham-Schroeder Symmetric Key Protocol)

1. $A \hookrightarrow S : A, B, N_A$

Protocoles d'échange de clés

Echange d'une clé symétrique avec un tiers de confiance S

(Extrait de Needham-Schroeder Symmetric Key Protocol)

1. $A \hookrightarrow S : A, B, N_A$
2. $S \hookrightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$

Protocoles d'échange de clés

Echange d'une clé symétrique avec un tiers de confiance S

(Extrait de Needham-Schroeder Symmetric Key Protocol)

1. $A \hookrightarrow S : A, B, N_A$
2. $S \hookrightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \hookrightarrow B : \{K_{AB}, A\}_{K_{BS}}$

Protocoles d'échange de clés

Echange d'une clé symétrique avec un tiers de confiance S

(Extrait de Needham-Schroeder Symmetric Key Protocol)

1. $A \hookrightarrow S : A, B, N_A$
2. $S \hookrightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \hookrightarrow B : \{K_{AB}, A\}_{K_{BS}}$

Propriétés attendues :

- secret de K_{AB}, partagé entre S, A et B

Protocoles d'échange de clés

Echange d'une clé symétrique avec un tiers de confiance S

(Extrait de Needham-Schroeder Symmetric Key Protocol)

1. $A \hookrightarrow S : A, B, N_A$
2. $S \hookrightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \hookrightarrow B : \{K_{AB}, A\}_{K_{BS}}$

Propriétés attendues :

- secret de K_{AB}, partagé entre S, A et B
- fraîcheur de K_{AB} (pour A mais pas pour B)

Protocoles d'échange de clés

Echange d'une clé symétrique avec un tiers de confiance S

(Extrait de Needham-Schroeder Symmetric Key Protocol)

1. $A \hookrightarrow S : A, B, N_A$
2. $S \hookrightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \hookrightarrow B : \{K_{AB}, A\}_{K_{BS}}$

Propriétés attendues :

- secret de K_{AB}, partagé entre S, A et B
- fraîcheur de K_{AB} (pour A mais pas pour B)
- authentification des messages émis par S pour A et B

Protocoles d'échange de clés

Echange d'une clé symétrique avec un tiers de confiance S

(Extrait de Needham-Schroeder Symmetric Key Protocol)

1. $A \hookrightarrow S : A, B, N_A$
2. $S \hookrightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \hookrightarrow B : \{K_{AB}, A\}_{K_{BS}}$

Propriétés attendues :

- secret de K_{AB}, partagé entre S, A et B
- fraîcheur de K_{AB} (pour A mais pas pour B)
- authentification des messages émis par S pour A et B
- A authentifie S

Protocoles d'échange de clés

Echange de clés assymmétriques avec un tiers de confiance S

(Extrait de Needham-Schroeder Public Key Protocol)

Protocoles d'échange de clés

Echange de clés assymmétriques avec un tiers de confiance S

(Extrait de Needham-Schroeder Public Key Protocol)

1. $A \hookrightarrow S : A, B$

Protocoles d'échange de clés

Echange de clés assymmétriques avec un tiers de confiance S

(Extrait de Needham-Schroeder Public Key Protocol)

1. $A \hookrightarrow S : A, B$
2. $S \hookrightarrow A : \{K_B, B\}_{K_S^{-1}}$

Protocoles d'échange de clés

Echange de clés assymmétriques avec un tiers de confiance S

(Extrait de Needham-Schroeder Public Key Protocol)

1. $A \hookrightarrow S : A, B$
2. $S \hookrightarrow A : \{K_B, B\}_{K_S^{-1}}$
3. $S \hookrightarrow B : \{K_A, A\}_{K_S^{-1}}$

Protocoles d'échange de clés

Echange de clés assymmétriques avec un tiers de confiance S

(Extrait de Needham-Schroeder Public Key Protocol)

1. $A \hookrightarrow S : A, B$
2. $S \hookrightarrow A : \{K_B, B\}_{K_S^{-1}}$
3. $S \hookrightarrow B : \{K_A, A\}_{K_S^{-1}}$

Propriétés attendues :

- authentification des clés publiques K_A et K_B

Protocoles d'échange de clés

Echange de clés assymmétriques avec un tiers de confiance S

(Extrait de Needham-Schroeder Public Key Protocol)

1. $A \hookrightarrow S : A, B$
2. $S \hookrightarrow A : \{K_B, B\}_{K_S^{-1}}$
3. $S \hookrightarrow B : \{K_A, A\}_{K_S^{-1}}$

Propriétés attendues :

- authentification des clés publiques K_A et K_B

Mais

- pas de secret (pas nécessaire)

Protocoles d'échange de clés

Echange de clés assymmétriques avec un tiers de confiance S

(Extrait de Needham-Schroeder Public Key Protocol)

1. $A \hookrightarrow S : A, B$
2. $S \hookrightarrow A : \{K_B, B\}_{K_S^{-1}}$
3. $S \hookrightarrow B : \{K_A, A\}_{K_S^{-1}}$

Propriétés attendues :

- authentification des clés publiques K_A et K_B

Mais

- pas de secret (pas nécessaire)
- pas de fraîcheur (pas nécessaire (?))

Protocoles d'échange de clés

Négociation d'une clé symétrique sans tiers de confiance

Protocoles d'échange de clés

Négociation d'une clé symétrique sans tiers de confiance

Protocole Diffie-Hellman : utilisé dans SSH mode dégradé

Protocoles d'échange de clés

Négociation d'une clé symétrique sans tiers de confiance

Protocole Diffie-Hellman : utilisé dans SSH mode dégradé

Entiers p et g publics et g générateur de Z_p^*

Protocoles d'échange de clés

Négociation d'une clé symétrique sans tiers de confiance

Protocole Diffie-Hellman : utilisé dans SSH mode dégradé

Entiers p et g publics et g générateur de Z_p^*

A choisit un secret $x \in Z$ et B choisit un secret $y \in Z$

Protocoles d'échange de clés

Négociation d'une clé symétrique sans tiers de confiance

Protocole Diffie-Hellman : utilisé dans SSH mode dégradé

Entiers p et g publics et g générateur de Z_p^*

A choisit un secret $x \in Z$ et B choisit un secret $y \in Z$

1. $A \hookrightarrow B : X = g^x \ mod \ p$
2. $B \hookrightarrow A : Y = g^y \ mod \ p$

Protocoles d'échange de clés

Négociation d'une clé symétrique sans tiers de confiance

Protocole Diffie-Hellman : utilisé dans SSH mode dégradé

Entiers p et g publics et g générateur de Z_p^*

A choisit un secret $x \in Z$ et B choisit un secret $y \in Z$

1. $A \hookrightarrow B : X = g^x \ mod \ p$
2. $B \hookrightarrow A : Y = g^y \ mod \ p$

A construit $Y^x \ mod \ p$ et B construit $X^y \ mod \ p$

Protocoles d'échange de clés

Négociation d'une clé symétrique sans tiers de confiance

Protocole Diffie-Hellman : utilisé dans SSH mode dégradé

Entiers p et g publics et g générateur de Z_p^*

A choisit un secret $x \in Z$ et B choisit un secret $y \in Z$

1. $A \hookrightarrow B : X = g^x \ mod \ p$
2. $B \hookrightarrow A : Y = g^y \ mod \ p$

A construit $Y^x \ mod \ p$ et B construit $X^y \ mod \ p$
et l'on a $K_{AB} = X^y \ mod \ p = Y^x \ mod \ p$

Protocoles d'échange de clés

Négociation d'une clé symétrique sans tiers de confiance

Protocole Diffie-Hellman : utilisé dans SSH mode dégradé

Entiers p et g publics et g générateur de Z_p^*

A choisit un secret $x \in Z$ et B choisit un secret $y \in Z$

1. $A \hookrightarrow B : X = g^x \ mod \ p$
2. $B \hookrightarrow A : Y = g^y \ mod \ p$

A construit $Y^x \ mod \ p$ et B construit $X^y \ mod \ p$
et l'on a $K_{AB} = X^y \ mod \ p = Y^x \ mod \ p$

car $(g^y)^x \ mod \ p = (g^x)^y \ mod \ p = g^{(xy)} \ mod \ p$

Protocoles d'échange de clés

Protocole Diffie-Hellman (suite)

1. $A \hookrightarrow B : X = g^x \ mod \ p$
2. $B \hookrightarrow A : Y = g^y \ mod \ p$

$K_{AB} = X^y \ mod \ p = Y^x \ mod \ p$

Protocoles d'échange de clés

Protocole Diffie-Hellman (suite)

1. $A \hookrightarrow B : X = g^x \ mod \ p$
2. $B \hookrightarrow A : Y = g^y \ mod \ p$

$K_{AB} = X^y \ mod \ p = Y^x \ mod \ p$

Propriétés attendues :

Protocoles d'échange de clés

Protocole Diffie-Hellman (suite)

1. $A \hookrightarrow B : X = g^x \ mod \ p$
2. $B \hookrightarrow A : Y = g^y \ mod \ p$

$K_{AB} = X^y \ mod \ p = Y^x \ mod \ p$

Propriétés attendues :

- secret de K_{AB}, partagé entre A et B

Protocoles d'échange de clés

Protocole Diffie-Hellman (suite)

1. $A \hookrightarrow B : X = g^x \ mod \ p$
2. $B \hookrightarrow A : Y = g^y \ mod \ p$

$K_{AB} = X^y \ mod \ p = Y^x \ mod \ p$

Propriétés attendues :

- secret de K_{AB}, partagé entre A et B
- fraîcheur de K_{AB}

Protocoles d'échange de clés

Protocole Diffie-Hellman (suite)

1. $A \hookrightarrow B : X = g^x \ mod \ p$
2. $B \hookrightarrow A : Y = g^y \ mod \ p$

$K_{AB} = X^y \ mod \ p = Y^x \ mod \ p$

Propriétés attendues :

- secret de K_{AB}, partagé entre A et B
- fraîcheur de K_{AB}

Mais

Protocoles d'échange de clés

Protocole Diffie-Hellman (suite)

1. $A \hookrightarrow B : X = g^x \ mod \ p$
2. $B \hookrightarrow A : Y = g^y \ mod \ p$

$K_{AB} = X^y \ mod \ p = Y^x \ mod \ p$

Propriétés attendues :

- secret de K_{AB}, partagé entre A et B
- fraîcheur de K_{AB}

Mais

- pas d'authentification des entités !

Protocoles d'échange de clés

Protocole Diffie-Hellman (suite)

1. $A \hookrightarrow B : X = g^x \ mod \ p$
2. $B \hookrightarrow A : Y = g^y \ mod \ p$

$K_{AB} = X^y \ mod \ p = Y^x \ mod \ p$

Propriétés attendues :

- secret de K_{AB}, partagé entre A et B
- fraîcheur de K_{AB}

Mais

- pas d'authentification des entités !
- $\Rightarrow$ pas de réelle authentification de K_{AB} par A et B !

Protocoles d'authentification : Login UNIX

Hypothèses :

- A connait P (son mot de passe)
- M connait A et $hash(P)$ (dans `/etc/passwd`)

Protocoles d'authentification : Login UNIX

Hypothèses :

- A connait P (son mot de passe)
- M connait A et $hash(P)$ (dans `/etc/passwd`)

1. $M \hookrightarrow A$: ``` ``login:'' ```
2. $A \hookrightarrow M$: A
3. $M \hookrightarrow A$: ``` ``passwd:'' ```
4. $A \hookrightarrow M$: P

Protocoles d'authentification : Login UNIX

Hypothèses :

- A connait P (son mot de passe)
- M connait A et $hash(P)$ (dans `/etc/passwd`)

1. $M \hookrightarrow A$: ``` ``login:'' ```
2. $A \hookrightarrow M$: A
3. $M \hookrightarrow A$: ``` ``passwd:'' ```
4. $A \hookrightarrow M$: P

Puis M calcule $hash(P)$ et compare dans `/etc/passwd`

Protocoles d'authentification : Login UNIX

Hypothèses :

- A connait P (son mot de passe)
- M connait A et $hash(P)$ (dans `/etc/passwd`)

1. $M \hookrightarrow A$: ``` ``login:'' ```
2. $A \hookrightarrow M$: A
3. $M \hookrightarrow A$: ``` ``passwd:'' ```
4. $A \hookrightarrow M$: P

Puis M calcule $hash(P)$ et compare dans `/etc/passwd`

$\Rightarrow$ suppose que la communication de A vers M est sûre

Protocoles d'authentification : Login UNIX

Hypothèses :

- A connait P (son mot de passe)
- M connait A et $hash(P)$ (dans `/etc/passwd`)

1. $M \hookrightarrow A$: ``` ``login:'' ```
2. $A \hookrightarrow M$: A
3. $M \hookrightarrow A$: ``` ``passwd:'' ```
4. $A \hookrightarrow M$: P

Puis M calcule $hash(P)$ et compare dans `/etc/passwd`

$\Rightarrow$ suppose que la communication de A vers M est sûre

Sinon, propriétés attendues (telnet) :

Protocoles d'authentification : Login UNIX

Hypothèses :

- A connait P (son mot de passe)
- M connait A et $hash(P)$ (dans `/etc/passwd`)

1. $M \hookrightarrow A$: ``` ``login:'' ```
2. $A \hookrightarrow M$: A
3. $M \hookrightarrow A$: ``` ``passwd:'' ```
4. $A \hookrightarrow M$: P

Puis M calcule $hash(P)$ et compare dans `/etc/passwd`

$\Rightarrow$ suppose que la communication de A vers M est sûre

Sinon, propriétés attendues (telnet) : aucune !

Protocoles d'authentification : Login SSH (1)

Hypothèses :

- A connait P (son mot de passe)
- M connait A et $hash(P)$ (dans `/etc/passwd`)

Protocoles d'authentification : Login SSH (1)

Hypothèses :

- A connait P (son mot de passe)
- M connait A et $hash(P)$ (dans `/etc/passwd`)

1. $M \hookrightarrow A : K_M$
2. $A \hookrightarrow M : \{K_{AM}\}_{K_M}$
3. $M \hookrightarrow A : \{\text{``}\texttt{login:}\text{''}\}_{K_{AM}}$
4. $A \hookrightarrow M : \{A\}_{K_{AM}}$
5. $M \hookrightarrow A : \{\text{``}\texttt{passwd:}\text{''}\}_{K_{AM}}$
6. $A \hookrightarrow M : \{P\}_{K_{AM}}$

Protocoles d'authentification : Login SSH (1)

Hypothèses :

- A connait P (son mot de passe)
- M connait A et $hash(P)$ (dans `/etc/passwd`)

1. $M \hookrightarrow A : K_M$
2. $A \hookrightarrow M : \{K_{AM}\}_{K_M}$
3. $M \hookrightarrow A : \{\text{``}\texttt{login:}\text{''}\}_{K_{AM}}$
4. $A \hookrightarrow M : \{A\}_{K_{AM}}$
5. $M \hookrightarrow A : \{\text{``}\texttt{passwd:}\text{''}\}_{K_{AM}}$
6. $A \hookrightarrow M : \{P\}_{K_{AM}}$

Propriétés attendues :

Protocoles d'authentification : Login SSH (1)

Hypothèses :

- A connait P (son mot de passe)
- M connait A et $hash(P)$ (dans `/etc/passwd`)

1. $M \hookrightarrow A : K_M$
2. $A \hookrightarrow M : \{K_{AM}\}_{K_M}$
3. $M \hookrightarrow A : \{\text{``}\texttt{login:}\text{''}\}_{K_{AM}}$
4. $A \hookrightarrow M : \{A\}_{K_{AM}}$
5. $M \hookrightarrow A : \{\text{``}\texttt{passwd:}\text{''}\}_{K_{AM}}$
6. $A \hookrightarrow M : \{P\}_{K_{AM}}$

Propriétés attendues :

- K_{AM}, A, P secrets partagés entre A et M
- M authentifie A, mais A n'authentifie pas M

Protocoles d'authentification : Login SSH (2)

Hypothèses :

- A a déposé (au préalable) sa clé publique K_A sur M

Protocoles d'authentification : Login SSH (2)

Hypothèses :

- A a déposé (au préalable) sa clé publique K_A sur M

1. $M \hookrightarrow A : K_M$
2. $A \hookrightarrow M : \{K_{AM}\}_{K_M}$
3. $M \hookrightarrow A : \{\{N_M\}_{K_A}\}_{K_{AM}}$
4. $A \hookrightarrow M : \{N_M\}_{K_{AM}}$

Protocoles d'authentification : Login SSH (2)

Hypothèses :

- A a déposé (au préalable) sa clé publique K_A sur M

1. $M \hookrightarrow A : K_M$
2. $A \hookrightarrow M : \{K_{AM}\}_{K_M}$
3. $M \hookrightarrow A : \{\{N_M\}_{K_A}\}_{K_{AM}}$
4. $A \hookrightarrow M : \{N_M\}_{K_{AM}}$

Propriétés attendues :

Protocoles d'authentification : Login SSH (2)

Hypothèses :

- A a déposé (au préalable) sa clé publique K_A sur M

1. $M \hookrightarrow A : K_M$
2. $A \hookrightarrow M : \{K_{AM}\}_{K_M}$
3. $M \hookrightarrow A : \{\{N_M\}_{K_A}\}_{K_{AM}}$
4. $A \hookrightarrow M : \{N_M\}_{K_{AM}}$

Propriétés attendues :

- K_{AM} secret partagé entre A et M
- M authentifie A, mais A n'authentifie pas M

Protocoles d'authentification : SSL/TLS (https)

Hypothèses :

- B(rowser) connait A et P (ou A les saisit dans B)

Protocoles d'authentification : SSL/TLS (https)

Hypothèses :

- B(rowser) connait A et P (ou A les saisit dans B)
- B connait K_S, où S est un tiers de confiance
  On parle aussi de "Certification Authority" (CA), *e.g.* VeriSign, ...

Protocoles d'authentification : SSL/TLS (https)

Hypothèses :

- B(rowser) connait A et P (ou A les saisit dans B)
- B connait K_S, où S est un tiers de confiance
  On parle aussi de "Certification Authority" (CA), *e.g.* VeriSign, ...
- V(endor) connait $\{V, K_V\}_{K_S^{-1}}$ "certificat" acheté à S par V
  et renouvelé tous les ans !

Protocoles d'authentification : SSL/TLS (https)

Hypothèses :

- B(rowser) connait A et P (ou A les saisit dans B)
- B connait K_S, où S est un tiers de confiance
  On parle aussi de "Certification Authority" (CA), *e.g.* VeriSign, ...
- V(endor) connait $\{V, K_V\}_{K_S^{-1}}$ "certificat" acheté à S par V
  et renouvelé tous les ans !
- V connait A et P

Protocoles d'authentification : SSL/TLS (https)

Hypothèses :

- B(rowser) connait A et P (ou A les saisit dans B)
- B connait K_S, où S est un tiers de confiance
  On parle aussi de "Certification Authority" (CA), *e.g.* VeriSign, ...
- V(endor) connait $\{V, K_V\}_{K_S^{-1}}$ "certificat" acheté à S par V
  et renouvelé tous les ans !
- V connait A et P

1. $V \hookrightarrow B : \{V, K_V\}_{K_S^{-1}}$
2. $B \hookrightarrow V : \{K_{BV}\}_{K_V}$
3. $V \hookrightarrow B : \{\text{``}\texttt{login:}\text{''}\}_{K_{BV}}$
4. $B \hookrightarrow V : \{A\}_{K_{BV}}$
5. $V \hookrightarrow B : \{\text{``}\texttt{passwd:}\text{''}\}_{K_{BV}}$
6. $B \hookrightarrow V : \{P\}_{K_{BV}}$

Protocoles d'authentification : SSL/TLS (https)

1. $V \hookrightarrow B : \{V, K_V\}_{K_S^{-1}}$
2. $B \hookrightarrow V : \{K_{BV}\}_{K_V}$
3. $V \hookrightarrow B : \{\text{``}\texttt{login:}\text{''}\}_{K_{BV}}$
4. $B \hookrightarrow V : \{A\}_{K_{BV}}$
5. $V \hookrightarrow B : \{\text{``}\texttt{passwd:}\text{''}\}_{K_{BV}}$
6. $B \hookrightarrow V : \{P\}_{K_{BV}}$

Protocoles d'authentification : SSL/TLS (https)

1. $V \hookrightarrow B : \{V, K_V\}_{K_S^{-1}}$
2. $B \hookrightarrow V : \{K_{BV}\}_{K_V}$
3. $V \hookrightarrow B : \{\text{``}\texttt{login:}\text{''}\}_{K_{BV}}$
4. $B \hookrightarrow V : \{A\}_{K_{BV}}$
5. $V \hookrightarrow B : \{\text{``}\texttt{passwd:}\text{''}\}_{K_{BV}}$
6. $B \hookrightarrow V : \{P\}_{K_{BV}}$

Propriétés attendues :

Protocoles d'authentification : SSL/TLS (https)

1. $V \hookrightarrow B : \{V, K_V\}_{K_S^{-1}}$
2. $B \hookrightarrow V : \{K_{BV}\}_{K_V}$
3. $V \hookrightarrow B : \{\text{``}\texttt{login:}\text{''}\}_{K_{BV}}$
4. $B \hookrightarrow V : \{A\}_{K_{BV}}$
5. $V \hookrightarrow B : \{\text{``}\texttt{passwd:}\text{''}\}_{K_{BV}}$
6. $B \hookrightarrow V : \{P\}_{K_{BV}}$

Propriétés attendues :

- K_{BV}, A, P secrets partagés entre B et V

Protocoles d'authentification : SSL/TLS (https)

1. $V \hookrightarrow B : \{V, K_V\}_{K_S^{-1}}$
2. $B \hookrightarrow V : \{K_{BV}\}_{K_V}$
3. $V \hookrightarrow B : \{\text{``}\texttt{login:}\text{''}\}_{K_{BV}}$
4. $B \hookrightarrow V : \{A\}_{K_{BV}}$
5. $V \hookrightarrow B : \{\text{``}\texttt{passwd:}\text{''}\}_{K_{BV}}$
6. $B \hookrightarrow V : \{P\}_{K_{BV}}$

Propriétés attendues :

- K_{BV}, A, P secrets partagés entre B et V
- B authentifie $\{V, K_V\}_{K_S^{-1}}$ comme construit par S

Protocoles d'authentification : SSL/TLS (https)

1. $V \hookrightarrow B : \{V, K_V\}_{K_S^{-1}}$
2. $B \hookrightarrow V : \{K_{BV}\}_{K_V}$
3. $V \hookrightarrow B : \{\text{``}\texttt{login:}\text{''}\}_{K_{BV}}$
4. $B \hookrightarrow V : \{A\}_{K_{BV}}$
5. $V \hookrightarrow B : \{\text{``}\texttt{passwd:}\text{''}\}_{K_{BV}}$
6. $B \hookrightarrow V : \{P\}_{K_{BV}}$

Propriétés attendues :

- K_{BV}, A, P secrets partagés entre B et V
- B authentifie $\{V, K_V\}_{K_S^{-1}}$ comme construit par S
- V authentifie A, B authentifie V (en faisant confiance à S)

Propriétés des protocoles crypto

Propriétés des protocoles crypto

Définition *(Accord non répudiable)* Un protocole établit un *accord non répudiable* entre deux agents si chaque agent peut fournir la preuve que l'autre a accepté les termes de l'accord.

Propriétés des protocoles crypto

Définition *(Accord non répudiable)* Un protocole établit un *accord non répudiable* entre deux agents si chaque agent peut fournir la preuve que l'autre a accepté les termes de l'accord.

Définition *(Équité)* Un protocole d'accord non répudiable entre deux agents A est B est *équitable* si aucun agent ne peut obtenir d'avantage sur l'autre : A n'obtient pas la preuve de l'accord de B avant que B n'ait une preuve de l'accord de A (et vice-versa).

Propriétés des protocoles crypto

Définition *(Accord non répudiable)* Un protocole établit un *accord non répudiable* entre deux agents si chaque agent peut fournir la preuve que l'autre a accepté les termes de l'accord.

Définition *(Équité)* Un protocole d'accord non répudiable entre deux agents A est B est *équitable* si aucun agent ne peut obtenir d'avantage sur l'autre : A n'obtient pas la preuve de l'accord de B avant que B n'ait une preuve de l'accord de A (et vice-versa).

Définition *(Anonymat)* Un protocole préserve l'*anonymat* d'un agent A s'il est impossible d'identifier A à partir des messages échangés.

Protocoles d'accord non-répudiable

Protocoles d'accord non-répudiable

Exemple rudimentaire :

1. $A \hookrightarrow B : \{A, B, \mathit{Contrat}\}_{K_A^{-1}}$

Protocoles d'accord non-répudiable

Exemple rudimentaire :

1. $A \hookrightarrow B : \{A, B, \mathit{Contrat}\}_{K_A^{-1}}$
2. $B \hookrightarrow A : \{A, B, \mathit{Contrat}\}_{K_B^{-1}}$

Protocoles d'accord non-répudiable

Exemple rudimentaire :

1. $A \hookrightarrow B : \{A, B, \mathit{Contrat}\}_{K_A^{-1}}$
2. $B \hookrightarrow A : \{A, B, \mathit{Contrat}\}_{K_B^{-1}}$

Propriétés attendues :

- accord non-répudiable entre A et B sur $\mathit{Contrat}$

Protocoles d'accord non-répudiable

Exemple rudimentaire :

1. $A \hookrightarrow B : \{A, B, \mathit{Contrat}\}_{K_A^{-1}}$
2. $B \hookrightarrow A : \{A, B, \mathit{Contrat}\}_{K_B^{-1}}$

Propriétés attendues :

- accord non-répudiable entre A et B sur $\mathit{Contrat}$

Mais

- non équitable car B obtient la signature avant A !

Protocoles d'accord non-répudiable

Exemple rudimentaire :

1. $A \hookrightarrow B : \{A, B, \mathit{Contrat}\}_{K_A^{-1}}$
2. $B \hookrightarrow A : \{A, B, \mathit{Contrat}\}_{K_B^{-1}}$

Propriétés attendues :

- accord non-répudiable entre A et B sur $\mathit{Contrat}$

Mais

- non équitable car B obtient la signature avant A !

Pour se rapprocher de l'équité, en pratique, ça sera plutôt :

1. $A \hookrightarrow B : \{A, B, \mathit{Cont}\}_{K_A^{-1}}$
2. $B \hookrightarrow A : \{A, B, \mathit{Cont}\}_{K_B^{-1}}$
3. $A \hookrightarrow B : \{A, B, \mathit{rat}\}_{K_A^{-1}}$
4. $B \hookrightarrow A : \{A, B, \mathit{rat}\}_{K_B^{-1}}$

Protocoles d'accord non-répudiable

Accusé réception équitable avec un tiers de confiance S
[Zhou-Gollmann 96]

Protocoles d'accord non-répudiable

Accusé réception équitable avec un tiers de confiance S
[Zhou-Gollmann 96]

1. $A \hookrightarrow B : \{propose, B, N_A, \{M\}_K\}_{K_A^{-1}}$

Protocoles d'accord non-répudiable

Accusé réception équitable avec un tiers de confiance S
[Zhou-Gollmann 96]

1. $A \hookrightarrow B : \{propose, B, N_A, \{M\}_K\}_{K_A^{-1}}$
2. $B \hookrightarrow A : \{ack, A, N_A, \{M\}_K\}_{K_B^{-1}}$

Protocoles d'accord non-répudiable

Accusé réception équitable avec un tiers de confiance S
[Zhou-Gollmann 96]

1. $A \hookrightarrow B : \{propose, B, N_A, \{M\}_K\}_{K_A^{-1}}$
2. $B \hookrightarrow A : \{ack, A, N_A, \{M\}_K\}_{K_B^{-1}}$
3. $A \hookrightarrow S : \{submit, B, N_A, K\}_{K_A^{-1}}$

Protocoles d'accord non-répudiable

Accusé réception équitable avec un tiers de confiance S
[Zhou-Gollmann 96]

1. $A \hookrightarrow B : \{propose, B, N_A, \{M\}_K\}_{K_A^{-1}}$
2. $B \hookrightarrow A : \{ack, A, N_A, \{M\}_K\}_{K_B^{-1}}$
3. $A \hookrightarrow S : \{submit, B, N_A, K\}_{K_A^{-1}}$
4. $S \hookrightarrow A, B : \{confirm, A, B, N_A, K\}_{K_S^{-1}}$

Protocoles d'accord non-répudiable

Accusé réception équitable avec un tiers de confiance S
[Zhou-Gollmann 96]

1. $A \hookrightarrow B : \{propose, B, N_A, \{M\}_K\}_{K_A^{-1}}$
2. $B \hookrightarrow A : \{ack, A, N_A, \{M\}_K\}_{K_B^{-1}}$
3. $A \hookrightarrow S : \{submit, B, N_A, K\}_{K_A^{-1}}$
4. $S \hookrightarrow A, B : \{confirm, A, B, N_A, K\}_{K_S^{-1}}$

Propriétés attendues :

- M secret pour tous (excepté A) jusqu'en 3.

Protocoles d'accord non-répudiable

Accusé réception équitable avec un tiers de confiance S
[Zhou-Gollmann 96]

1. $A \hookrightarrow B : \{propose, B, N_A, \{M\}_K\}_{K_A^{-1}}$
2. $B \hookrightarrow A : \{ack, A, N_A, \{M\}_K\}_{K_B^{-1}}$
3. $A \hookrightarrow S : \{submit, B, N_A, K\}_{K_A^{-1}}$
4. $S \hookrightarrow A, B : \{confirm, A, B, N_A, K\}_{K_S^{-1}}$

Propriétés attendues :

- M secret pour tous (excepté A) jusqu'en 3.
- Les 4 messages sont authentifiables (signatures K_X^{-1})

Protocoles d'accord non-répudiable

Accusé réception équitable avec un tiers de confiance S
[Zhou-Gollmann 96]

1. $A \hookrightarrow B : \{propose, B, N_A, \{M\}_K\}_{K_A^{-1}}$
2. $B \hookrightarrow A : \{ack, A, N_A, \{M\}_K\}_{K_B^{-1}}$
3. $A \hookrightarrow S : \{submit, B, N_A, K\}_{K_A^{-1}}$
4. $S \hookrightarrow A, B : \{confirm, A, B, N_A, K\}_{K_S^{-1}}$

Propriétés attendues :

- M secret pour tous (excepté A) jusqu'en 3.
- Les 4 messages sont authentifiables (signatures K_X^{-1})
- A et B ont en même temps (en 4.) la preuve que :
  - pour A : que le message a été reçu par B
  - pour B : que A a bien envoyé le message

Exemples de protocoles de paiement électronique

- Paiement sur internet
  - Vérification en ligne : SET/CSET)
  - Vérification hors ligne (et distribuée) : Bitcoin

Exemples de protocoles de paiement électronique

- Paiement sur internet
  - Vérification en ligne : SET/CSET)
  - Vérification hors ligne (et distribuée) : Bitcoin
- Paiement sur terminal protégé
  - Vérification en ligne : carte Bancaire
  - Vérification hors ligne : carte Bancaire à puce

SET/CSET

- Protocole proposé par Visa et MasterCard en 1997

SET/CSET

- Protocole proposé par Visa et MasterCard en 1997
- Spécification totalement ouverte (!)

SET/CSET

- Protocole proposé par Visa et MasterCard en 1997
- Spécification totalement ouverte (!)
- Paiement *en ligne* entre 3 agents connectés :
  - (C)lient
  - (M)archand
  - (B)anque (en fait un portail bancaire)

SET/CSET

- Protocole proposé par Visa et MasterCard en 1997
- Spécification totalement ouverte (!)
- Paiement *en ligne* entre 3 agents connectés :
  - (C)lient
  - (M)archand
  - (B)anque (en fait un portail bancaire)
- N_M et N_C nonces

SET/CSET

- Protocole proposé par Visa et MasterCard en 1997
- Spécification totalement ouverte (!)
- Paiement *en ligne* entre 3 agents connectés :
  - (C)lient
  - (M)archand
  - (B)anque (en fait un portail bancaire)
- N_M et N_C nonces
- $Purchamt$ = montant de la transaction

SET/CSET

- Protocole proposé par Visa et MasterCard en 1997
- Spécification totalement ouverte (!)
- Paiement *en ligne* entre 3 agents connectés :
  - (C)lient
  - (M)archand
  - (B)anque (en fait un portail bancaire)
- N_M et N_C nonces
- $Purchamt$ = montant de la transaction
- Od = "Order details" = détails de la commande

SET/CSET

- Protocole proposé par Visa et MasterCard en 1997
- Spécification totalement ouverte (!)
- Paiement *en ligne* entre 3 agents connectés :
  - (C)lient
  - (M)archand
  - (B)anque (en fait un portail bancaire)
- N_M et N_C nonces
- $Purchamt$ = montant de la transaction
- Od = "Order details" = détails de la commande
- Pd = "Payment details" = détails du règlement

SET/CSET

$$(PInitReq) \qquad C \hookrightarrow M \quad : \quad C, M$$

SET/CSET

$$(PInitReq) \qquad C \hookrightarrow M \;:\; C, M$$
$$(PInitRes) \qquad M \hookrightarrow C \;:\; N_M$$

SET/CSET

$$
\begin{array}{lcl}
(PInitReq) & C \hookrightarrow M & : \quad C, M \\
(PInitRes) & M \hookrightarrow C & : \quad N_M \\
(PReq) & C \hookrightarrow M & : \quad \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

SET/CSET

$$
\begin{array}{lcl}
(PInitReq) & C \hookrightarrow M & : \quad C, M \\
(PInitRes) & M \hookrightarrow C & : \quad N_M \\
(PReq) & C \hookrightarrow M & : \quad \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B} \\
(AuthReq) & M \hookrightarrow B & : \quad \{Trans\}_{K_C^{-1}}, \{Trans\}_{K_M^{-1}}, \{Pd\}_{K_B}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

SET/CSET

$$
\begin{array}{lcl}
(PInitReq) & C \hookrightarrow M & : \quad C, M \\
(PInitRes) & M \hookrightarrow C & : \quad N_M \\
(PReq) & C \hookrightarrow M & : \quad \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B} \\
(AuthReq) & M \hookrightarrow B & : \quad \{Trans\}_{K_C^{-1}}, \{Trans\}_{K_M^{-1}}, \{Pd\}_{K_B} \\
(AuthRes) & B \hookrightarrow M & : \quad \{Results, hash(Trans)\}_{K_B^{-1}}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

SET/CSET

$$
\begin{array}{lcl}
(PInitReq) & C \hookrightarrow M & : \quad C, M \\
(PInitRes) & M \hookrightarrow C & : \quad N_M \\
(PReq) & C \hookrightarrow M & : \quad \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B} \\
(AuthReq) & M \hookrightarrow B & : \quad \{Trans\}_{K_C^{-1}}, \{Trans\}_{K_M^{-1}}, \{Pd\}_{K_B} \\
(AuthRes) & B \hookrightarrow M & : \quad \{Results, hash(Trans)\}_{K_B^{-1}} \\
(PRes) & M \hookrightarrow C & : \quad \{Results, hash(Trans)\}_{K_B^{-1}}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

SET/CSET

$$
\begin{array}{lrcl}
(PInitReq) & C \hookrightarrow M & : & C, M \\
(PInitRes) & M \hookrightarrow C & : & N_M \\
(PReq) & C \hookrightarrow M & : & \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B} \\
(AuthReq) & M \hookrightarrow B & : & \{Trans\}_{K_C^{-1}}, \{Trans\}_{K_M^{-1}}, \{Pd\}_{K_B} \\
(AuthRes) & B \hookrightarrow M & : & \{Results, hash(Trans)\}_{K_B^{-1}} \\
(PRes) & M \hookrightarrow C & : & \{Results, hash(Trans)\}_{K_B^{-1}}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

Propriétés attendues :

- Od secret partagé entre C et M et inconnu à tout autre agent (y compris B)

SET/CSET

$$
\begin{array}{lcll}
(PInitReq) & C \hookrightarrow M & : & C, M \\
(PInitRes) & M \hookrightarrow C & : & N_M \\
(PReq) & C \hookrightarrow M & : & \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B} \\
(AuthReq) & M \hookrightarrow B & : & \{Trans\}_{K_C^{-1}}, \{Trans\}_{K_M^{-1}}, \{Pd\}_{K_B} \\
(AuthRes) & B \hookrightarrow M & : & \{Results, hash(Trans)\}_{K_B^{-1}} \\
(PRes) & M \hookrightarrow C & : & \{Results, hash(Trans)\}_{K_B^{-1}}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

Propriétés attendues :

- Od secret partagé entre C et M et inconnu à tout autre agent (y compris B)
- Pd secret partagé entre C et B et inconnu à tout autre agent (y compris M)

SET/CSET

$$
\begin{array}{lcl}
(PInitReq) & C \hookrightarrow M & : \; C, M \\
(PInitRes) & M \hookrightarrow C & : \; N_M \\
(PReq) & C \hookrightarrow M & : \; \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B} \\
(AuthReq) & M \hookrightarrow B & : \; \{Trans\}_{K_C^{-1}}, \{Trans\}_{K_M^{-1}}, \{Pd\}_{K_B} \\
(AuthRes) & B \hookrightarrow M & : \; \{Results, hash(Trans)\}_{K_B^{-1}} \\
(PRes) & M \hookrightarrow C & : \; \{Results, hash(Trans)\}_{K_B^{-1}}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

- Authentification mutuelle des entités C et M grâce à $\{\ldots N_M \ldots\}_{K_C^{-1}}$ et $\{\ldots N_C \ldots\}_{K_M^{-1}}$

SET/CSET

$$
\begin{array}{lcl}
(PInitReq) & C \hookrightarrow M & : \quad C, M \\
(PInitRes) & M \hookrightarrow C & : \quad N_M \\
(PReq) & C \hookrightarrow M & : \quad \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B} \\
(AuthReq) & M \hookrightarrow B & : \quad \{Trans\}_{K_C^{-1}}, \{Trans\}_{K_M^{-1}}, \{Pd\}_{K_B} \\
(AuthRes) & B \hookrightarrow M & : \quad \{Results, hash(Trans)\}_{K_B^{-1}} \\
(PRes) & M \hookrightarrow C & : \quad \{Results, hash(Trans)\}_{K_B^{-1}}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

- Authentification mutuelle des entités C et M grâce à $\{\ldots N_M \ldots\}_{K_C^{-1}}$ et $\{\ldots N_C \ldots\}_{K_M^{-1}}$
- Authentification de l'entité B par C et M grâce à $\{\ldots hash(\ldots N_C, N_M \ldots) \ldots\}_{K_B^{-1}}$

SET/CSET

$$
\begin{array}{lcll}
(PInitReq) & C \hookrightarrow M & : & C, M \\
(PInitRes) & M \hookrightarrow C & : & N_M \\
(PReq) & C \hookrightarrow M & : & \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B} \\
(AuthReq) & M \hookrightarrow B & : & \{Trans\}_{K_C^{-1}}, \{Trans\}_{K_M^{-1}}, \{Pd\}_{K_B} \\
(AuthRes) & B \hookrightarrow M & : & \{Results, hash(Trans)\}_{K_B^{-1}} \\
(PRes) & M \hookrightarrow C & : & \{Results, hash(Trans)\}_{K_B^{-1}}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

- Authentification par tous les agents des 3 messages $\{Trans\}_{K_C^{-1}}$, $\{Trans\}_{K_M^{-1}}$ et $\{Results, hash(Trans)\}_{K_B^{-1}}$

SET/CSET

$$
\begin{array}{lcl}
(PInitReq) & C \hookrightarrow M & : \quad C, M \\
(PInitRes) & M \hookrightarrow C & : \quad N_M \\
(PReq) & C \hookrightarrow M & : \quad \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B} \\
(AuthReq) & M \hookrightarrow B & : \quad \{Trans\}_{K_C^{-1}}, \{Trans\}_{K_M^{-1}}, \{Pd\}_{K_B} \\
(AuthRes) & B \hookrightarrow M & : \quad \{Results, hash(Trans)\}_{K_B^{-1}} \\
(PRes) & M \hookrightarrow C & : \quad \{Results, hash(Trans)\}_{K_B^{-1}}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

- Authentification par tous les agents des 3 messages $\{Trans\}_{K_C^{-1}}$, $\{Trans\}_{K_M^{-1}}$ et $\{Results, hash(Trans)\}_{K_B^{-1}}$
- Accord tripartite C, M, B non répudiable grâce à $\{Trans\}_{K_C^{-1}}$, $\{Trans\}_{K_M^{-1}}$ et $\{Results, hash(Trans)\}_{K_B^{-1}}$

SET/CSET

$$
\begin{array}{lcll}
(PInitReq) & C \hookrightarrow M & : & C, M \\
(PInitRes) & M \hookrightarrow C & : & N_M \\
(PReq) & C \hookrightarrow M & : & \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B} \\
(AuthReq) & M \hookrightarrow B & : & \{Trans\}_{K_C^{-1}}, \{Trans\}_{K_M^{-1}}, \{Pd\}_{K_B} \\
(AuthRes) & B \hookrightarrow M & : & \{Results, hash(Trans)\}_{K_B^{-1}} \\
(PRes) & M \hookrightarrow C & : & \{Results, hash(Trans)\}_{K_B^{-1}}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

- Authentification par tous les agents des 3 messages $\{Trans\}_{K_C^{-1}}$, $\{Trans\}_{K_M^{-1}}$ et $\{Results, hash(Trans)\}_{K_B^{-1}}$
- Accord tripartite C, M, B non répudiable grâce à $\{Trans\}_{K_C^{-1}}$, $\{Trans\}_{K_M^{-1}}$ et $\{Results, hash(Trans)\}_{K_B^{-1}}$
- Fraîcheur transaction vérifiée par B avec N_C et N_M

SET/CSET

$$
\begin{array}{lcl}
(PInitReq) & C \hookrightarrow M & : \quad C, M \\
(PInitRes) & M \hookrightarrow C & : \quad N_M \\
(PReq) & C \hookrightarrow M & : \quad \{Trans\}_{K_C^{-1}}, \{Od\}_{K_M}, \{Pd\}_{K_B} \\
(AuthReq) & M \hookrightarrow B & : \quad \{Trans\}_{K_C^{-1}}, \{\{Trans\}_{K_M^{-1}}, \{Pd\}_{K_B} \\
(AuthRes) & B \hookrightarrow M & : \quad \{Results, hash(Trans)\}_{K_B^{-1}} \\
(PRes) & M \hookrightarrow C & : \quad \{Results, hash(Trans)\}_{K_B^{-1}}
\end{array}
$$

$$Trans = (C,\ M,\ (N_C, N_M),\ Purchamt,\ hash(Od),\ hash(Pd))$$

Mais

- pas d'anonymat : B sait que C a acheté chez M !
- pas d'équité : C signe avant M

Monnaie électronique : Le protocole Bitcoin

- Protocole public proposé par ≪ Satoshi Nakamoto ≫ [1] en 2008

1. A l'heure actuelle, on ne sait pas s'il s'agit d'une personne réelle ou d'un groupe.

Monnaie électronique : Le protocole Bitcoin

- Protocole public proposé par ≪ Satoshi Nakamoto ≫ [1] en 2008
- Le protocole est sans tiers de confiance et décentralisé

1. A l'heure actuelle, on ne sait pas s'il s'agit d'une personne réelle ou d'un groupe.

Monnaie électronique : Le protocole Bitcoin

- Protocole public proposé par ≪ Satoshi Nakamoto ≫ [1] en 2008
- Le protocole est sans tiers de confiance et décentralisé :

  Les transactions sont validées par un réseau pair-à-pair d'utilisateurs

1. A l'heure actuelle, on ne sait pas s'il s'agit d'une personne réelle ou d'un groupe.

Monnaie électronique : Le protocole Bitcoin

- Protocole public proposé par « Satoshi Nakamoto »[1] en 2008
- Le protocole est sans tiers de confiance et décentralisé :
  Les transactions sont validées par un réseau pair-à-pair d'utilisateurs
- Transaction= transfert de Bitcoins (฿) entre comptes

1. A l'heure actuelle, on ne sait pas s'il s'agit d'une personne réelle ou d'un groupe.

Monnaie électronique : Le protocole Bitcoin

- Protocole public proposé par « Satoshi Nakamoto »[1] en 2008
- Le protocole est sans tiers de confiance et décentralisé :

  Les transactions sont validées par un réseau pair-à-pair d'utilisateurs
- Transaction= transfert de Bitcoins (฿) entre comptes
  - Un compte est représenté par un couple (clé publique, clé privée)

1. A l'heure actuelle, on ne sait pas s'il s'agit d'une personne réelle ou d'un groupe.

Monnaie électronique : Le protocole Bitcoin

- Protocole public proposé par « Satoshi Nakamoto »[1] en 2008
- Le protocole est sans tiers de confiance et décentralisé :

  Les transactions sont validées par un réseau pair-à-pair d'utilisateurs
- Transaction= transfert de Bitcoins (฿) entre comptes
  - Un compte est représenté par un couple (clé publique, clé privée)
  - Le numéro de compte est la clé publique

1. A l'heure actuelle, on ne sait pas s'il s'agit d'une personne réelle ou d'un groupe.

Monnaie électronique : Le protocole Bitcoin

- Protocole public proposé par « Satoshi Nakamoto »[1] en 2008
- Le protocole est sans tiers de confiance et décentralisé :

  Les transactions sont validées par un réseau pair-à-pair d'utilisateurs
- Transaction= transfert de Bitcoins (฿) entre comptes
  - Un compte est représenté par un couple (clé publique, clé privée)
  - Le numéro de compte est la clé publique
  - Seul le titulaire du compte connait la clé privée du compte

1. A l'heure actuelle, on ne sait pas s'il s'agit d'une personne réelle ou d'un groupe.

Monnaie électronique : Le protocole Bitcoin

- Protocole public proposé par « Satoshi Nakamoto »[1] en 2008
- Le protocole est sans tiers de confiance et décentralisé :

  Les transactions sont validées par un réseau pair-à-pair d'utilisateurs
- Transaction= transfert de Bitcoins (฿) entre comptes
  - Un compte est représenté par un couple (clé publique, clé privée)
  - Le numéro de compte est la clé publique
  - Seul le titulaire du compte connait la clé privée du compte
- Transaction *hors ligne* entre 2 *comptes* de numéros K_A et K_B
  - On suppose que Alice détient le compte K_A (elle connait K_A^{-1})
  - On suppose que Bob détient le compte K_B (il connait K_B^{-1})

1. A l'heure actuelle, on ne sait pas s'il s'agit d'une personne réelle ou d'un groupe.

Monnaie électronique : Le protocole Bitcoin

- Protocole public proposé par « Satoshi Nakamoto »[1] en 2008
- Le protocole est sans tiers de confiance et décentralisé :

  Les transactions sont validées par un réseau pair-à-pair d'utilisateurs
- Transaction= transfert de Bitcoins (฿) entre comptes
  - Un compte est représenté par un couple (clé publique, clé privée)
  - Le numéro de compte est la clé publique
  - Seul le titulaire du compte connait la clé privée du compte
- Transaction *hors ligne* entre 2 *comptes* de numéros K_A et K_B
  - On suppose que Alice détient le compte K_A (elle connait K_A^{-1})
  - On suppose que Bob détient le compte K_B (il connait K_B^{-1})
  - Pour donner 20฿ à Bob, Alice envoie $\{20, i, K_B\}_{K_A^{-1}}$ sur le réseau P2P où i sera un « numéro de transaction » (pour simplifier)

1. A l'heure actuelle, on ne sait pas s'il s'agit d'une personne réelle ou d'un groupe.

Protocole Bitcoin (II)

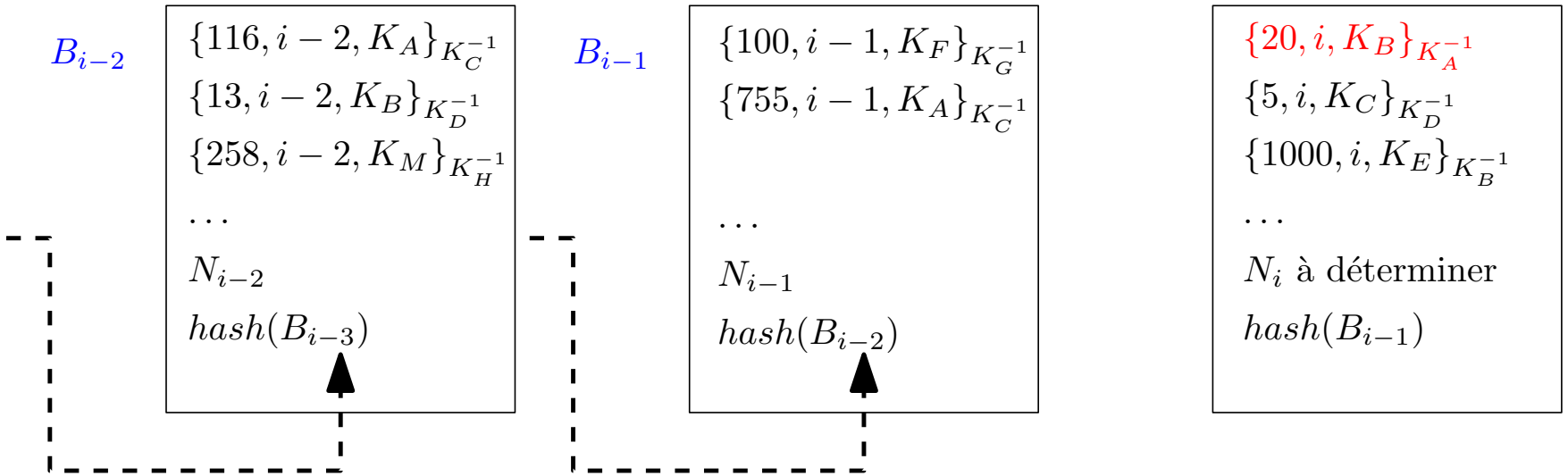
- Toutes les 10 minutes, chaque noeud du P2P déclenche la vérification

Protocole Bitcoin (II)

- Toutes les 10 minutes, chaque noeud du P2P déclenche la vérification
- Les nouvelles transactions sont vérifiées par rapport à la BlockChain
  BlockChain=historique des transactions depuis le démarrage (2009)

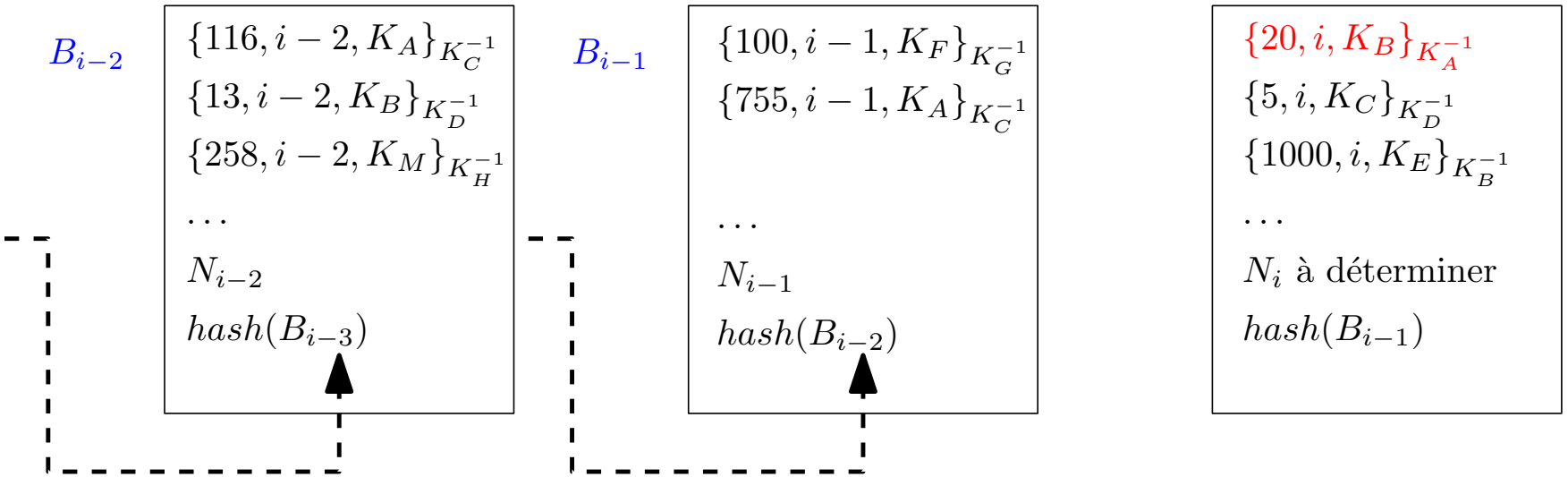
Protocole Bitcoin (II)

- Toutes les 10 minutes, chaque noeud du P2P déclenche la vérification
- Les nouvelles transactions sont vérifiées par rapport à la BlockChain
  BlockChain=historique des transactions depuis le démarrage (2009)



Protocole Bitcoin (II)

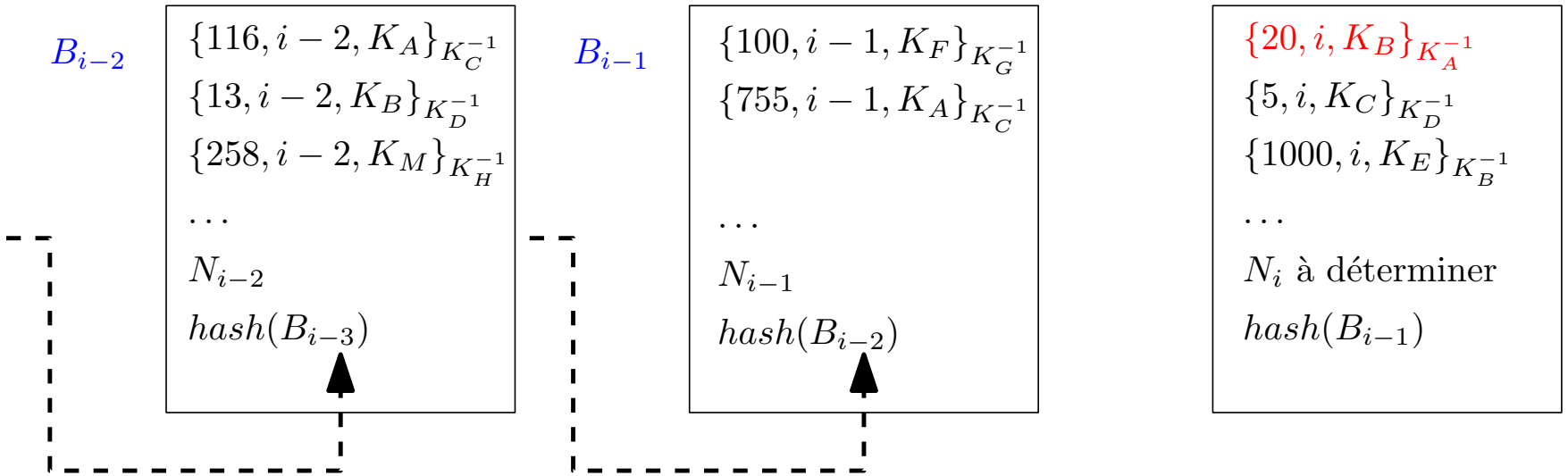
- Toutes les 10 minutes, chaque noeud du P2P déclenche la vérification
- Les nouvelles transactions sont vérifiées par rapport à la BlockChain
  BlockChain=historique des transactions depuis le démarrage (2009)



- Dans la BlockChain, on vérifie que K_A est crédité d'au moins 20฿

Protocole Bitcoin (II)

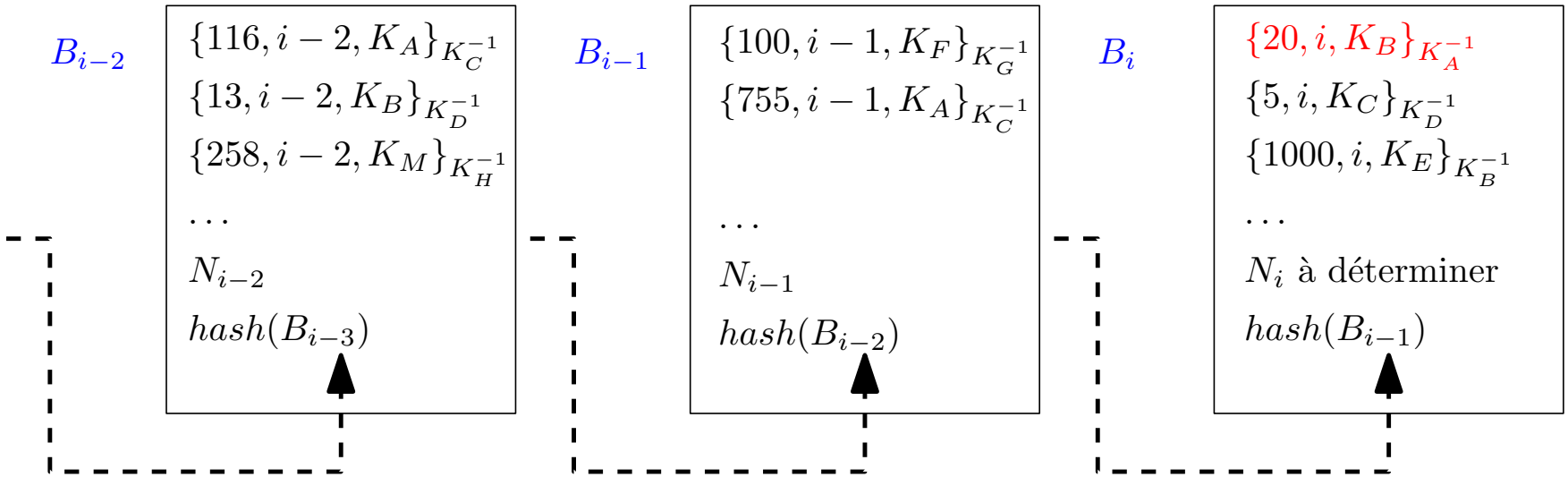
- Toutes les 10 minutes, chaque noeud du P2P déclenche la vérification
- Les nouvelles transactions sont vérifiées par rapport à la BlockChain
  BlockChain=historique des transactions depuis le démarrage (2009)



- Dans la BlockChain, on vérifie que K_A est crédité d'au moins 20฿
- Une fois validées les nouvelles transactions sont ajoutées à BlockChain

Protocole Bitcoin (II)

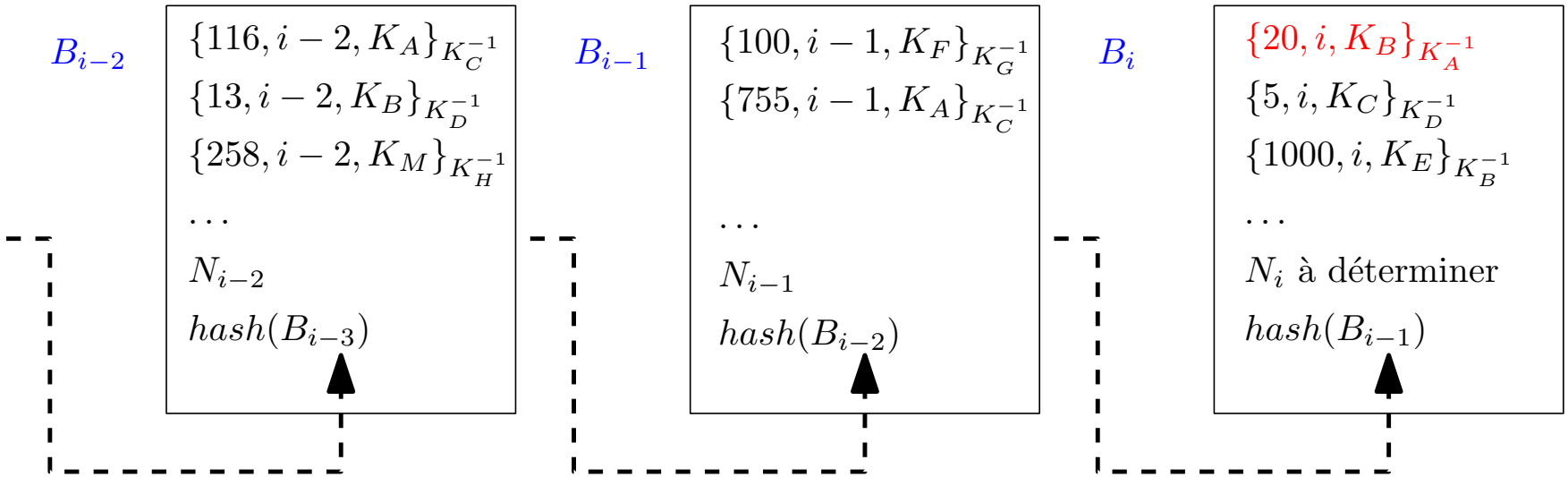
- Toutes les 10 minutes, chaque noeud du P2P déclenche la vérification
- Les nouvelles transactions sont vérifiées par rapport à la BlockChain
  BlockChain=historique des transactions depuis le démarrage (2009)



- Dans la BlockChain, on vérifie que K_A est crédité d'au moins 20Ƀ
- Une fois validées les nouvelles transactions sont ajoutées à BlockChain
- Block B_i = nouvelles transactions + $hash(B_{i-1})$ + N_i (à trouver)

Protocole Bitcoin (II)

- Toutes les 10 minutes, chaque noeud du P2P déclenche la vérification
- Les nouvelles transactions sont vérifiées par rapport à la BlockChain
  BlockChain=historique des transactions depuis le démarrage (2009)



- Dans la BlockChain, on vérifie que K_A est crédité d'au moins 20฿
- Une fois validées les nouvelles transactions sont ajoutées à BlockChain
- Block B_i = nouvelles transactions + $hash(B_{i-1})$ + N_i (à trouver)
- La BlockChain complétée est partagée sur le réseau P2P

Protocole Bitcoin : « sécurité par le consensus massif »

- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i

Protocole Bitcoin : « sécurité par le consensus massif »

- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i
- Pour pouvoir valider un bloc, les pairs doivent déterminer N_i tel que :
  - $hash(B_i)$ a une valeur binaire commençant par quatre 0 (par ex.)

Protocole Bitcoin : « sécurité par le consensus massif »

- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i
- Pour pouvoir valider un bloc, les pairs doivent déterminer N_i tel que :
  - $hash(B_i)$ a une valeur binaire commençant par quatre 0 (par ex.)
  - $hash(\underbrace{\text{nouvelles transactions} + hash(B_{i-1})}_{c} + N_i) = 0000\ldots$

Protocole Bitcoin : « sécurité par le consensus massif »

- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i
- Pour pouvoir valider un bloc, les pairs doivent déterminer N_i tel que :
  - $hash(B_i)$ a une valeur binaire commençant par quatre 0 (par ex.)
  - $hash(\underbrace{\text{nouvelles transactions} + hash(B_{i-1})}_{C} + N_i) = 0000\ldots$
- Comment trouver N_i t.q. $hash(C + N_i) = 0000\ldots$?

Protocole Bitcoin : « sécurité par le consensus massif »

- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i
- Pour pouvoir valider un bloc, les pairs doivent déterminer N_i tel que :
  - $hash(B_i)$ a une valeur binaire commençant par quatre 0 (par ex.)
  - $hash(\underbrace{\text{nouvelles transactions} + hash(B_{i-1})}_{C} + N_i) = 0000\ldots$
- Comment trouver N_i t.q. $hash(C + N_i) = 0000\ldots$?
  - Problème calculatoire difficile ! !

Protocole Bitcoin : « sécurité par le consensus massif »

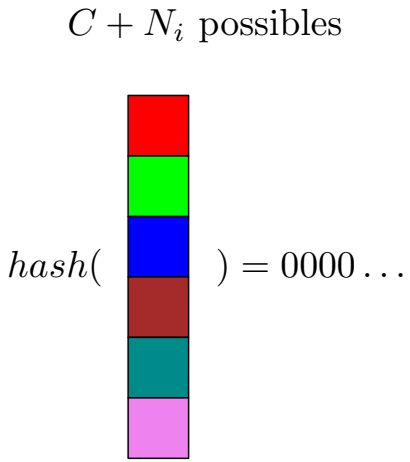
- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i
- Pour pouvoir valider un bloc, les pairs doivent déterminer N_i tel que :
  - $hash(B_i)$ a une valeur binaire commençant par quatre 0 (par ex.)
  - $hash(\underbrace{\text{nouvelles transactions} + hash(B_{i-1})}_{C} + N_i) = 0000\ldots$
- Comment trouver N_i t.q. $hash(C + N_i) = 0000\ldots$?
  - Problème calculatoire difficile ! !
  - Impossible à résoudre en 10 minutes avec 1, 10, 100 machines

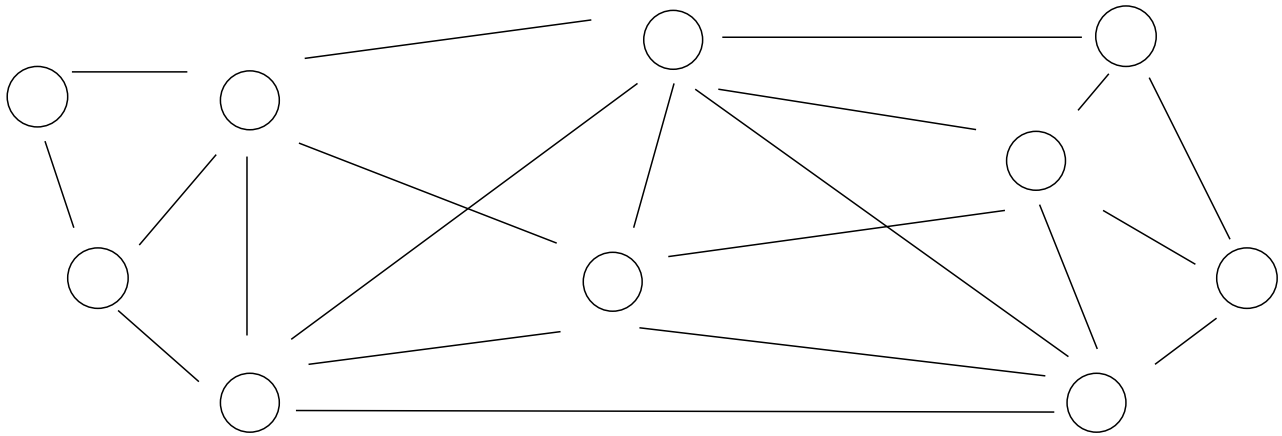
Protocole Bitcoin : « sécurité par le consensus massif »

- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i
- Pour pouvoir valider un bloc, les pairs doivent déterminer N_i tel que :
  - $hash(B_i)$ a une valeur binaire commençant par quatre 0 (par ex.)
  - $hash(\underbrace{\text{nouvelles transactions} + hash(B_{i-1})}_{C} + N_i) = 0000\ldots$
- Comment trouver N_i t.q. $hash(C + N_i) = 0000\ldots$?
  - Problème calculatoire difficile ! !
  - Impossible à résoudre en 10 minutes avec 1, 10, 100 machines
  - Possible uniquement si résolution distribuée entre de nombreux pairs

Protocole Bitcoin : « sécurité par le consensus massif »

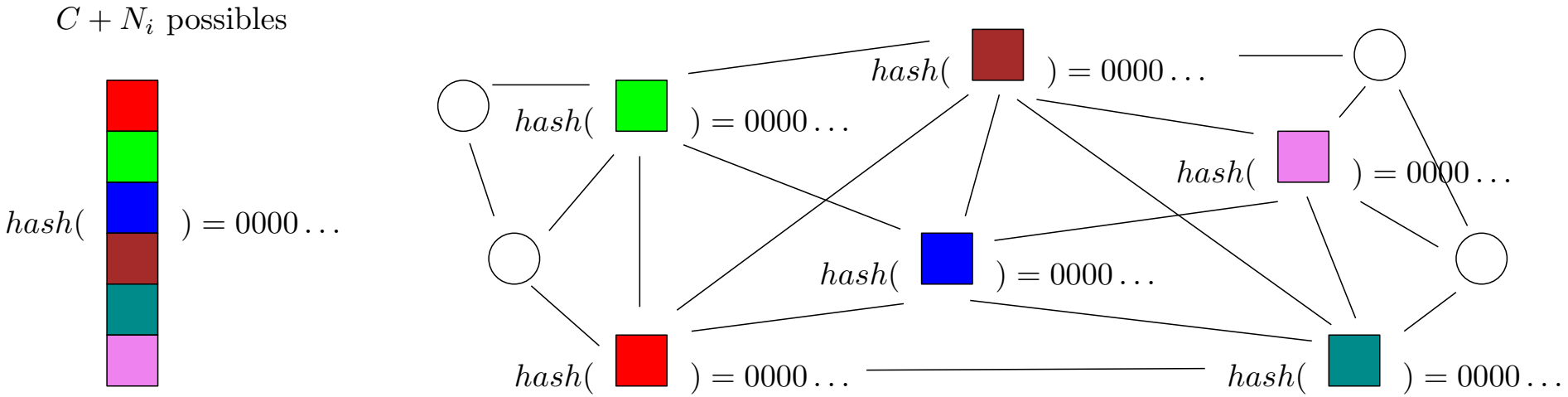
- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i
- Pour pouvoir valider un bloc, les pairs doivent déterminer N_i tel que :
  - $hash(B_i)$ a une valeur binaire commençant par quatre 0 (par ex.)
  - $hash(\underbrace{\text{nouvelles transactions} + hash(B_{i-1})}_{C} + N_i) = 0000\ldots$
- Comment trouver N_i t.q. $hash(C + N_i) = 0000\ldots$?
  - Problème calculatoire difficile ! !
  - Impossible à résoudre en 10 minutes avec 1, 10, 100 machines
  - Possible uniquement si résolution distribuée entre de nombreux pairs





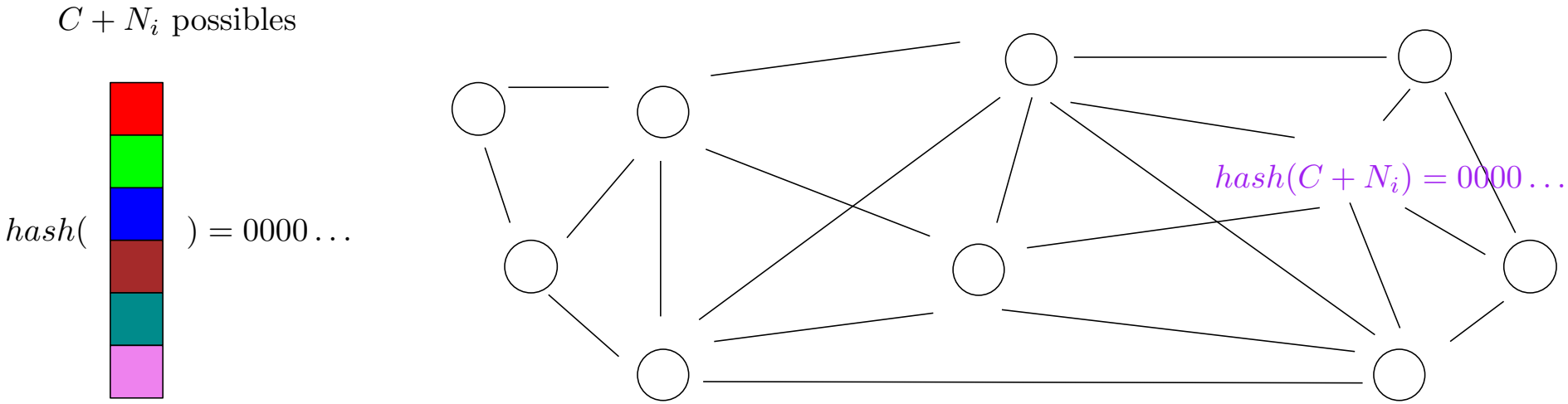
Protocole Bitcoin : « sécurité par le consensus massif »

- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i
- Pour pouvoir valider un bloc, les pairs doivent déterminer N_i tel que :
  - $hash(B_i)$ a une valeur binaire commençant par quatre 0 (par ex.)
  - $hash(\underbrace{\text{nouvelles transactions} + hash(B_{i-1})}_{C} + N_i) = 0000\ldots$
- Comment trouver N_i t.q. $hash(C + N_i) = 0000\ldots$?
  - Problème calculatoire difficile ! !
  - Impossible à résoudre en 10 minutes avec 1, 10, 100 machines
  - Possible uniquement si résolution distribuée entre de nombreux pairs



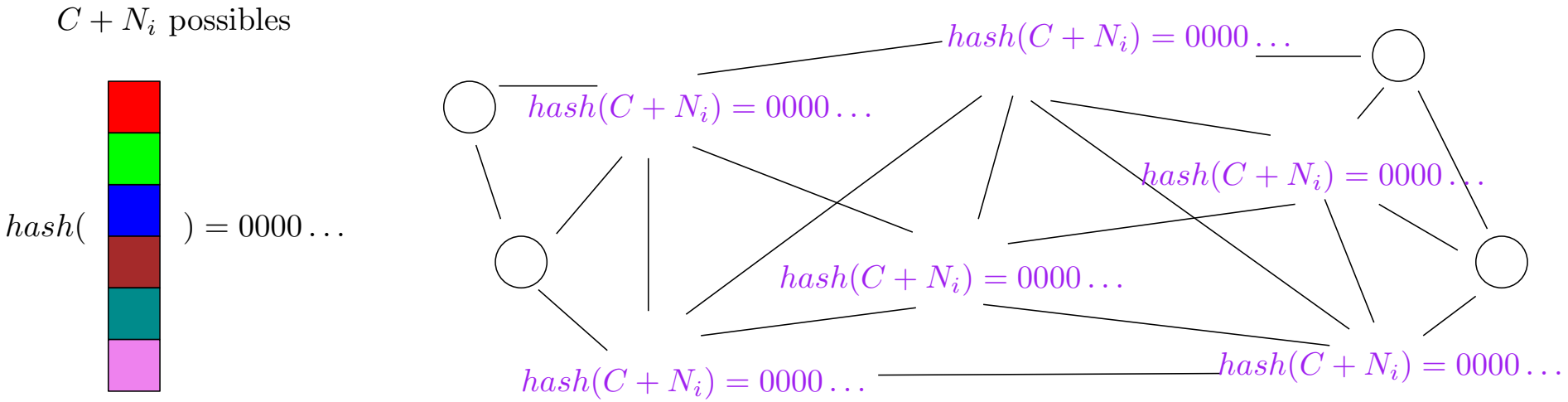
Protocole Bitcoin : « sécurité par le consensus massif »

- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i
- Pour pouvoir valider un bloc, les pairs doivent déterminer N_i tel que :
  - $hash(B_i)$ a une valeur binaire commençant par quatre 0 (par ex.)
  - $hash(\underbrace{\text{nouvelles transactions} + hash(B_{i-1})}_{C} + N_i) = 0000\ldots$
- Comment trouver N_i t.q. $hash(C + N_i) = 0000\ldots$?
  - Problème calculatoire difficile ! !
  - Impossible à résoudre en 10 minutes avec 1, 10, 100 machines
  - Possible uniquement si résolution distribuée entre de nombreux pairs



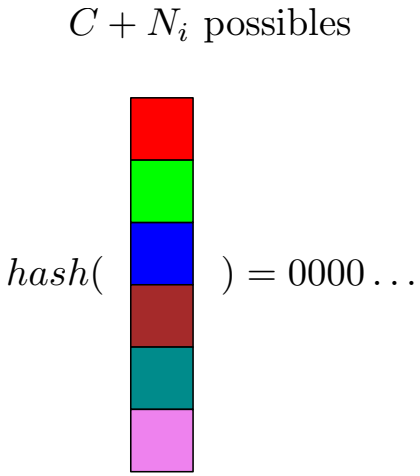
Protocole Bitcoin : « sécurité par le consensus massif »

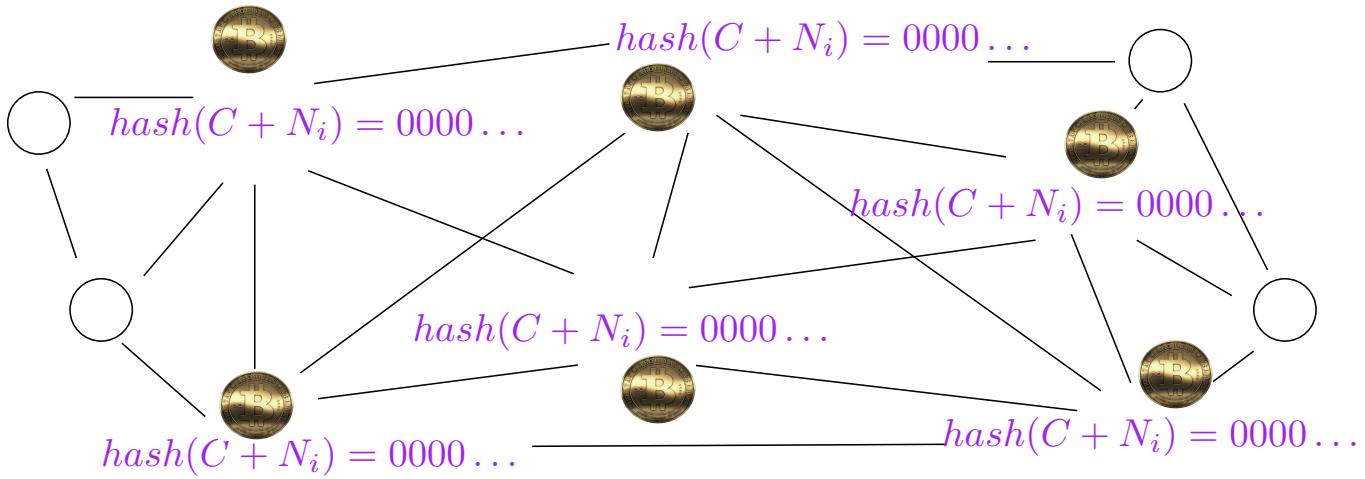
- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i
- Pour pouvoir valider un bloc, les pairs doivent déterminer N_i tel que :
  - $hash(B_i)$ a une valeur binaire commençant par quatre 0 (par ex.)
  - $hash(\underbrace{\text{nouvelles transactions} + hash(B_{i-1})}_{C} + N_i) = 0000\ldots$
- Comment trouver N_i t.q. $hash(C + N_i) = 0000\ldots$?
  - Problème calculatoire difficile ! !
  - Impossible à résoudre en 10 minutes avec 1, 10, 100 machines
  - Possible uniquement si résolution distribuée entre de nombreux pairs



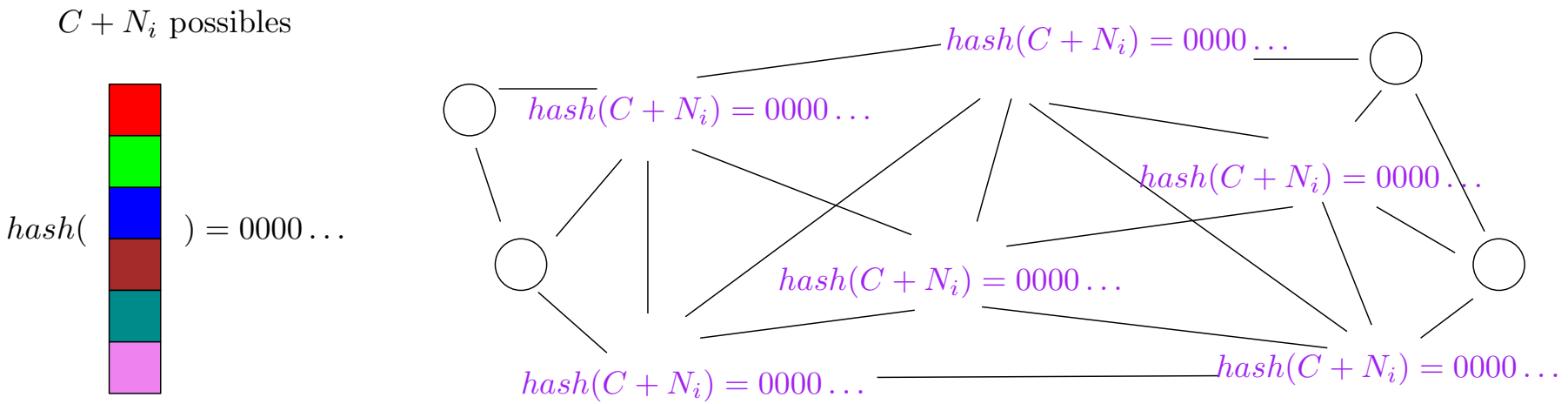
Protocole Bitcoin : « sécurité par le consensus massif »

- Pour éviter la tricherie (*e.g.* double dépense) la sécurité repose sur N_i
- Pour pouvoir valider un bloc, les pairs doivent déterminer N_i tel que :
  - $hash(B_i)$ a une valeur binaire commençant par quatre 0 (par ex.)
  - $hash(\underbrace{\text{nouvelles transactions} + hash(B_{i-1})}_{C} + N_i) = 0000\ldots$
- Comment trouver N_i t.q. $hash(C + N_i) = 0000\ldots$?
  - Problème calculatoire difficile ! !
  - Impossible à résoudre en 10 minutes avec 1, 10, 100 machines
  - Possible uniquement si résolution distribuée entre de nombreux pairs



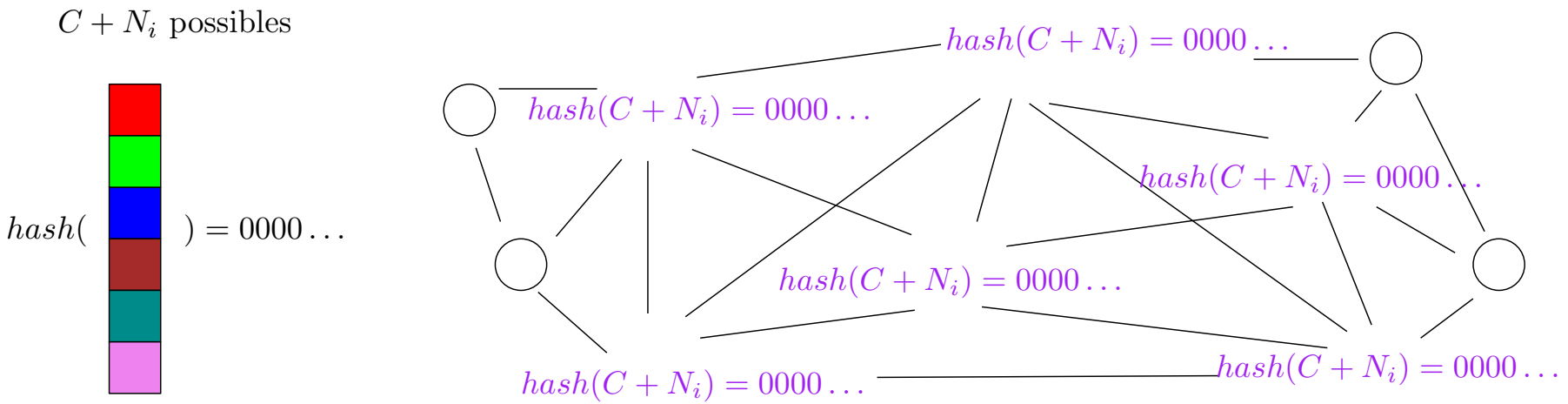


Protocole Bitcoin : « sécurité par le consensus massif »(II)



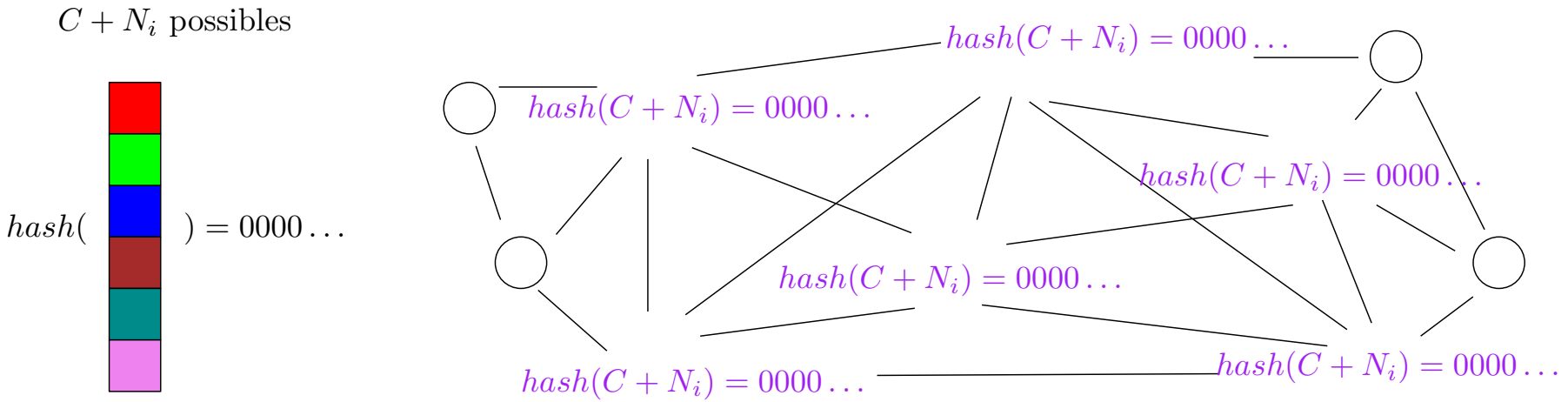
- Consensus des pairs sur $C + N_i$ (C contient les nouvelles transactions)

Protocole Bitcoin : « sécurité par le consensus massif »(II)



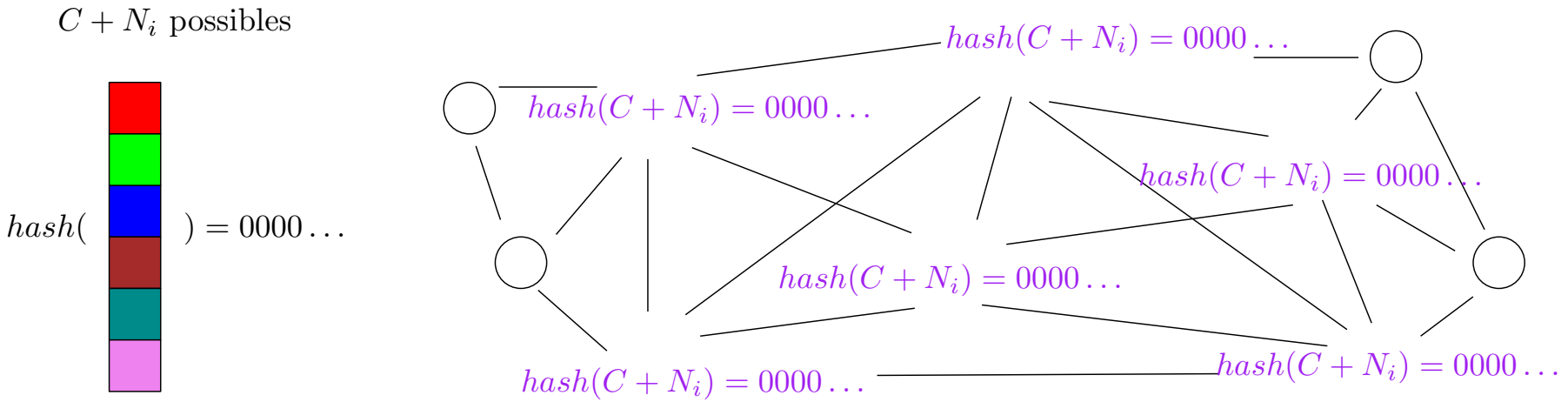
- Consensus des pairs sur $C + N_i$ (C contient les nouvelles transactions)
- D'où consensus des pairs sur les nouvelles transactions !

Protocole Bitcoin : « sécurité par le consensus massif »(II)



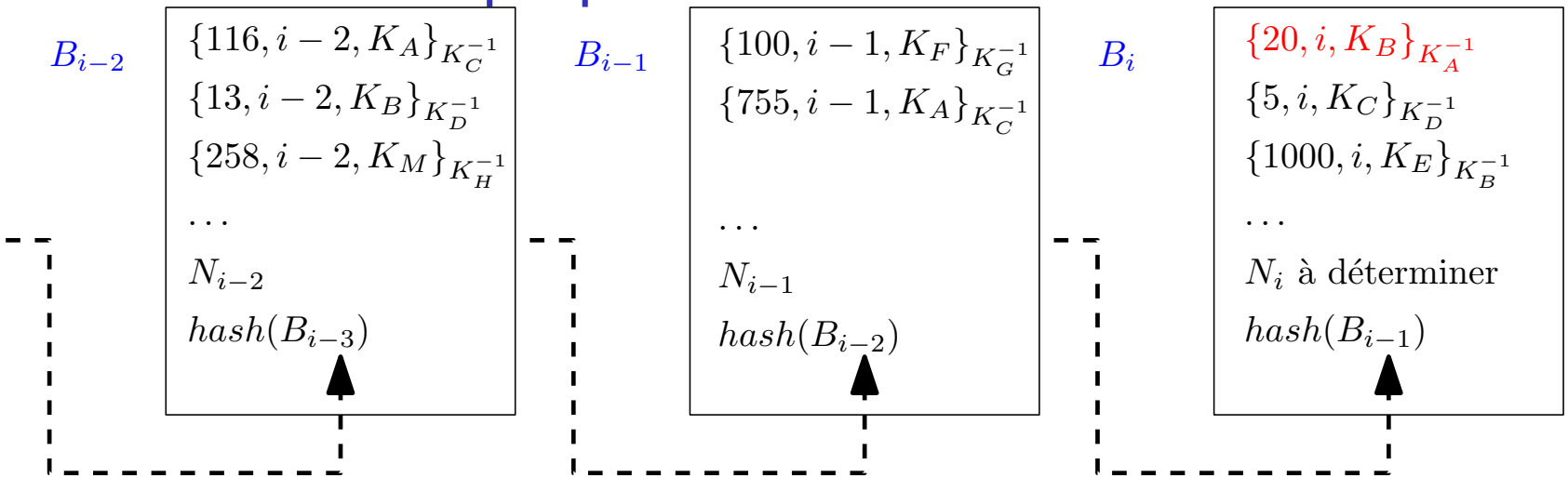
- Consensus des pairs sur $C + N_i$ (C contient les nouvelles transactions)
- D'où consensus des pairs sur les nouvelles transactions !
- Il est impossible pour un agent isolé de tricher :
  - double dépense
  - dépenser plus que le crédit d'un compte

Protocole Bitcoin : « sécurité par le consensus massif »(II)



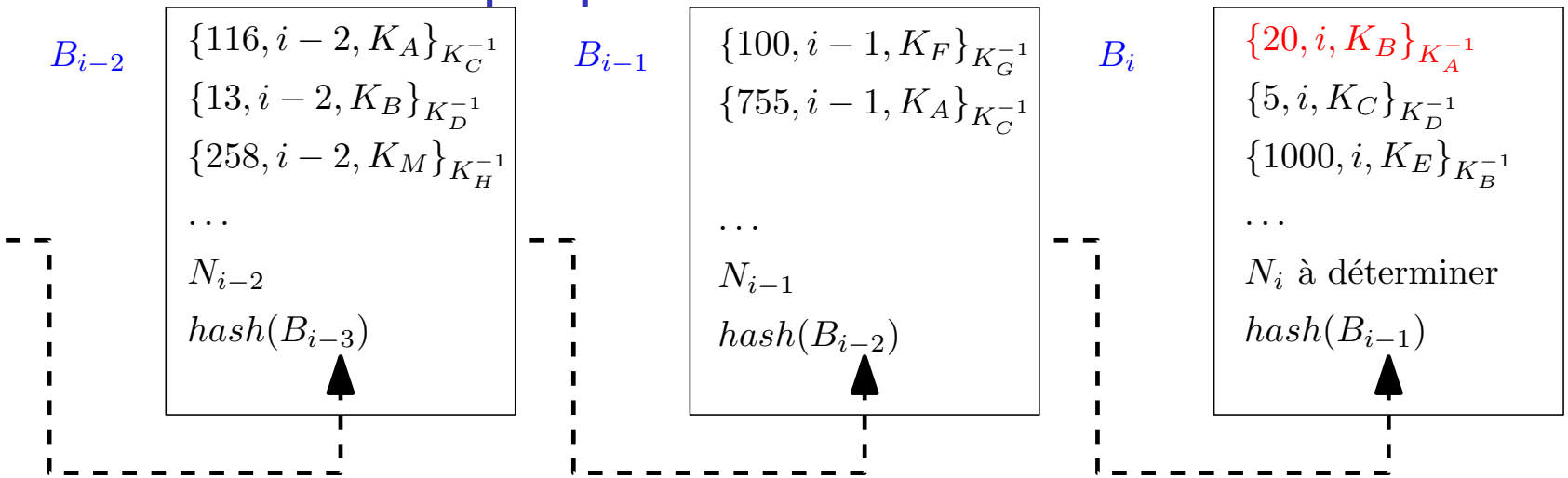
- Consensus des pairs sur $C + N_i$ (C contient les nouvelles transactions)
- D'où consensus des pairs sur les nouvelles transactions !
- Il est impossible pour un agent isolé de tricher :
  - double dépense
  - dépenser plus que le crédit d'un compte
- Pour tricher, il faut détenir plus de 51% de la puissance de calcul !

Protocole Bitcoin : propriétés attendues



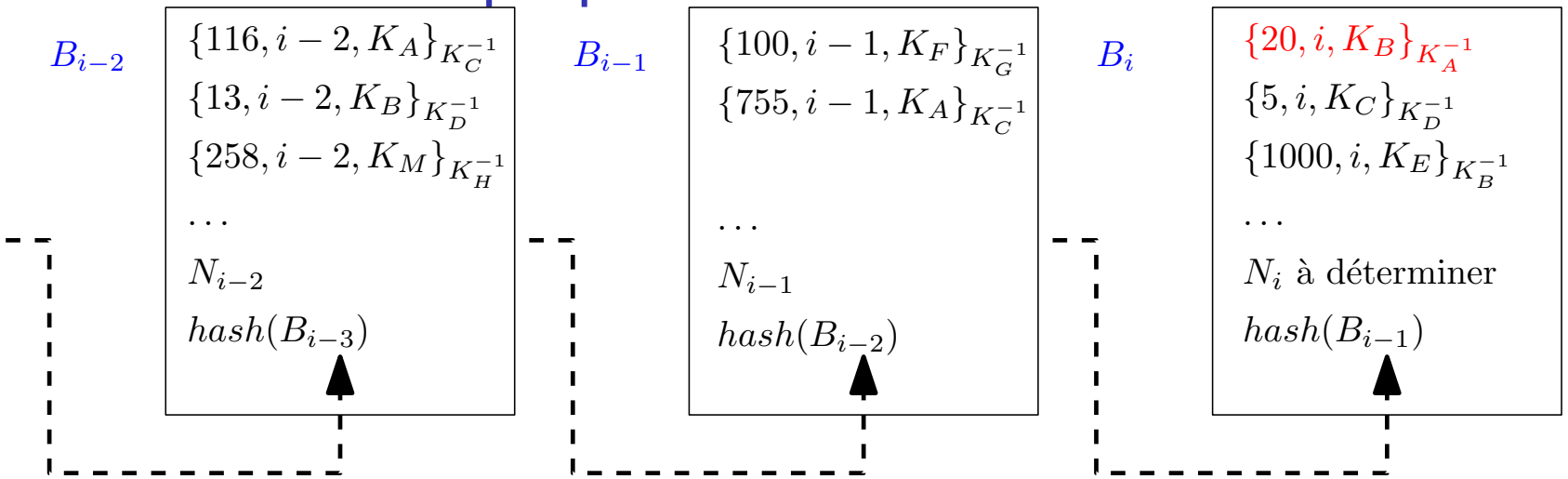
- Secrets :

Protocole Bitcoin : propriétés attendues



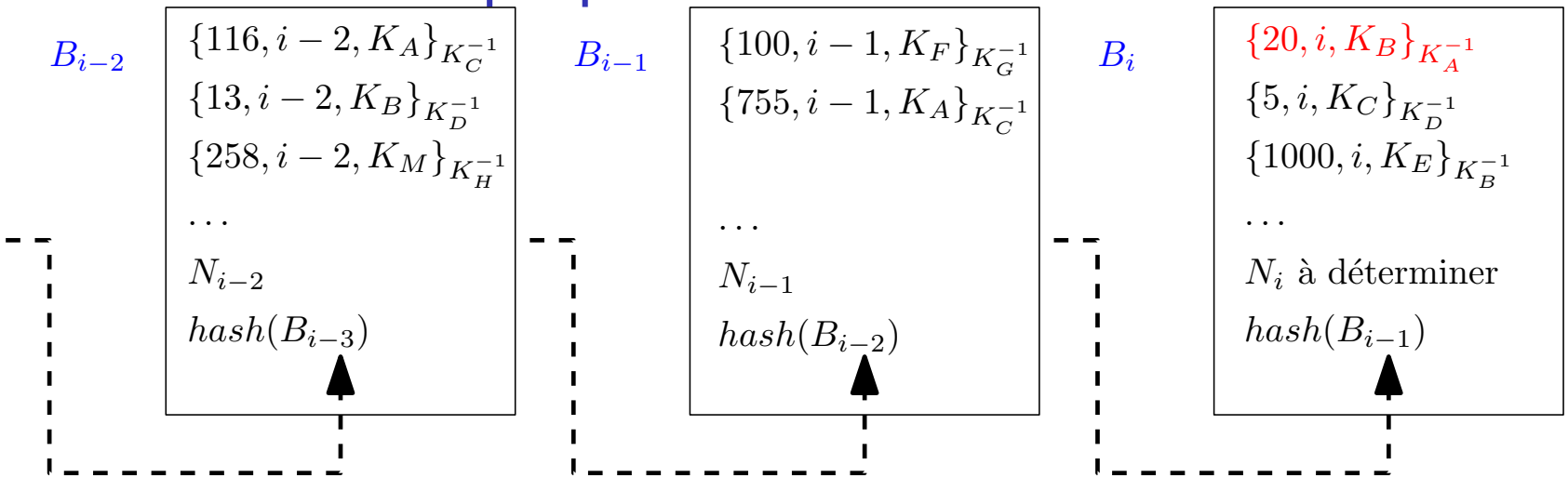
- Secrets : aucun (tous les transferts de compte à compte sont publics)
- Messages authentifiés :

Protocole Bitcoin : propriétés attendues



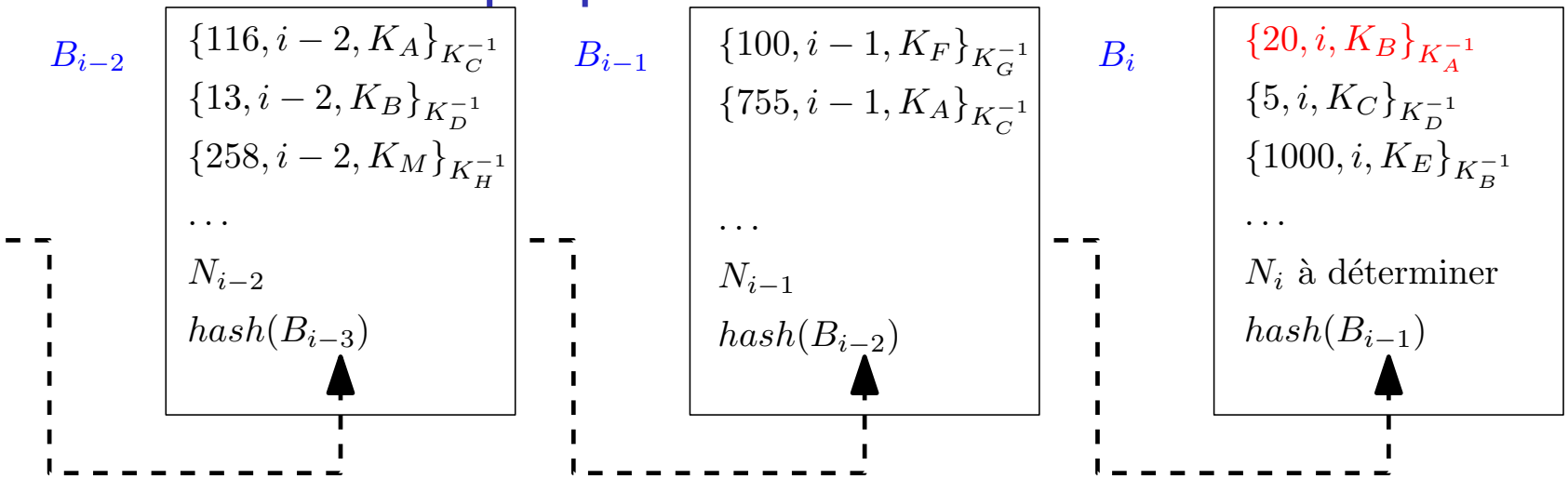
- Secrets : aucun (tous les transferts de compte à compte sont publics)
- Messages authentifiés : toutes les transactions
- Fraîcheur :

Protocole Bitcoin : propriétés attendues



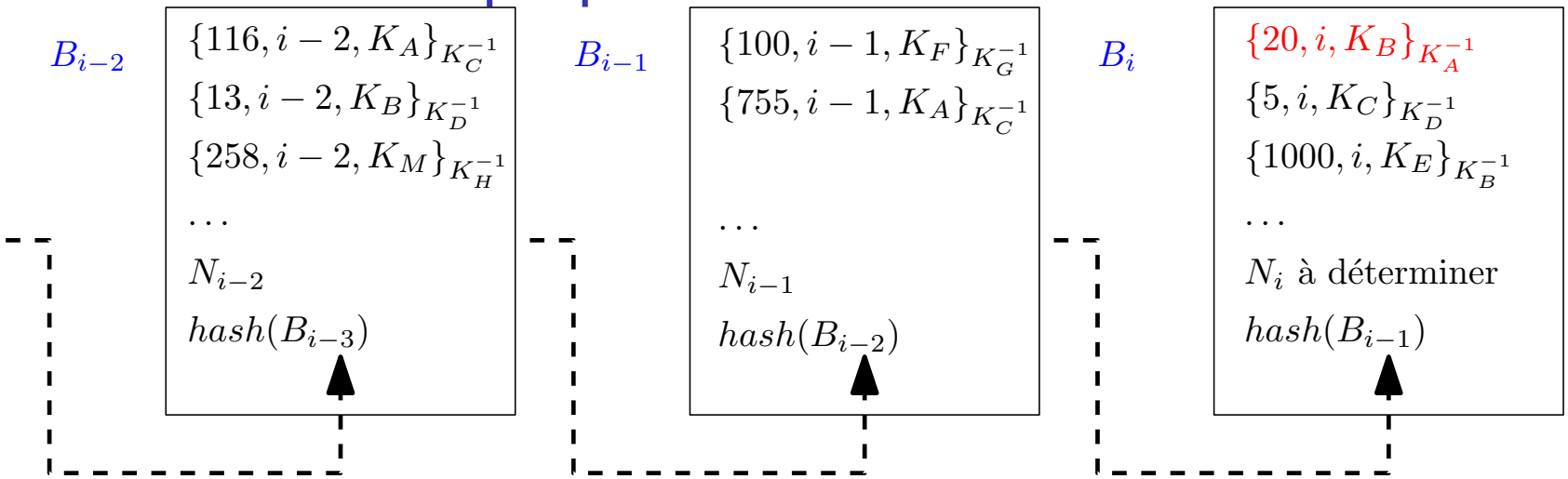
- Secrets : aucun (tous les transferts de compte à compte sont publics)
- Messages authentifiés : toutes les transactions
- Fraîcheur : toutes les transactions, tous les N_i, tous les *hash*
- Agents authentifiés :

Protocole Bitcoin : propriétés attendues



- Secrets : aucun (tous les transferts de compte à compte sont publics)
- Messages authentifiés : toutes les transactions
- Fraîcheur : toutes les transactions, tous les N_i, tous les *hash*
- Agents authentifiés : les émetteurs de transaction sont authentifiés par tous les participants au réseau P2P
- Anonymat :

Protocole Bitcoin : propriétés attendues



- Secrets : aucun (tous les transferts de compte à compte sont publics)
- Messages authentifiés : toutes les transactions
- Fraîcheur : toutes les transactions, tous les N_i, tous les *hash*
- Agents authentifiés : les émetteurs de transaction sont authentifiés par tous les participants au réseau P2P
- Anonymat : s'il n'est pas possible de savoir qui émet une transaction sur le P2P, les agents associés aux comptes restent anonymes

Sécurité de Bitcoin : ce que tout utilisateur doit vérifier

- l'homogéinité du réseau P2P : moins de 51% de la puissance de calcul doit être aux mains d'une entité malveillante

Sécurité de Bitcoin : ce que tout utilisateur doit vérifier

- l'homogéinité du réseau P2P : moins de 51% de la puissance de calcul doit être aux mains d'une entité malveillante
- que son numéro de compte et sa clé privée restent secrets

  par ex., le numéro : `1CfvVCqnXwVEdxXaSETP7Yx8PvFCKs66a2` et la clé : `5JHS5prvTHw7V4pX9hAiMu9tUNFJDRoURTuzn6V8wctsxb9TtJT`

Sécurité de Bitcoin : ce que tout utilisateur doit vérifier

- l'homogéinité du réseau P2P : moins de 51% de la puissance de calcul doit être aux mains d'une entité malveillante
- que son numéro de compte et sa clé privée restent secrets

  par ex., le numéro : `1CfvVCqnXwVEdxXaSETP7Yx8PvFCKs66a2` et la clé : `5JHS5prvTHw7V4pX9hAiMu9tUNFJDRoURTuzn6V8wctsxb9TtJT`

  - Stockées sur un disque externe, clé USB, etc.

Sécurité de Bitcoin : ce que tout utilisateur doit vérifier

- l'homogéinité du réseau P2P : moins de 51% de la puissance de calcul doit être aux mains d'une entité malveillante
- que son numéro de compte et sa clé privée restent secrets

  par ex., le numéro : `1CfvVCqnXwVEdxXaSETP7Yx8PvFCKs66a2` et la clé : `5JHS5prvTHw7V4pX9hAiMu9tUNFJDRoURTuzn6V8wctsxb9TtJT`

  - Stockées sur un disque externe, clé USB, etc.
  - Stockées sur du papier :

Partie II :

Protocoles crypto, attaques et vérification

Protocoles crypto, attaques et vérification

Plan

- Bibliographie
- Mises en garde
- Le paiement avec une carte bancaire à puce
- Les failles
- Vérification des protocoles cryptographiques
  - Définition des Capacités de l'intrus
  - Le modèle de Dolev-Yao
  - Vérification "logique" des protocoles

Bibliographie

- Jacques Patarin. *La Cryptographie des cartes bancaires*. Pour la science. Hors série. Juillet-Octobre 2002.
- L. Paulson. *The inductive approach to verifying cryptographic protocols*. Journal of Computer Security. 1998.
  `http://www.cl.cam.ac.uk/~lp15/papers/Auth/jcs.pdf`

Mises en garde

Art. 67-1. (L. n 91-1382 du 30 déc. 1991)

Seront punis d'un emprisonnement d'un an à sept ans et d'une amende de 3.600 F à 5.000.000 F ou de l'une de ces deux peines seulement :

1. Ceux qui auront contrefait ou falsifié une carte de paiement ou de retrait ;
2. Ceux qui, en connaissance de cause, auront fait usage d'une carte de paiement ou de retrait contrefaite ou falsifiée ;
3. ...

[Bruce Schneier]
"Il est certainement plus simple d'implémenter de la mauvaise sécurité et de rendre hors la loi tous ceux qui le font remarquer que d'implémenter de la bonne sécurité."

Exemple de la Carte bancaire

Acteurs (ou agents) :

- (A)lice
- (C)arte bancaire (détenue par A)
- (T)erminal du commerçant
- (B)anque (banque émettrice de la carte)

Exemple de la Carte bancaire

Protocole de transaction :

Exemple de la Carte bancaire

Protocole de transaction :

- A introduit sa carte C

Exemple de la Carte bancaire

Protocole de transaction :

- A introduit sa carte C
- le commerçant saisit le montant m sur T

Exemple de la Carte bancaire

Protocole de transaction :

- A introduit sa carte C
- le commerçant saisit le montant m sur T
- T authentifie C

Exemple de la Carte bancaire

Protocole de transaction :

- A introduit sa carte C
- le commerçant saisit le montant m sur T
- T authentifie C
- A donne son code (3456) à C

Exemple de la Carte bancaire

Protocole de transaction :

- A introduit sa carte C
- le commerçant saisit le montant m sur T
- T authentifie C
- A donne son code (3456) à C

Si m dépasse 100 euros (et dans 20% des cas)

Exemple de la Carte bancaire

Protocole de transaction :

- A introduit sa carte C
- le commerçant saisit le montant m sur T
- T authentifie C
- A donne son code (3456) à C

  Si m dépasse 100 euros (et dans 20% des cas)

- T demande l'autorisation à B pour C

Exemple de la Carte bancaire

Protocole de transaction :

- A introduit sa carte C
- le commerçant saisit le montant m sur T
- T authentifie C
- A donne son code (3456) à C

Si m dépasse 100 euros (et dans 20% des cas)

- T demande l'autorisation à B pour C
- B donne l'autorisation

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

- (A)lice possède un code secret : 3456

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

- (A)lice possède un code secret : 3456
- (B)anque possède
  - une clé "publique" K_B
  - des clés secrètes K_B^{-1}, K_{CB}

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

- (A)lice possède un code secret : 3456
- (B)anque possède
  - une clé "publique" K_B
  - des clés secrètes K_B^{-1}, K_{CB}
- (C)arte possède des informations "publiques"

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

- (A)lice possède un code secret : 3456
- (B)anque possède
  - une clé "publique" K_B
  - des clés secrètes K_B^{-1}, K_{CB}
- (C)arte possède des informations "publiques"
  - $Data$= nom, prénom, numéro carte, date de validité

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

- (A)lice possède un code secret : 3456
- (B)anque possède
  - une clé "publique" K_B
  - des clés secrètes K_B^{-1}, K_{CB}
- (C)arte possède des informations "publiques"
  - $Data$= nom, prénom, numéro carte, date de validité
  - Valeur de Signature $VS = \{hash(Data)\}_{K_B^{-1}}$

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

- (A)lice possède un code secret : 3456
- (B)anque possède
  - une clé "publique" K_B
  - des clés secrètes K_B^{-1}, K_{CB}
- (C)arte possède des informations "publiques"
  - $Data$= nom, prénom, numéro carte, date de validité
  - Valeur de Signature $VS = \{hash(Data)\}_{K_B^{-1}}$

  et une clé "secrète" : K_{CB} (DES)

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

- (A)lice possède un code secret : 3456
- (B)anque possède
  - une clé "publique" K_B
  - des clés secrètes K_B^{-1}, K_{CB}
- (C)arte possède des informations "publiques"
  - $Data$= nom, prénom, numéro carte, date de validité
  - Valeur de Signature $VS = \{hash(Data)\}_{K_B^{-1}}$

  et une clé "secrète" : K_{CB} (DES)
- (T)erminal possède
  - $hash$
  - une clé "publique" K_B (RSA)

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

- (A)lice possède son code : 3456
- (C)arte possède $Data$ et $\{hash(Data)\}_{K_B^{-1}}$
- (T)erminal possède $hash$ et la clé "publique" K_B

Phase hors ligne de la transaction (estimation) :

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

- (A)lice possède son code : 3456
- (C)arte possède $Data$ et $\{hash(Data)\}_{K_B^{-1}}$
- (T)erminal possède $hash$ et la clé "publique" K_B

Phase hors ligne de la transaction (estimation) :

- T authentifie C
  1. $C \hookrightarrow T : Data, \{hash(Data)\}_{K_B^{-1}}$

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

- (A)lice possède son code : 3456
- (C)arte possède $Data$ et $\{hash(Data)\}_{K_B^{-1}}$
- (T)erminal possède $hash$ et la clé "publique" K_B

Phase hors ligne de la transaction (estimation) :

- T authentifie C
  1. $C \hookrightarrow T : Data, \{hash(Data)\}_{K_B^{-1}}$
- A donne son code à C (C authentifie A)
  2. $T \hookrightarrow A$: code ?
  3. $A \hookrightarrow C$: 3456
  4. $C \hookrightarrow T$: ok

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

Si le montant est supérieur à 100 euros

- T demande l'autorisation à B pour C
  5. $T \hookrightarrow B$: Demande d'authentification

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

Si le montant est supérieur à 100 euros

- T demande l'autorisation à B pour C
  5. $T \hookrightarrow B$: Demande d'authentification
- B réalise l'authentification en ligne de C
  6. $B \hookrightarrow T : N_B$
  7. $T \hookrightarrow C : N_B$
  8. $C \hookrightarrow T : A, \{N_B\}_{K_{CB}}$
  9. $T \hookrightarrow B : A, \{N_B\}_{K_{CB}}$

Exemple de la Carte bancaire (Protocole B0, 1987-2005)

Si le montant est supérieur à 100 euros

- T demande l'autorisation à B pour C
  5. $T \hookrightarrow B$: Demande d'authentification
- B réalise l'authentification en ligne de C
  6. $B \hookrightarrow T : N_B$
  7. $T \hookrightarrow C : N_B$
  8. $C \hookrightarrow T : A, \{N_B\}_{K_{CB}}$
  9. $T \hookrightarrow B : A, \{N_B\}_{K_{CB}}$
- B donne l'autorisation
  10. $B \hookrightarrow T$: ok

Quelques failles de la carte bancaire (Protocole B0, 1987-2005)

Initialement la sécurité de la carte reposait beaucoup sur :

- la non réplicabilité de la carte
- le secret autour des clés employées, du protocole, ...

Quelques failles de la carte bancaire (Protocole B0, 1987-2005)

Initialement la sécurité de la carte reposait beaucoup sur :

- la non réplicabilité de la carte
- le secret autour des clés employées, du protocole, ...

Mais

Quelques failles de la carte bancaire (Protocole B0, 1987-2005)

Initialement la sécurité de la carte reposait beaucoup sur :

- la non réplicabilité de la carte
- le secret autour des clés employées, du protocole, ...

Mais

- Faille cryptographique :
  clés RSA 320 bits ne sont plus sûres (1988)

Quelques failles de la carte bancaire (Protocole B0, 1987-2005)

Initialement la sécurité de la carte reposait beaucoup sur :

- la non réplicabilité de la carte
- le secret autour des clés employées, du protocole, ...

Mais

- Faille cryptographique :
  clés RSA 320 bits ne sont plus sûres (1988)
- Faille logique du protocole :
  pas de lien "code à 4 chiffres" et authentification !

Quelques failles de la carte bancaire (Protocole B0, 1987-2005)

Initialement la sécurité de la carte reposait beaucoup sur :

- la non réplicabilité de la carte
- le secret autour des clés employées, du protocole, ...

Mais

- Faille cryptographique :
  clés RSA 320 bits ne sont plus sûres (1988)
- Faille logique du protocole :
  pas de lien "code à 4 chiffres" et authentification !
- Faille physique :
  réplicabilité $\Rightarrow$ Yescard ! (Humpich 1998)

Quelques failles (Protocole B0, 1987-2005)

Faille logique du protocole :

1. $C \hookrightarrow T : Data, \{hash(Data)\}_{K_B^{-1}}$
2. $T \hookrightarrow A$: code ?
3. $A \hookrightarrow C$: 3456
4. $C \hookrightarrow T$: ok

Quelques failles (Protocole B0, 1987-2005)

Faille logique du protocole :

1. $C \hookrightarrow T : Data, \{hash(Data)\}_{K_B^{-1}}$
2. $T \hookrightarrow A$: code ?
3. $A \hookrightarrow C$: 3456
4. $C \hookrightarrow T$: ok

$A \hookrightarrow C' : 7575$

$C' \hookrightarrow T$: ok

Quelques failles (Protocole B0, 1987-2005)

Faille logique du protocole :

1. $C \hookrightarrow T : Data, \{hash(Data)\}_{K_B^{-1}}$
2. $T \hookrightarrow A$: code ?
3. $A \hookrightarrow C' : 7575$
4. $C' \hookrightarrow T$: ok

Quelques failles (Protocole B0, 1987-2005)

Faille logique du protocole :

1. $C \hookrightarrow T : Data, \{hash(Data)\}_{K_B^{-1}}$
2. $T \hookrightarrow A$: code ?
3. $A \hookrightarrow C'$: 7575
4. $C' \hookrightarrow T$: ok

Implantée dans une Yescard :

1. $C \hookrightarrow T : Data, \{hash(Data)\}_{K_B^{-1}}$
2. $T \hookrightarrow A$: code ?
3. $A \hookrightarrow C$: 0000
4. $C \hookrightarrow T$: ok

Quelques failles (Protocole B0, 1987-2005)

Faille logique du protocole :

1. $C \hookrightarrow T : Data, \{hash(Data)\}_{K_B^{-1}}$
2. $T \hookrightarrow A$: code ?
3. $A \hookrightarrow C'$: 7575
4. $C' \hookrightarrow T$: ok

Implantée dans une Yescard (et fausse signature RSA)

1. $C \hookrightarrow T : ZZZ, \{hash(ZZZ)\}_{K_B^{-1}}$
2. $T \hookrightarrow A$: code ?
3. $A \hookrightarrow C$: 0000
4. $C \hookrightarrow T$: ok

Corrections appliquées sur le protocole B0

- 1999 : le GIE Cartes Bancaires alonge la taille des clés RSA

Corrections appliquées sur le protocole B0

- 1999 : le GIE Cartes Bancaires alonge la taille des clés RSA
- 2003 : Europay, MasterCard et Visa remplacent *B0* par *EMV*

Corrections appliquées sur le protocole B0

- 1999 : le GIE Cartes Bancaires alonge la taille des clés RSA
- 2003 : Europay, MasterCard et Visa remplacent *B0* par *EMV*
  - non pas un mais 3 protocoles : SDA, DDA, CDA

Corrections appliquées sur le protocole B0

- 1999 : le GIE Cartes Bancaires alonge la taille des clés RSA
- 2003 : Europay, MasterCard et Visa remplacent *B0* par *EMV*
  - non pas un mais 3 protocoles : SDA, DDA, CDA
  - conçu en 2003 et (*officiellement*) largement déployé depuis 2006

Corrections appliquées sur le protocole B0

- 1999 : le GIE Cartes Bancaires alonge la taille des clés RSA
- 2003 : Europay, MasterCard et Visa remplacent *B0* par *EMV*
  - non pas un mais 3 protocoles : SDA, DDA, CDA
  - conçu en 2003 et (*officiellement*) largement déployé depuis 2006
  - spécifications disponibles sur le web !

Corrections appliquées sur le protocole B0

- 1999 : le GIE Cartes Bancaires alonge la taille des clés RSA
- 2003 : Europay, MasterCard et Visa remplacent *B0* par *EMV*
  - non pas un mais 3 protocoles : SDA, DDA, CDA
  - conçu en 2003 et (*officiellement*) largement déployé depuis 2006
  - spécifications disponibles sur le web !
  - multi-applications

Vérification des protocoles crypto

Beaucoup de protocoles de la littérature ont des failles !

Il s'agit de failles logiques :

- liées à l'enchaînement des messages
- pas liées à l'implantation
- pas de cryptanalyse

protocoles crypto.= cas d'étude idéal pour les méthodes formelles

- relativement abstraits
- de taille réduite
- difficiles à vérifier manuellement

Il faut formaliser les capacités de l'intrus

Les capacités de l'intrus

① L'intrus ne fait pas de cryptanalyse ⇒ suppose les clés incassables

En pratique, il faut utiliser des clés de longueurs suffisantes

⇒ suivre les résultats de cassage de clés

Ex : RSA record à 768 bits (2010)

Carte bancaires *doivent être* au moins à RSA 1024 bits...

Compétition de factorisation RSA

RSA Number	Decimal digits	Binary digits	Cash prize offered	Factored on	Factored by
RSA-100	100	330		April 1991	Arjen K. Lenstra
RSA-110	110	364		April 1992	Arjen K. Lenstra and M.S. Manasse
RSA-120	120	397		June 1993	T. Denny et al.
RSA-129	129	426	$100 USD	April 1994	Arjen K. Lenstra et al.
RSA-130	130	430		April 10, 1996	Arjen K. Lenstra et al.
RSA-140	140	463		February 2, 1999	Herman J. J. te Riele et al.
RSA-150[3]	150	496		April 16, 2004	Kazumaro Aoki et al.
RSA-155	155	512		August 22, 1999	Herman J. J. te Riele et al.
RSA-160	160	530		April 1, 2003	Jens Franke et al., University of Bonn
RSA-170	170	563		*open*	
RSA-576	174	576	$10,000 USD	December 3, 2003	Jens Franke et al., University of Bonn
RSA-180	180	596		*open*	
RSA-190	190	629		*open*	
RSA-640	193	640	$20,000 USD	November 2, 2005	Jens Franke et al., University of Bonn
RSA-200	200	663		May 9, 2005	Jens Franke et al., University of Bonn
RSA-210	210	696		*open*	
RSA-704	212	704	$30,000 USD	*open*	

Compétition de factorisation RSA

Compétition	Prix	Statut	Date de factorisation	Par
RSA-576	USD 10 000	Factorisé	3 décembre 2003	J. Franke et al.
RSA-640	USD 20 000	Factorisé	2 novembre 2005	F. Bahr et al.
RSA-704	USD 30 000	Annulé	-	-
RSA-768	USD 50 000	Factorisé	15 janvier 2010	Divers organismes
RSA-896	USD 75 000	Annulé	-	-
RSA-1024	USD 100 000	Annulé	-	-
RSA-1536	USD 150 000	Annulé	-	-
RSA-2048	USD 200 000	Annulé	-	-

Exemple de faille et capacités de l'intrus

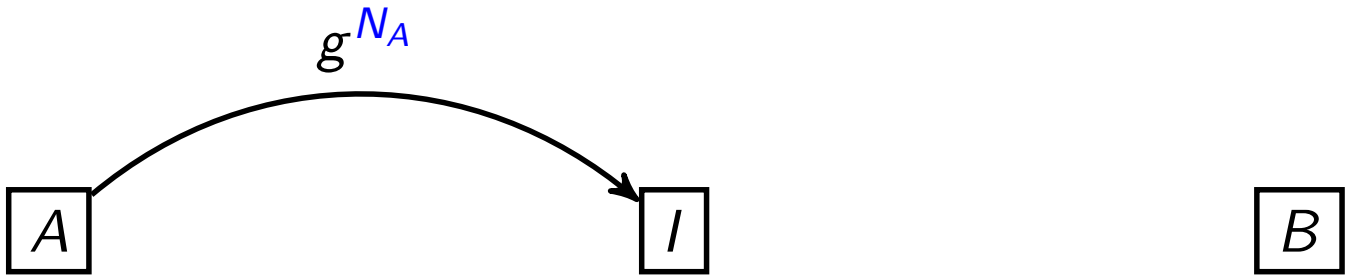
Version simplifiée de SSH en mode dégradé

1. $A \hookrightarrow B : g^x$
2. $B \hookrightarrow A : g^y$
3. $B \hookrightarrow A : \{login :\}_K$
4. $A \hookrightarrow B : \{A\}_K$
5. $B \hookrightarrow A : \{passwd :\}_K$
6. $A \hookrightarrow B : \{P\}_K$

$$K = (g^x)^y = (g^y)^x$$

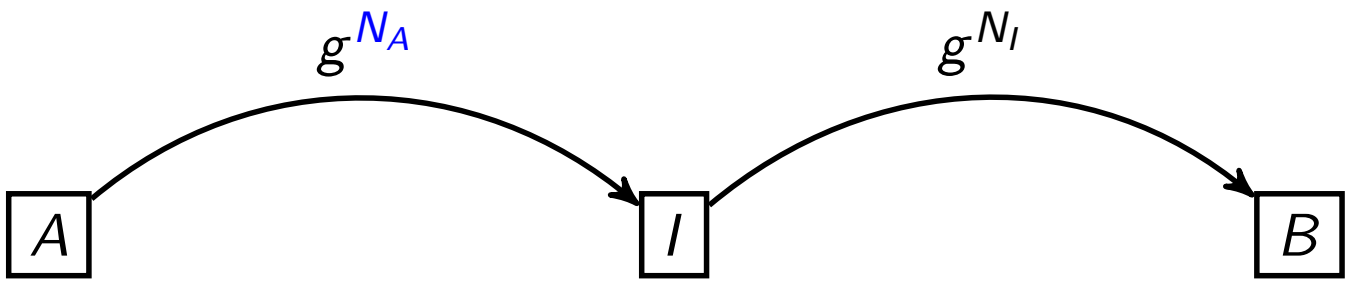
Exemple de failles et capacités de l'intrus

Exemple de failles et capacités de l'intrus



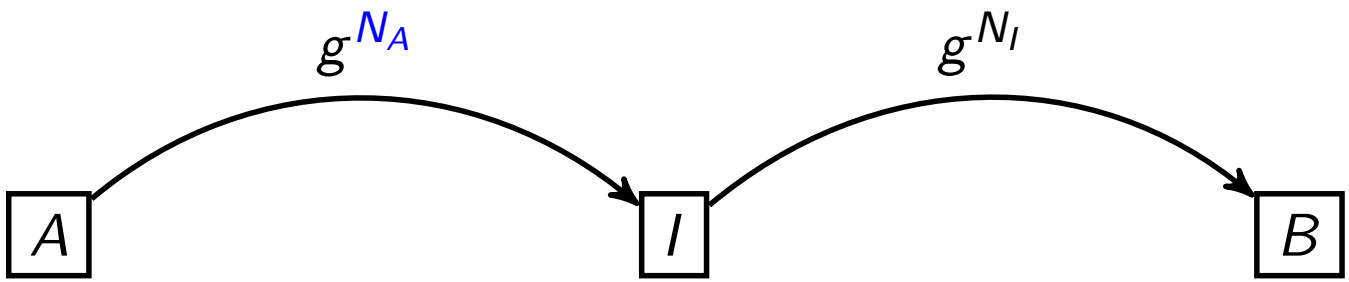
② L'intrus peut intercepter/bloquer tous les messages

Exemple de failles et capacités de l'intrus



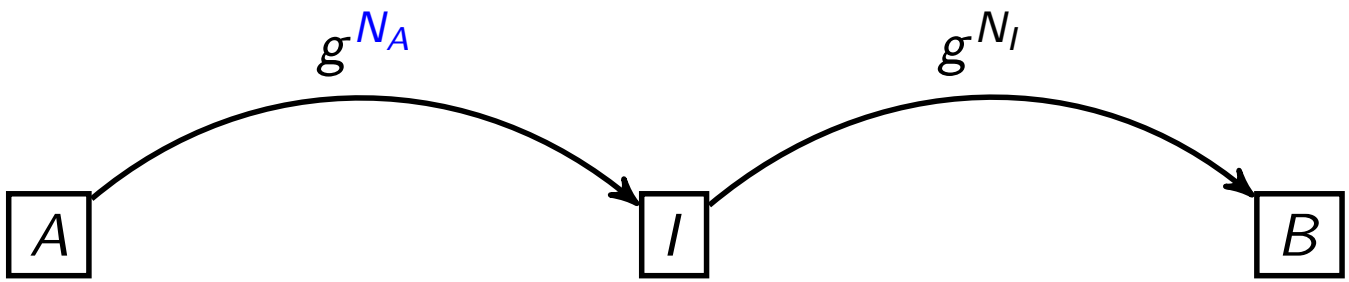
③ L'intrus connaît les données publiques (g)

Exemple de failles et capacités de l'intrus



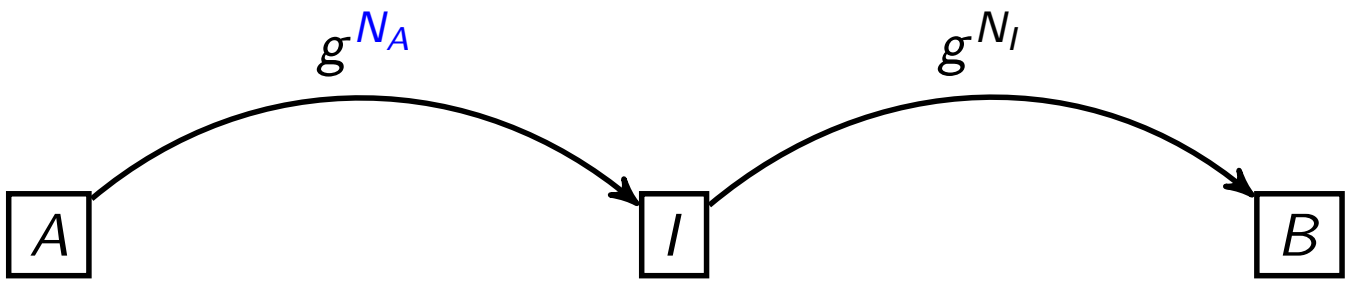
④ Il peut générer des valeurs aléatoires (N_I)

Exemple de failles et capacités de l'intrus



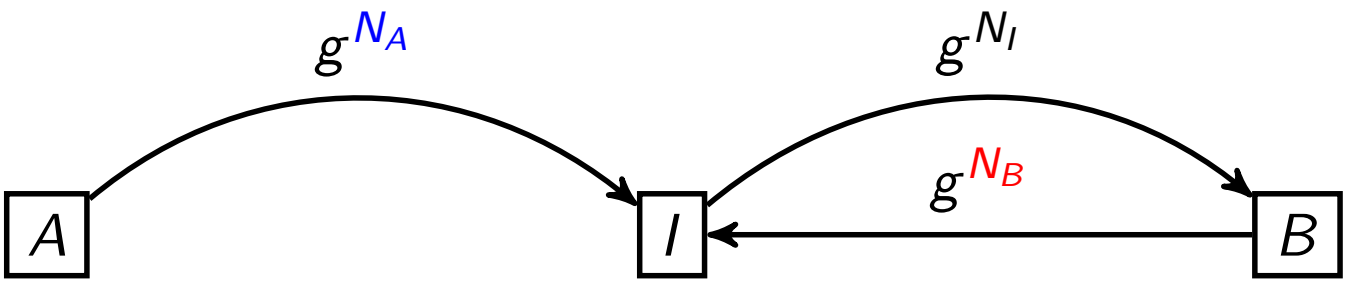
⑤ Il peut calculer des exponentielles (g^{N_I})

Exemple de failles et capacités de l'intrus

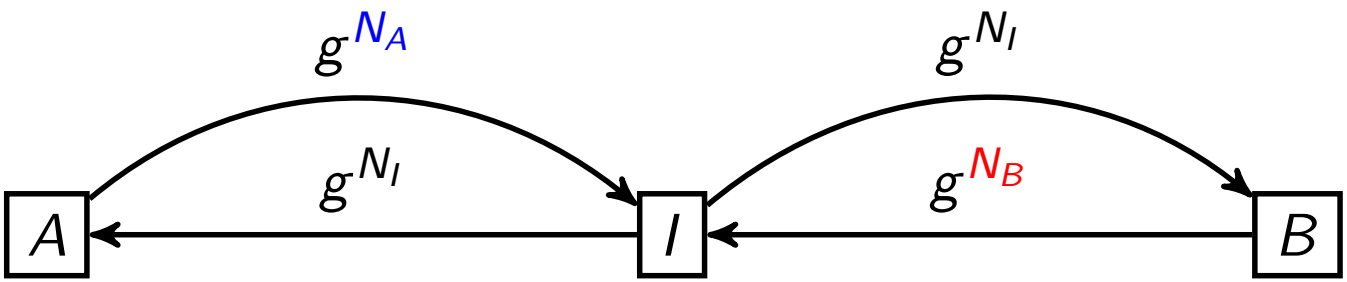


⑥ Il peut envoyer n'importe quel message construit à partir des connaissances qu'il a accumulé

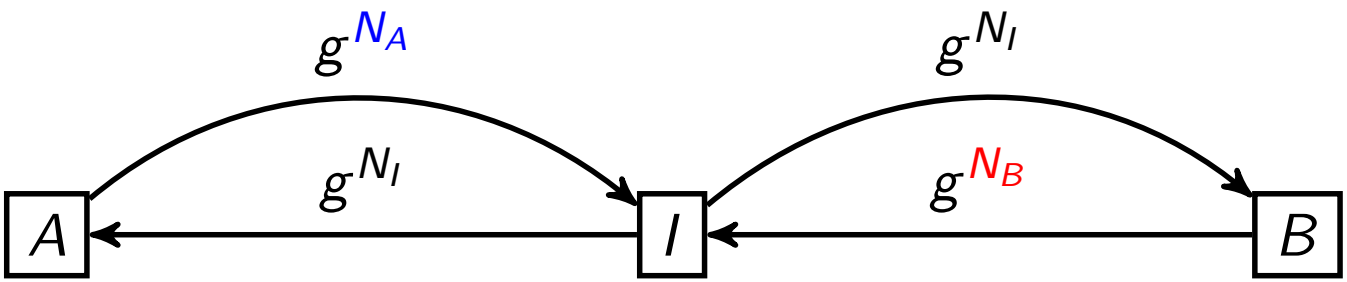
Exemple de failles et capacités de l'intrus



Exemple de failles et capacités de l'intrus

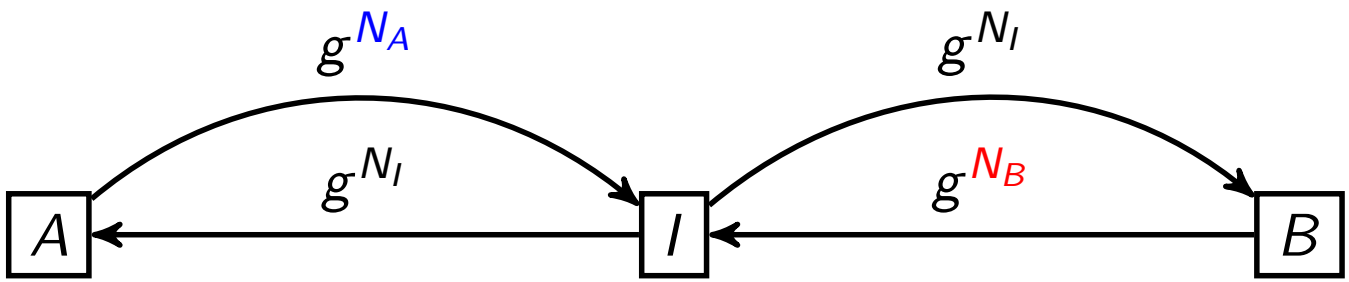


Exemple de failles et capacités de l'intrus



$K_1 = (g^{N_A})^{N_I}$

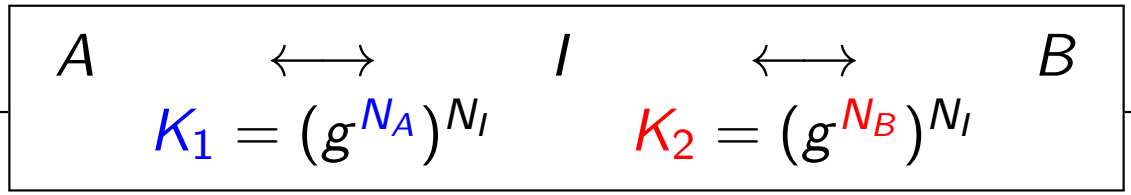
Exemple de failles et capacités de l'intrus



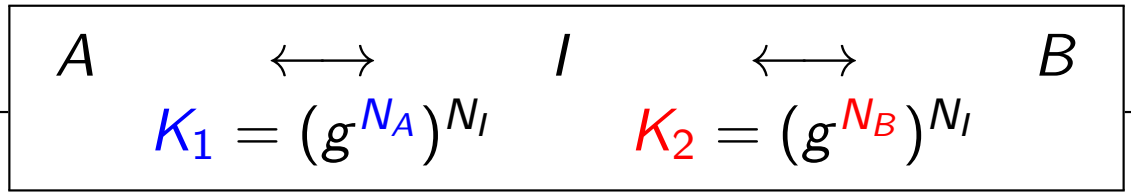
$K_1 = (g^{N_A})^{N_I}$ $K_2 = (g^{N_B})^{N_I}$

⑦ L'intrus peut exécuter plusieurs sessions entrelacées d'un même protocole

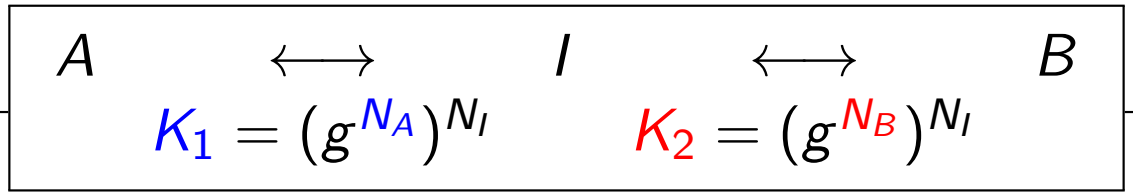
Exemple de faille et capacités de l'intrus



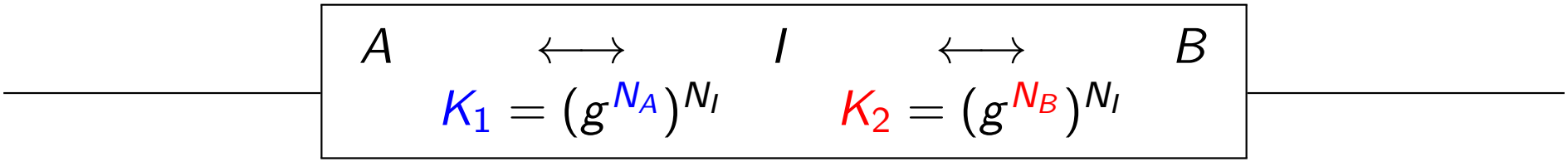
Exemple de faille et capacités de l'intrus

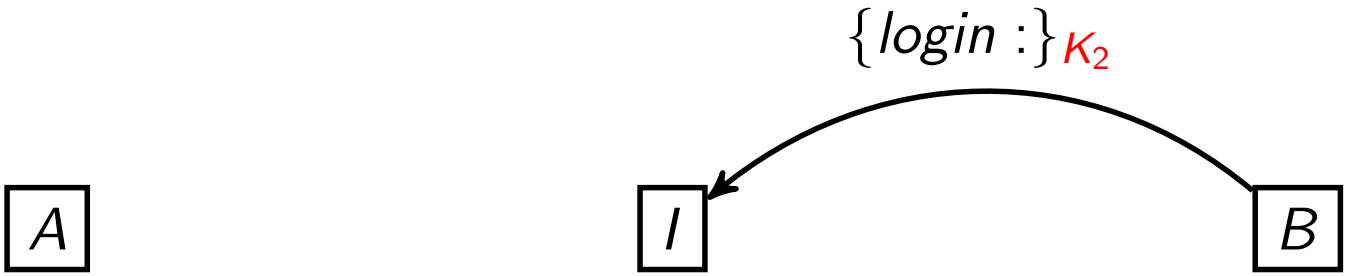


Exemple de faille et capacités de l'intrus



Exemple de faille et capacités de l'intrus

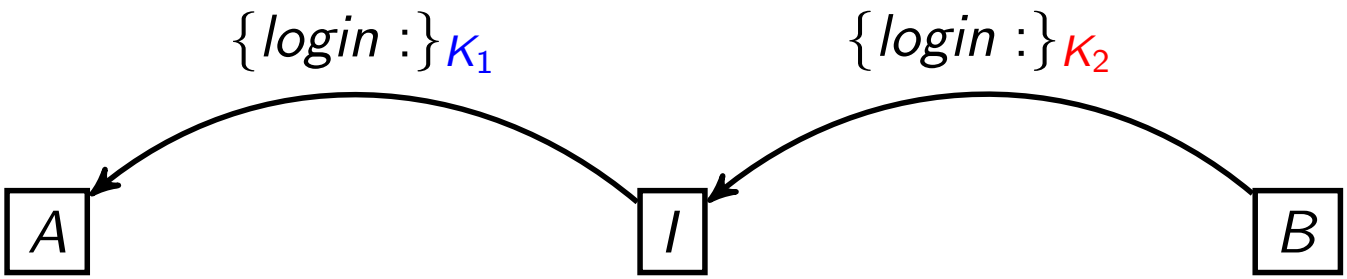




⑧ L'intrus peut déchiffrer un message chiffré s'il a la clé inverse

Exemple de faille et capacités de l'intrus

$$A \quad \longleftrightarrow \quad I \quad \longleftrightarrow \quad B$$
$$K_1 = (g^{N_A})^{N_I} \qquad K_2 = (g^{N_B})^{N_I}$$

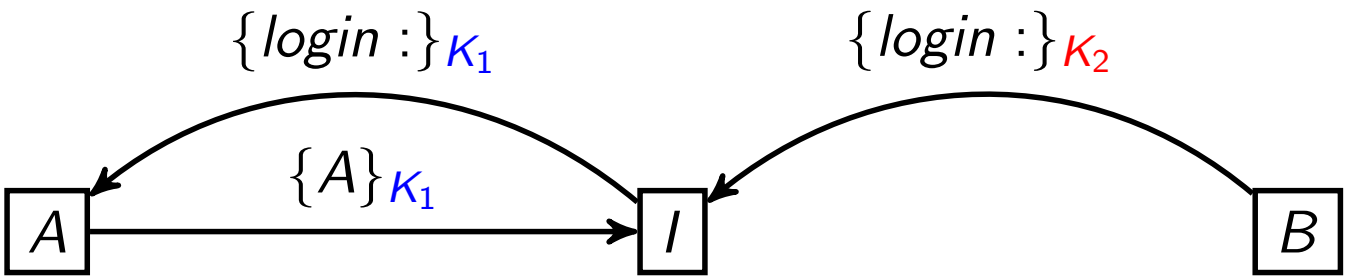


⑨ L'intrus peut *chiffrer* s'il a la clé

Exemple de faille et capacités de l'intrus

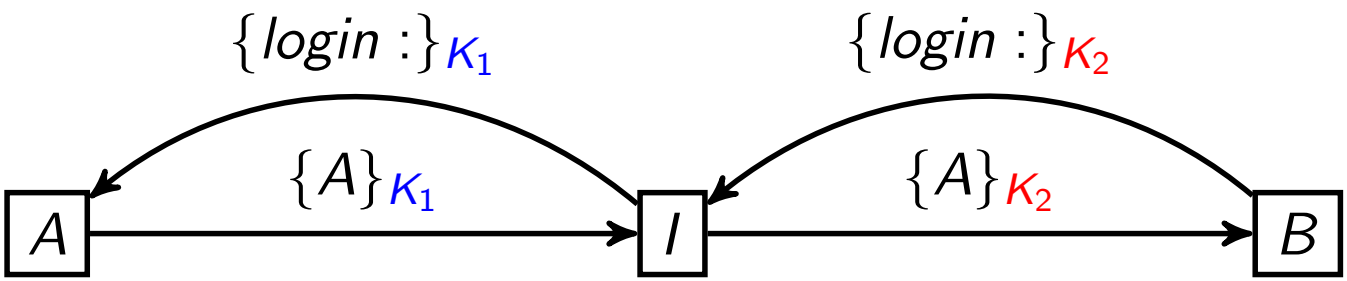
$$A \longleftrightarrow I \longleftrightarrow B$$

$$K_1 = (g^{N_A})^{N_I} \qquad K_2 = (g^{N_B})^{N_I}$$



Exemple de faille et capacités de l'intrus

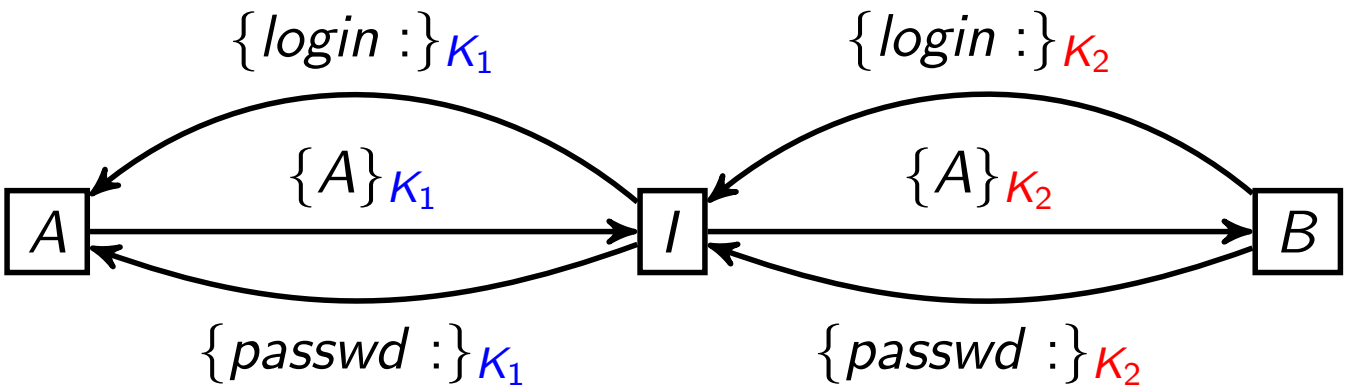
$$A \longleftrightarrow I \longleftrightarrow B$$
$$K_1 = (g^{N_A})^{N_I} \qquad K_2 = (g^{N_B})^{N_I}$$



Exemple de faille et capacités de l'intrus

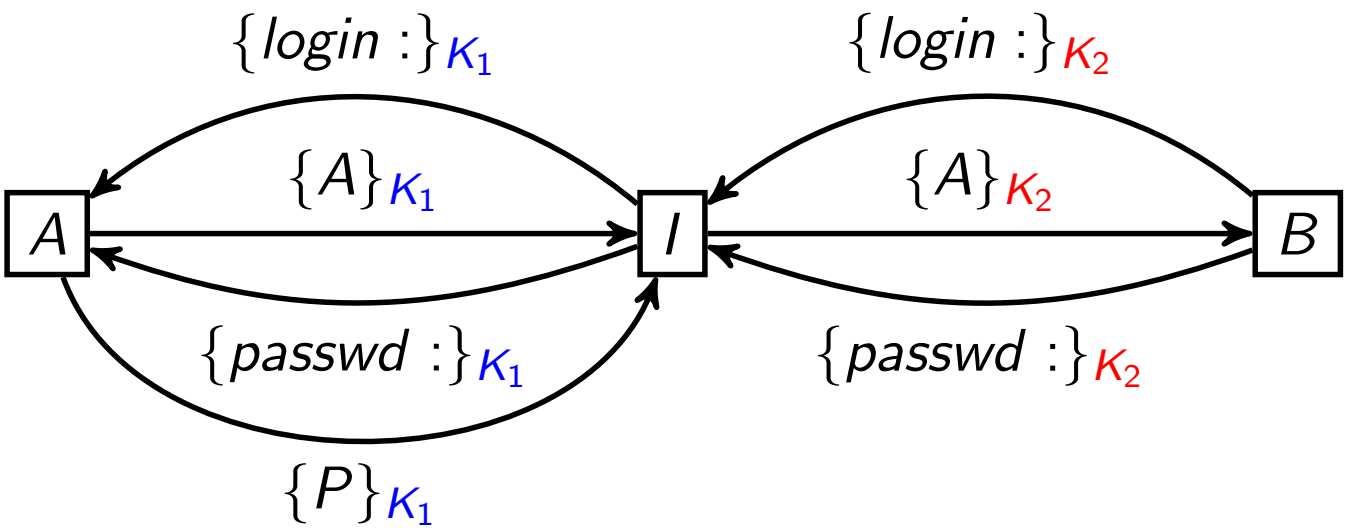
$$A \longleftrightarrow I \longleftrightarrow B$$

$$K_1 = (g^{N_A})^{N_I} \qquad K_2 = (g^{N_B})^{N_I}$$



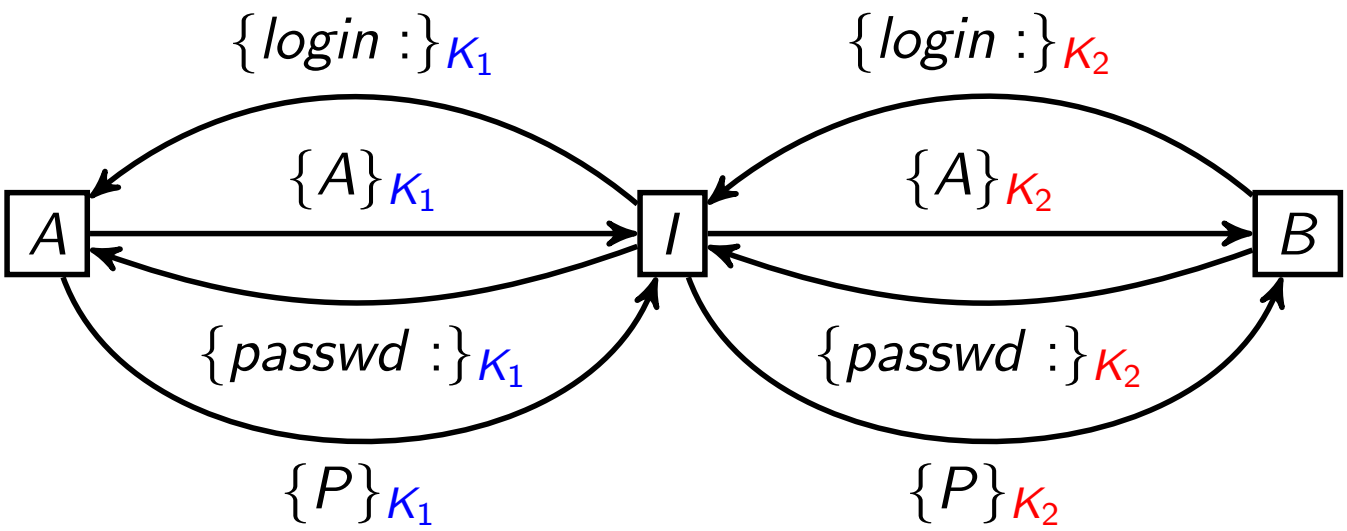
Exemple de faille et capacités de l'intrus

$$A \longleftrightarrow I \longleftrightarrow B$$
$$K_1 = (g^{N_A})^{N_I} \qquad K_2 = (g^{N_B})^{N_I}$$



Exemple de faille et capacités de l'intrus

$$A \longleftrightarrow I \longleftrightarrow B$$
$$K_1 = (g^{N_A})^{N_I} \qquad K_2 = (g^{N_B})^{N_I}$$



Les capacités de l'intrus

① L'intrus ne fait pas de cryptanalyse

② L'intrus peut intercepter/bloquer tous les messages

③ L'intrus connaît les données publiques (g)

④ Il peut générer des valeurs aléatoires (N_I)

⑤ Il peut calculer des exponentielles (g^{N_I})

⑥ Il peut envoyer n'importe quel message construit à partir des connaissances qu'il a accumulé

⑦ L'intrus peut exécuter plusieurs sessions entrelacées

⑧ L'intrus peut déchiffrer s'il a la clé inverse

⑨ L'intrus peut *chiffrer* s'il a la clé

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow S : A, B, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_B\}_{K_S^{-1}}$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow S : A, B, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_B\}_{K_S^{-1}}$

Z peut convaincre A que la clé publique de B est K_Z !

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow S : A, B, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_B\}_{K_S^{-1}}$

Z peut convaincre A que la clé publique de B est K_Z !

1. $A \hookrightarrow S : A, B, N_A$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow S : A, B, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_B\}_{K_S^{-1}}$

Z peut convaincre A que la clé publique de B est K_Z !

1. $A \hookrightarrow S : A, B, N_A$

1_Z. $Z(A) \hookrightarrow S : A, Z, N_A$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow S : A, B, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_B\}_{K_S^{-1}}$

Z peut convaincre A que la clé publique de B est K_Z !

1. $A \hookrightarrow S : A, B, N_A$
1_Z. $Z(A) \hookrightarrow S : A, Z, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_Z\}_{K_S^{-1}}$

A attribue K_Z à B !

Authentification invalide de K_Z par A

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow S : A, B, N_A$
1_Z. $Z(A) \hookrightarrow S : A, Z, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_Z\}_{K_S^{-1}}$

Manipulations de Z nécessaires pour cette attaque :

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow S : A, B, N_A$
1_Z. $Z(A) \hookrightarrow S : A, Z, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_Z\}_{K_S^{-1}}$

Manipulations de Z nécessaires pour cette attaque :

- lire, effacer, stocker tous les messages

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow S : A, B, N_A$
1_Z. $Z(A) \hookrightarrow S : A, Z, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_Z\}_{K_S^{-1}}$

Manipulations de Z nécessaires pour cette attaque :

- lire, effacer, stocker tous les messages
- Z peut jouer le rôle d'un autre acteur

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow S : A, B, N_A$
1_Z. $Z(A) \hookrightarrow S : A, Z, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_Z\}_{K_S^{-1}}$

Manipulations de Z nécessaires pour cette attaque :

- lire, effacer, stocker tous les messages
- Z peut jouer le rôle d'un autre acteur
- démonter les messages

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow S : A, B, N_A$
1_Z. $Z(A) \hookrightarrow S : A, Z, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_Z\}_{K_S^{-1}}$

Manipulations de Z nécessaires pour cette attaque :

- lire, effacer, stocker tous les messages
- Z peut jouer le rôle d'un autre acteur
- démonter les messages
- stocker les données obtenues

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow S : A, B, N_A$
1_Z. $Z(A) \hookrightarrow S : A, Z, N_A$
2. $S \hookrightarrow A : S, \{S, A, N_A, K_Z\}_{K_S^{-1}}$

Manipulations de Z nécessaires pour cette attaque :

- lire, effacer, stocker tous les messages
- Z peut jouer le rôle d'un autre acteur
- démonter les messages
- stocker les données obtenues
- construire des messages à partir des données obtenues

Exemple de faille et capacités de l'intrus

Needham-Schroeder 1978 (Authentification protocol)

Exemple de faille et capacités de l'intrus

Needham-Schroeder 1978 (Authentification protocol)

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$

Exemple de faille et capacités de l'intrus

Needham-Schroeder 1978 (Authentification protocol)

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$

Exemple de faille et capacités de l'intrus

Needham-Schroeder 1978 (Authentification protocol)

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow B : \{N_B\}_{K_B}$

Exemple de faille et capacités de l'intrus

Needham-Schroeder 1978 (Authentification protocol)

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$ N_B attribué à B
3. $A \hookrightarrow B : \{N_B\}_{K_B}$

Exemple de faille et capacités de l'intrus

Needham-Schroeder 1978 (Authentification protocol)

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$ N_B attribué à B
3. $A \hookrightarrow B : \{N_B\}_{K_B}$ N_A attribué à A

Exemple de faille et capacités de l'intrus

Needham-Schroeder 1978 (Authentification protocol)

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow B : \{N_B\}_{K_B}$

N_B attribué à B

N_A attribué à A

N_A et N_B connus de A et B seuls

Exemple de faille et capacités de l'intrus

Needham-Schroeder 1978 (Authentification protocol)

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow B : \{N_B\}_{K_B}$

N_B attribué à B

N_A attribué à A

N_A et N_B connus de A et B seuls

Protocole prouvé correct (preuves manuelles)

Exemple de faille et capacités de l'intrus

Needham-Schroeder 1978 (Authentification protocol)

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow B : \{N_B\}_{K_B}$

N_B attribué à B
N_A attribué à A
N_A et N_B connus de A et B seuls

Protocole prouvé correct (preuves manuelles)

Faille trouvée par Lowe en 1995 avec des méthodes formelles (model-checking)

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow B : \{N_B\}_{K_B}$

N_B attribué à B

N_A attribué à A

N_A et N_B connus de A et B seuls

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow B : \{N_B\}_{K_B}$

N_B attribué à B
N_A attribué à A
N_A et N_B connus de A et B seuls

---

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow B : \{N_B\}_{K_B}$

N_B attribué à B
N_A attribué à A
N_A et N_B connus de A et B seuls

---

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$
1_Z. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow B : \{N_B\}_{K_B}$

N_B attribué à B
N_A attribué à A
N_A et N_B connus de A et B seuls

---

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$
1_Z. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow B : \{N_B\}_{K_B}$

N_B attribué à B
N_A attribué à A
N_A et N_B connus de A et B seuls

---

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$
1_Z. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow Z : \{N_B\}_{K_Z}$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow B : \{N_B\}_{K_B}$

N_B attribué à B
N_A attribué à A
N_A et N_B connus de A et B seuls

---

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$
1_Z. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow Z : \{N_B\}_{K_Z}$
3_Z. $Z(A) \hookrightarrow B : \{N_B\}_{K_B}$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$ — N_B attribué à B
3. $A \hookrightarrow B : \{N_B\}_{K_B}$ — N_A attribué à A

N_A et N_B connus de A et B seuls

---

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$ — Z obtient N_A
- 1_Z. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow Z : \{N_B\}_{K_Z}$
- 3_Z. $Z(A) \hookrightarrow B : \{N_B\}_{K_B}$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$ — N_B attribué à B
3. $A \hookrightarrow B : \{N_B\}_{K_B}$ — N_A attribué à A

N_A et N_B connus de A et B seuls

---

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$ — Z obtient N_A
1_Z. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$ — N_B attribué à Z
3. $A \hookrightarrow Z : \{N_B\}_{K_Z}$
3_Z. $Z(A) \hookrightarrow B : \{N_B\}_{K_B}$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$ — N_B attribué à B
3. $A \hookrightarrow B : \{N_B\}_{K_B}$ — N_A attribué à A

N_A et N_B connus de A et B seuls

---

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$ — Z obtient N_A
1$_Z$. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$ — N_B attribué à Z
3. $A \hookrightarrow Z : \{N_B\}_{K_Z}$ — Z obtient N_B
3$_Z$. $Z(A) \hookrightarrow B : \{N_B\}_{K_B}$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$ — N_B attribué à B
3. $A \hookrightarrow B : \{N_B\}_{K_B}$ — N_A attribué à A

N_A et N_B connus de A et B seuls

---

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$ — Z obtient N_A
1$_Z$. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$ — N_B attribué à Z
3. $A \hookrightarrow Z : \{N_B\}_{K_Z}$ — Z obtient N_B
3$_Z$. $Z(A) \hookrightarrow B : \{N_B\}_{K_B}$ — N_A attribué à A

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$ — N_B attribué à B
3. $A \hookrightarrow B : \{N_B\}_{K_B}$ — N_A attribué à A

N_A et N_B connus de A et B seuls

---

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$ — Z obtient N_A
1_Z. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$ — N_B attribué à Z
3. $A \hookrightarrow Z : \{N_B\}_{K_Z}$ — Z obtient N_B
3_Z. $Z(A) \hookrightarrow B : \{N_B\}_{K_B}$ — N_A attribué à A

Invalide secret de (N_A, N_B) et authentification $A \leftrightarrow Z \leftrightarrow B$

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$
1_Z. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow Z : \{N_B\}_{K_Z}$
3_Z. $Z(A) \hookrightarrow B : \{N_B\}_{K_B}$

Manipulations de Z nécessaires à cette attaque :

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$
1_Z. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$

Manipulations de Z nécessaires à cette attaque :

- lancer des sessions du protocole en parallèle

Exemple de faille et capacités de l'intrus

3. $A \hookrightarrow Z : \{N_B\}_{K_Z}$
3_Z. $Z(A) \hookrightarrow B : \{N_B\}_{K_B}$

Manipulations de Z nécessaires à cette attaque :

- lancer des sessions du protocole en parallèle
- déchiffrer les messages s'il a la clé : K_Z^{-1}

Exemple de faille et capacités de l'intrus

$3_Z.\quad Z(A) \hookrightarrow B : \{N_B\}_{K_B}$

Manipulations de Z nécessaires à cette attaque :

- lancer des sessions du protocole en parallèle
- déchiffrer les messages s'il a la clé : K_Z^{-1}
- chiffrer les messages avec ses clés : K_B

Exemple de faille et capacités de l'intrus

1. $A \hookrightarrow Z : \{N_A, A\}_{K_Z}$
1_Z. $Z(A) \hookrightarrow B : \{N_A, A\}_{K_B}$
2. $B \hookrightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \hookrightarrow Z : \{N_B\}_{K_Z}$
3_Z. $Z(A) \hookrightarrow B : \{N_B\}_{K_B}$

Manipulations de Z nécessaires à cette attaque :

- lancer des sessions du protocole en parallèle
- déchiffrer les messages s'il a la clé : K_Z^{-1}
- chiffrer les messages avec ses clés : K_B

Vérification logique : hypothèses (I)

Par défaut nos hypothèses de vérification seront :

- la cryptographie est sûre
  - impossible de déchiffrer un texte chiffré sans la clé
  - impossible de deviner une clé ou un nonce
- nb d'acteurs honnêtes/malhonnêtes, non borné
- nombre de sessions du protocole, non borné
- le protocole peut être utilisé dans des sessions //

Vérification logique : hypothèses (II)

L'intrus a le contrôle total du réseau

- il connait toutes les données publiques des agents
- il peut disposer des privilèges/clés des agents malhonnêtes
- il peut lire, mémoriser, effacer tous les messages
- il peut construire et envoyer des messages
- il peut composer/décomposer un message
- il peut chiffrer/déchiffrer un message s'il a la clé

$\Rightarrow$ Modèle de Dolev-Yao : réseau = intrus

Exemples de recherche d'attaques

Soient les 3 protocoles suivants :

1. $A \hookrightarrow B : \{A, N_A\}_{K_B}$

---

1. $A \hookrightarrow B : \{A, N_A\}_{K_{AB}}$

---

1. $A \hookrightarrow B : \{A, N_A\}_{K_A^{-1}}$

Sous les hypothèses de vérification logique (p67/68)

pour chaque protocole, dites s'il satisfait la propriété ou donnez une attaque.

- N_A est secret ?
- B peut authentifier N_A ?
- B peut authentifier A ?

Exemple de recherche d'attaques (II)

1. $A \hookrightarrow S : B, \{K_{AS}\}_{K_S}$
2. $S \hookrightarrow B : A$
3. $B \hookrightarrow S : A, \{K_{AB}\}_{K_S}$ (TMN)
4. $S \hookrightarrow A : B, \{K_{AB}\}_{K_{AS}}$
5. $A \hookrightarrow B : \{m\}_{K_{AB}}$

A-t-on les propriétés attendues suivantes :

- K_{AS}, K_{AB}, m sont secrets ?

Exemple de recherche d'attaques (II)

1. $A \hookrightarrow S : B, \{K_{AS}\}_{K_S}$
2. $S \hookrightarrow B : A$
3. $B \hookrightarrow S : A, \{K_{AB}\}_{K_S}$ (TMN)
4. $S \hookrightarrow A : B, \{K_{AB}\}_{K_{AS}}$
5. $A \hookrightarrow B : \{m\}_{K_{AB}}$

A-t-on les propriétés attendues suivantes :

- K_{AS}, K_{AB}, m sont secrets ?

Donnez les 2 attaques permettant respectivement de :

- se faire passer pour B et de réccupérer m
- se faire passer pour A et de réccupérer K_{AB} et m

Exemple de recherche d'attaques (II)

1. $A \hookrightarrow S : B, \{K_{AS}\}_{K_S}$
2. $S \hookrightarrow B : A$
3. $B \hookrightarrow S : A, \{K_{AB}\}_{K_S}$ (TMN)
4. $S \hookrightarrow A : B, \{K_{AB}\}_{K_{AS}}$
5. $A \hookrightarrow B : \{m\}_{K_{AB}}$

A-t-on les propriétés attendues suivantes :

- K_{AS}, K_{AB}, m sont secrets ?
- S peut authentifier K_{AS} et K_{AB} ?

Donnez les 2 attaques permettant respectivement de :

- se faire passer pour B et de réccupérer m
- se faire passer pour A et de réccupérer K_{AB} et m

Exemple de recherche d'attaques (III)

1. $A \hookrightarrow B : \{N_A\}_{K_{AB}}$
2. $B \hookrightarrow A : \{N_B\}_{K_{AB}}$
3. $A \hookrightarrow B : \{N_A, N_B\}_{K_{AB}}$
4. $B \hookrightarrow A : \{N_B, N_A\}_{K_{AB}}$

- Les nonces N_A et N_B sont secrets ?
- A peut authentifier N_B ?
- B peut authentifier N_A ?
- A peut authentifier B ?
- B peut authentifier A ?

Formalisation logique de Dolev-Yao

Formalisation logique de Dolev-Yao

- $I =$ connaissance initiale de l'intrus

Formalisation logique de Dolev-Yao

- I = connaissance initiale de l'intrus
  - les identités de tous les acteurs
  - toutes les clés publiques
  - des clés privées corrompues

Formalisation logique de Dolev-Yao

- I = connaissance initiale de l'intrus
  - les identités de tous les acteurs
  - toutes les clés publiques
  - des clés privées corrompues
- $\left.\begin{array}{l}\text{Manipulations de l'intrus}\\ \text{Exécution du protocole}\end{array}\right| = \text{déductions sur } I$

Formalisation logique de Dolev-Yao

- I = connaissance initiale de l'intrus
  - les identités de tous les acteurs
  - toutes les clés publiques
  - des clés privées corrompues
- $\left.\begin{array}{l}\text{Manipulations de l'intrus}\\ \text{Exécution du protocole}\end{array}\right| = \text{déductions sur } I$

Manipulations de l'intrus

Formalisation logique de Dolev-Yao

- I = connaissance initiale de l'intrus
  - les identités de tous les acteurs
  - toutes les clés publiques
  - des clés privées corrompues
- $\left.\begin{array}{l}\text{Manipulations de l'intrus}\\ \text{Exécution du protocole}\end{array}\right| = \text{déductions sur } I$

Manipulations de l'intrus

$$\frac{}{I, M \vdash M}$$

Formalisation logique de Dolev-Yao

- I= connaissance initiale de l'intrus
  - les identités de tous les acteurs
  - toutes les clés publiques
  - des clés privées corrompues
- Manipulations de l'intrus
Exécution du protocole $\Big|$ = déductions sur I

Manipulations de l'intrus

$$\frac{}{I, M \vdash M} \qquad \frac{I \vdash \{M\}_K \quad I \vdash K^{-1}}{I \vdash M}$$

Formalisation logique de Dolev-Yao

- I= connaissance initiale de l'intrus
  - les identités de tous les acteurs
  - toutes les clés publiques
  - des clés privées corrompues
- $\left.\begin{array}{l}\text{Manipulations de l'intrus}\\ \text{Exécution du protocole}\end{array}\right| = \text{déductions sur } I$

Manipulations de l'intrus

$$\frac{}{I, M \vdash M} \qquad \frac{I \vdash \{M\}_K \quad I \vdash K^{-1}}{I \vdash M} \qquad \frac{I \vdash M \quad I \vdash K}{I \vdash \{M\}_K}$$

Formalisation logique de Dolev-Yao

- I= connaissance initiale de l'intrus
  - les identités de tous les acteurs
  - toutes les clés publiques
  - des clés privées corrompues
- $\left.\begin{array}{l}\text{Manipulations de l'intrus}\\ \text{Exécution du protocole}\end{array}\right| = \text{déductions sur } I$

Manipulations de l'intrus

$$\frac{}{I, M \vdash M} \qquad \frac{I \vdash \{M\}_K \quad I \vdash K^{-1}}{I \vdash M} \qquad \frac{I \vdash M \quad I \vdash K}{I \vdash \{M\}_K}$$

$$\frac{I \vdash X, Y}{I \vdash X} \qquad \frac{I \vdash X, Y}{I \vdash Y}$$

Formalisation logique de Dolev-Yao

- I = connaissance initiale de l'intrus
  - les identités de tous les acteurs
  - toutes les clés publiques
  - des clés privées corrompues
- $\left.\begin{array}{l}\text{Manipulations de l'intrus}\\ \text{Exécution du protocole}\end{array}\right| = \text{déductions sur } I$

Manipulations de l'intrus

$$\frac{}{I, M \vdash M} \qquad \frac{I \vdash \{M\}_K \quad I \vdash K^{-1}}{I \vdash M} \qquad \frac{I \vdash M \quad I \vdash K}{I \vdash \{M\}_K}$$

$$\frac{I \vdash X, Y}{I \vdash X} \qquad \frac{I \vdash X, Y}{I \vdash Y} \qquad \frac{I \vdash X \quad I \vdash Y}{I \vdash X, Y}$$

Formalisation logique de Dolev-Yao

Exécution du protocole = règles supplémentaires

Formalisation logique de Dolev-Yao

Exécution du protocole = règles supplémentaires

Exemple : $\begin{cases} 1. & A \hookrightarrow B : g^{N_A} \\ 2. & B \hookrightarrow A : g^{N_B} \\ 3. & A \hookrightarrow B : \{s\}_K \quad \text{où} \quad K = (g^{N_B})^{N_A} \end{cases}$

Formalisation logique de Dolev-Yao

Exécution du protocole = règles supplémentaires

Exemple : $\begin{cases} 1. & A \hookrightarrow B : g^{N_A} \\ 2. & B \hookrightarrow A : g^{N_B} \\ 3. & A \hookrightarrow B : \{s\}_K \quad \text{où} \quad K = (g^{N_B})^{N_A} \end{cases}$

$$1.\ \frac{I \vdash a \quad I \vdash g}{I \vdash g^{N_a}}$$

Formalisation logique de Dolev-Yao

Exécution du protocole = règles supplémentaires

Exemple : $\begin{cases} 1. & A \hookrightarrow B : g^{N_A} \\ 2. & B \hookrightarrow A : g^{N_B} \\ 3. & A \hookrightarrow B : \{s\}_K \quad \text{où} \quad K = (g^{N_B})^{N_A} \end{cases}$

$$1.\frac{I \vdash a \quad I \vdash g}{I \vdash g^{N_a}} \qquad 2.\frac{I \vdash b \quad I \vdash X}{I \vdash g^{N_b}}$$

Formalisation logique de Dolev-Yao

Exécution du protocole = règles supplémentaires

Exemple : $\begin{cases} 1. & A \hookrightarrow B : g^{N_A} \\ 2. & B \hookrightarrow A : g^{N_B} \\ 3. & A \hookrightarrow B : \{s\}_K \quad \text{où} \quad K = (g^{N_B})^{N_A} \end{cases}$

$$1.\frac{I \vdash a \quad I \vdash g}{I \vdash g^{N_a}} \qquad 2.\frac{I \vdash b \quad I \vdash X}{I \vdash g^{N_b}} \qquad 3.\frac{I \vdash a \quad I \vdash Y}{I \vdash \{s\}_{Y^{N_a}}}$$

Formalisation logique de Dolev-Yao

$$1.\ \frac{I \vdash a \quad I \vdash g}{I \vdash g^{N_a}} \qquad 2.\ \frac{I \vdash b \quad I \vdash X}{I \vdash g^{N_b}} \qquad 3.\ \frac{I \vdash a \quad I \vdash Y}{I \vdash \{s\}_{Y^{N_a}}}$$

$$\frac{}{I, M \vdash M} \qquad \frac{I \vdash \{M\}_K \quad I \vdash K^{-1}}{I \vdash M} \qquad \frac{I \vdash M \quad I \vdash K}{I \vdash \{M\}_K}$$

$$\frac{I \vdash X, Y}{I \vdash X} \qquad \frac{I \vdash X, Y}{I \vdash Y} \qquad \frac{I \vdash X \quad I \vdash Y}{I \vdash X, Y}$$

$$a, b, g \vdash \{\{s\}_{g^{N_a}}\}_g$$

Formalisation logique de Dolev-Yao

$$1.\ \frac{I \vdash a \quad I \vdash g}{I \vdash g^{N_a}} \qquad 2.\ \frac{I \vdash b \quad I \vdash X}{I \vdash g^{N_b}} \qquad 3.\ \frac{I \vdash a \quad I \vdash Y}{I \vdash \{s\}_{Y^{N_a}}}$$

$$\frac{}{I, M \vdash M} \qquad \frac{I \vdash \{M\}_K \quad I \vdash K^{-1}}{I \vdash M} \qquad \frac{I \vdash M \quad I \vdash K}{I \vdash \{M\}_K}$$

$$\frac{I \vdash X, Y}{I \vdash X} \qquad \frac{I \vdash X, Y}{I \vdash Y} \qquad \frac{I \vdash X \quad I \vdash Y}{I \vdash X, Y}$$

$$\frac{a, b, g \vdash \{s\}_{g^{N_a}} \qquad a, b, g \vdash g}{a, b, g \vdash \{\{s\}_{g^{N_a}}\}_g}$$

Formalisation logique de Dolev-Yao

$$1.\ \frac{I \vdash a \quad I \vdash g}{I \vdash g^{N_a}} \qquad 2.\ \frac{I \vdash b \quad I \vdash X}{I \vdash g^{N_b}} \qquad 3.\ \frac{I \vdash a \quad I \vdash Y}{I \vdash \{s\}_{Y^{N_a}}}$$

$$\frac{}{I, M \vdash M} \qquad \frac{I \vdash \{M\}_K \quad I \vdash K^{-1}}{I \vdash M} \qquad \frac{I \vdash M \quad I \vdash K}{I \vdash \{M\}_K}$$

$$\frac{I \vdash X, Y}{I \vdash X} \qquad \frac{I \vdash X, Y}{I \vdash Y} \qquad \frac{I \vdash X \quad I \vdash Y}{I \vdash X, Y}$$

$$\frac{a, b, g \vdash \{s\}_{g^{N_a}} \qquad \dfrac{}{a, b, g \vdash g}}{a, b, g \vdash \{\{s\}_{g^{N_a}}\}_g}$$

Formalisation logique de Dolev-Yao

$$1.\ \frac{I \vdash a \quad I \vdash g}{I \vdash g^{N_a}} \qquad 2.\ \frac{I \vdash b \quad I \vdash X}{I \vdash g^{N_b}} \qquad 3.\ \frac{I \vdash a \quad I \vdash Y}{I \vdash \{s\}_{Y^{N_a}}}$$

$$\frac{}{I, M \vdash M} \qquad \frac{I \vdash \{M\}_K \quad I \vdash K^{-1}}{I \vdash M} \qquad \frac{I \vdash M \quad I \vdash K}{I \vdash \{M\}_K}$$

$$\frac{I \vdash X, Y}{I \vdash X} \qquad \frac{I \vdash X, Y}{I \vdash Y} \qquad \frac{I \vdash X \quad I \vdash Y}{I \vdash X, Y}$$

$$\frac{\dfrac{a, b, g \vdash a \qquad a, b, g \vdash g}{a, b, g \vdash \{s\}_{g^{N_a}}} \qquad \dfrac{}{a, b, g \vdash g}}{a, b, g \vdash \{\{s\}_{g^{N_a}}\}_g}$$

Formalisation logique de Dolev-Yao

$$1.\ \frac{I \vdash a \quad I \vdash g}{I \vdash g^{N_a}} \qquad 2.\ \frac{I \vdash b \quad I \vdash X}{I \vdash g^{N_b}} \qquad 3.\ \frac{I \vdash a \quad I \vdash Y}{I \vdash \{s\}_{Y^{N_a}}}$$

$$\frac{}{I, M \vdash M} \qquad \frac{I \vdash \{M\}_K \quad I \vdash K^{-1}}{I \vdash M} \qquad \frac{I \vdash M \quad I \vdash K}{I \vdash \{M\}_K}$$

$$\frac{I \vdash X, Y}{I \vdash X} \qquad \frac{I \vdash X, Y}{I \vdash Y} \qquad \frac{I \vdash X \quad I \vdash Y}{I \vdash X, Y}$$

$$\dfrac{\dfrac{a, b, g \vdash a \qquad \dfrac{}{a, b, g \vdash g}}{a, b, g \vdash \{s\}_{g^{N_a}}} \qquad \dfrac{}{a, b, g \vdash g}}{a, b, g \vdash \{\{s\}_{g^{N_a}}\}_g}$$

Formalisation logique de Dolev-Yao

$$1.\ \frac{I \vdash a \quad I \vdash g}{I \vdash g^{N_a}} \qquad 2.\ \frac{I \vdash b \quad I \vdash X}{I \vdash g^{N_b}} \qquad 3.\ \frac{I \vdash a \quad I \vdash Y}{I \vdash \{s\}_{Y^{N_a}}}$$

$$\frac{}{I, M \vdash M} \qquad \frac{I \vdash \{M\}_K \quad I \vdash K^{-1}}{I \vdash M} \qquad \frac{I \vdash M \quad I \vdash K}{I \vdash \{M\}_K}$$

$$\frac{I \vdash X, Y}{I \vdash X} \qquad \frac{I \vdash X, Y}{I \vdash Y} \qquad \frac{I \vdash X \quad I \vdash Y}{I \vdash X, Y}$$

$$\dfrac{\dfrac{\dfrac{}{a, b, g \vdash a} \quad \dfrac{}{a, b, g \vdash g}}{a, b, g \vdash \{s\}_{g^{N_a}}} \quad \dfrac{}{a, b, g \vdash g}}{a, b, g \vdash \{\{s\}_{g^{N_a}}\}_g}$$

Formalisation logique de Dolev-Yao

$$1.\ \frac{I \vdash a \quad I \vdash g}{I \vdash g^{N_a}} \qquad 2.\ \frac{I \vdash b \quad I \vdash X}{I \vdash g^{N_b}} \qquad 3.\ \frac{I \vdash a \quad I \vdash Y}{I \vdash \{s\}_{Y^{N_a}}}$$

$$\frac{}{I, M \vdash M} \qquad \frac{I \vdash \{M\}_K \quad I \vdash K^{-1}}{I \vdash M} \qquad \frac{I \vdash M \quad I \vdash K}{I \vdash \{M\}_K}$$

$$\frac{I \vdash X, Y}{I \vdash X} \qquad \frac{I \vdash X, Y}{I \vdash Y} \qquad \frac{I \vdash X \quad I \vdash Y}{I \vdash X, Y}$$

$$\dfrac{\dfrac{\dfrac{}{a, b, g \vdash a} \quad \dfrac{}{a, b, g \vdash g}}{a, b, g \vdash \{s\}_{g^{N_a}}} \quad \dfrac{}{a, b, g \vdash g}}{a, b, g \vdash \{\{s\}_{g^{N_a}}\}_g}$$

Abrégé en $a, b, g \vdash^2 \{\{s\}_{g^{N_a}}\}_g$

Vérification logique sur Dolev-Yao

Preuve que s est secret $= \boxed{\forall n : I \not\vdash^{n} s}$

Vérification logique sur Dolev-Yao

Preuve que s est secret $= \boxed{\forall n : I \not\vdash^{n} s}$

- Vérification complexe : 3 dimensions non bornées

Vérification logique sur Dolev-Yao

Preuve que s est secret $= \boxed{\forall n : I \not\vdash^{n} s}$

- Vérification complexe : 3 dimensions non bornées
  - Nombre d'agents exécutant le protocole en //

Vérification logique sur Dolev-Yao

Preuve que s est secret $= \boxed{\forall n : I \not\vdash^n s}$

- Vérification complexe : 3 dimensions non bornées
  - Nombre d'agents exécutant le protocole en //
  - Nombre de sessions pour chaque agent

Vérification logique sur Dolev-Yao

Preuve que s est secret = $\boxed{\forall n : I \not\vdash^n s}$

- Vérification complexe : 3 dimensions non bornées
  - Nombre d'agents exécutant le protocole en //
  - Nombre de sessions pour chaque agent
  - Déductions et constructions de l'intrus sur sa connaissance

Vérification logique sur Dolev-Yao

- Trois grandes catégories de travaux et d'outils :

Vérification logique sur Dolev-Yao

- Trois grandes catégories de travaux et d'outils :
  - Vérification finie automatique (model-checking)
    Pour nb d'agents fixé et k fixé : $\boxed{\forall n \leq k : I \not\vdash^n s}$

Vérification logique sur Dolev-Yao

- Trois grandes catégories de travaux et d'outils :
  - Vérification finie automatique (model-checking)
    Pour nb d'agents fixé et k fixé : $\boxed{\forall n \leq k : I \not\vdash^n s}$

    Détection d'attaques : $\boxed{\exists k : I \vdash^k s}$

Vérification logique sur Dolev-Yao

- Trois grandes catégories de travaux et d'outils :
  - Vérification finie automatique (model-checking)
    Pour nb d'agents fixé et k fixé : $\boxed{\forall n \leq k : I \not\models^n s}$
    Détection d'attaques : $\boxed{\exists k : I \vdash^k s}$
  - Vérification semi-automatique (preuve assistée)
    Par induction sur nb d'agents et n : $\boxed{\forall n : I \not\models^n s}$

Vérification logique sur Dolev-Yao

- Trois grandes catégories de travaux et d'outils :
  - Vérification finie automatique (model-checking)
    Pour nb d'agents fixé et k fixé : $\boxed{\forall n \leq k : I \not\vdash^n s}$
    Détection d'attaques : $\boxed{\exists k : I \vdash^k s}$
  - Vérification semi-automatique (preuve assistée)
    Par induction sur nb d'agents et n : $\boxed{\forall n : I \not\vdash^n s}$
  - Vérification automatique par approximation
    $I^\sharp \supseteq I$ et $\vdash^\sharp \supseteq \vdash$ telles que $\boxed{\forall n : I^\sharp \not\vdash^{\sharp n} s}$

Vérification par approximation

Idée= construire le langage des termes déductibles =*Store*

Vérification par approximation

Idée= construire le langage des termes déductibles =*Store*

$$\dfrac{\dfrac{\dfrac{}{M, K \vdash M} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{M\}_K} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{\{M\}_K\}_K}$$

Vérification par approximation

Idée= construire le langage des termes déductibles =*Store*

$$\dfrac{\dfrac{\dfrac{}{M, K \vdash M} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{M\}_K} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{\{M\}_K\}_K}$$

$$Store = \{M, K, \{M\}_K, \{\{M\}_K\}_K, \ldots\} \;=\; \{T \mid \forall n : I \vdash^n T\}$$

Vérification par approximation

Idée= construire le langage des termes déductibles =*Store*

$$\dfrac{\dfrac{\dfrac{}{M, K \vdash M} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{M\}_K} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{\{M\}_K\}_K}$$

$$Store = \{M, K, \{M\}_K, \{\{M\}_K\}_K, \ldots\} \;=\; \{T \mid \forall n : I \vdash^n T\}$$

Construction *automate approximation* $\mathcal{A}$

Vérification par approximation

Idée= construire le langage des termes déductibles =*Store*

$$\dfrac{\dfrac{\dfrac{}{M, K \vdash M} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{M\}_K} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{\{M\}_K\}_K}$$

$$Store = \{M, K, \{M\}_K, \{\{M\}_K\}_K, \ldots\} \;=\; \{T \mid \forall n : I \vdash^n T\}$$

Construction *automate approximation* $\mathcal{A}$ t.q. $L(\mathcal{A}) \supseteq Store$

Vérification par approximation

Idée= construire le langage des termes déductibles =*Store*

$$\dfrac{\dfrac{\dfrac{}{M, K \vdash M} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{M\}_K} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{\{M\}_K\}_K}$$

$$Store = \{M, K, \{M\}_K, \{\{M\}_K\}_K, \ldots\} \;=\; \{T \mid \forall n : I \vdash^n T\}$$

Construction *automate approximation* $\mathcal{A}$ t.q. $L(\mathcal{A}) \supseteq Store$

Secret de $s \equiv$ $\quad s \notin L(\mathcal{A})$

Vérification par approximation

Idée= construire le langage des termes déductibles =$Store$

$$\dfrac{\dfrac{\dfrac{}{M, K \vdash M} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{M\}_K} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{\{M\}_K\}_K}$$

$$Store = \{M, K, \{M\}_K, \{\{M\}_K\}_K, \ldots\} \;=\; \{T \mid \forall n : I \vdash^n T\}$$

Construction *automate approximation* $\mathcal{A}$ t.q. $L(\mathcal{A}) \supseteq Store$

$$\boxed{\text{Secret de } s \equiv \quad s \notin L(\mathcal{A})} \quad \Rightarrow s \notin Store$$

Vérification par approximation

Idée= construire le langage des termes déductibles $=Store$

$$\dfrac{\dfrac{\dfrac{}{M, K \vdash M} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{M\}_K} \qquad \dfrac{}{M, K \vdash K}}{M, K \vdash \{\{M\}_K\}_K}$$

$$Store = \{M, K, \{M\}_K, \{\{M\}_K\}_K, \ldots\} \;=\; \{T \mid \forall n : I \vdash^n T\}$$

Construction *automate approximation* $\mathcal{A}$ t.q. $L(\mathcal{A}) \supseteq Store$

$$\boxed{\text{Secret de } s \equiv \quad s \notin L(\mathcal{A})} \quad \Rightarrow s \notin Store \quad \Rightarrow \forall n : I \not\vdash^n s$$

Langages de spécification de protocoles

Formats proches des notations standards

- Casper [Lowe]
- Capsl [Mitchell]
- Eva, etc. [Jacquemard, Le Métayer]

Casper	Capsl	Eva
0. ->A: B 1. A->B: {na, A}{PK(B)} 2. B->A: {na, nb}{PK(A)} 3. A->B: {nb}{PK(B)}	A->B: {A,Na}pk(B); B->A: {Na,Nb}pk(A); A->B: {Nb}pk(B);	1. A->B: {Na, A}KPb 2. B->A: {Na, Nb}KPa 3. A->B: {Nb}KPb
Secret(A, na, [B]) Secret(B, nb, [A]) Agreement(A,B,[na,nb]) Agreement(B,A,[na,nb])	SECRET Na; SECRET Nb; PRECEDES A: B | Na; PRECEDES B: A | Nb;	Claim Agreement(A,B,Na,Na) Agreement(A,B,Nb,Nb)

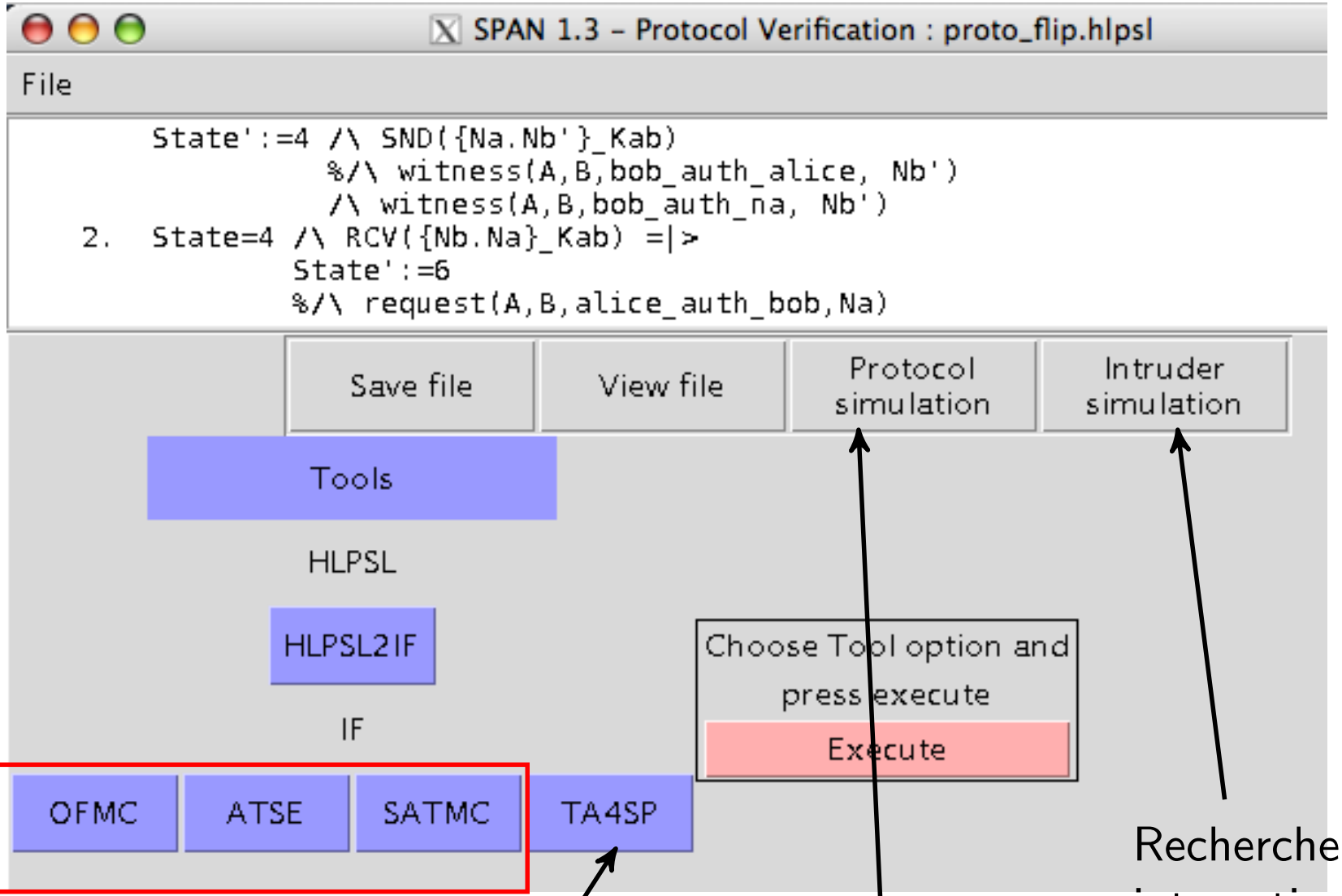
Les langages de spécification de protocoles

Formats inspirés des langages de processus

- ProVerif [Blanchet]
- Prouvé [Kremer, Laknech, Treinen]
- AVISPA (HLPSL) [Armando, et col.]

Alice	Bob
role alice (A,B: agent, ...) local State: nat, Na,Nb: text init State:= 0 transition 0. State=0 /\ RCV(start) =|> State':= 2 /\ Na' := new() /\ SND({Na'.A}_Kb) /\ secret(Na',na,{A,B}) 2. State=2 /\ RCV({Na.Nb'}_Ka) =|> State':= 4 /\ SND({Nb'}_Kb) end role	role bob(A, B: agent, ...) local State : nat, Na,Nb: text init State:= 1 transition 1. State= 1 /\ RCV({Na'.A}_Kb) =|> State':= 3 /\ Nb' := new() /\ SND({Na'.Nb'}_Ka) /\ secret(Nb',nb,{A,B}) 3. State= 3 /\ RCV({Nb'}_Kb) =|> State':= 5 /\ end role

AVISPA+SPAN


SPAN 1.3 – Protocol Verification : proto_flip.hlpsl
File
State':=4 /\ SND({Na.Nb'}_Kab)
%/\ witness(A,B,bob_auth_alice, Nb')
/\ witness(A,B,bob_auth_na, Nb')
2. State=4 /\ RCV({Nb.Na}_Kab) =|>
State':=6
%/\ request(A,B,alice_auth_bob,Na)
Save file
View file
Protocol simulation
Intruder simulation
Tools
HLPSL
HLPSL2IF
IF
Choose Tool option and press execute
Execute
OFMC
ATSE
SATMC
TA4SP
Model-checkers
Approximation
Simulation
Recherche interactive d'attaque

Démo AVISPA+SPAN

- Spécification et vérification des protocoles
  1. $B \hookrightarrow A : \{B, Secret\}_{K_B^{-1}}$

---

  1. $B \hookrightarrow A : \{B, Secret\}_{K_A}$

---

  1. $A \hookrightarrow B : \{A, N_A\}_{K_B}$
  2. $B \hookrightarrow A : \{A, B, N_A\}_{K_A}$

---

  1. $B \hookrightarrow A : \{B, Secret\}_{K_{AB}}$

- Diffie-Hellman
- A vous : `http://www.irisa.fr/celtique/genet/span`

Conclusion

Conclusion

- L'utilisation de la cryptographie et des protocoles cryptographiques se généralise dans notre société

Conclusion

- L'utilisation de la cryptographie et des protocoles cryptographiques se généralise dans notre société
- La cryptographie est relativement solide même si les garanties qu'elle offre sont généralement empiriques

Conclusion

- L'utilisation de la cryptographie et des protocoles cryptographiques se généralise dans notre société
- La cryptographie est relativement solide même si les garanties qu'elle offre sont généralement empiriques
- Les protocoles cryptographiques sont sujets à des failles logiques redoutables

Conclusion

- L'utilisation de la cryptographie et des protocoles cryptographiques se généralise dans notre société
- La cryptographie est relativement solide même si les garanties qu'elle offre sont généralement empiriques
- Les protocoles cryptographiques sont sujets à des failles logiques redoutables
- La vérification formelle des protocoles cryptographiques est en plein essor depuis 15 ans (depuis la faille trouvée par G. Lowe)

Vérification des protocoles crypto

Actuellement

Vérification des protocoles crypto

Actuellement

- Nombreux modèles (Processus, Réécriture, Clauses,...)

Vérification des protocoles crypto

Actuellement

- Nombreux modèles (Processus, Réécriture, Clauses,...)
- Nombreuses techniques de vérification

Vérification des protocoles crypto

Actuellement

- Nombreux modèles (Processus, Réécriture, Clauses,...)
- Nombreuses techniques de vérification
  - Énumération pour des cas d'études finis

Vérification des protocoles crypto

Actuellement

- Nombreux modèles (Processus, Réécriture, Clauses,...)
- Nombreuses techniques de vérification
  - Énumération pour des cas d'études finis
  - Procédures de décision pour des sous-classes

Vérification des protocoles crypto

Actuellement

- Nombreux modèles (Processus, Réécriture, Clauses,...)
- Nombreuses techniques de vérification
  - Énumération pour des cas d'études finis
  - Procédures de décision pour des sous-classes
  - Interprétation abstraite et approximation

Vérification des protocoles crypto

Actuellement

- Nombreux modèles (Processus, Réécriture, Clauses,...)
- Nombreuses techniques de vérification
  - Énumération pour des cas d'études finis
  - Procédures de décision pour des sous-classes
  - Interprétation abstraite et approximation
  - Démonstration assistée

Vérification des protocoles crypto

Actuellement

- Nombreux modèles (Processus, Réécriture, Clauses,...)
- Nombreuses techniques de vérification
  - Énumération pour des cas d'études finis
  - Procédures de décision pour des sous-classes
  - Interprétation abstraite et approximation
  - Démonstration assistée

Perspectives

- Obstacles à dépasser pour des protocoles industriels :

Vérification des protocoles crypto

Actuellement

- Nombreux modèles (Processus, Réécriture, Clauses,...)
- Nombreuses techniques de vérification
  - Énumération pour des cas d'études finis
  - Procédures de décision pour des sous-classes
  - Interprétation abstraite et approximation
  - Démonstration assistée

Perspectives

- Obstacles à dépasser pour des protocoles industriels :
  - Preuve de propriétés exotiques

Vérification des protocoles crypto

Actuellement

- Nombreux modèles (Processus, Réécriture, Clauses,...)
- Nombreuses techniques de vérification
  - Énumération pour des cas d'études finis
  - Procédures de décision pour des sous-classes
  - Interprétation abstraite et approximation
  - Démonstration assistée

Perspectives

- Obstacles à dépasser pour des protocoles industriels :
  - Preuve de propriétés exotiques
  - Vérification efficace : combinaison de protocoles

Vérification des protocoles crypto

Actuellement

- Nombreux modèles (Processus, Réécriture, Clauses,...)
- Nombreuses techniques de vérification
  - Énumération pour des cas d'études finis
  - Procédures de décision pour des sous-classes
  - Interprétation abstraite et approximation
  - Démonstration assistée

Perspectives

- Obstacles à dépasser pour des protocoles industriels :
  - Preuve de propriétés exotiques
  - Vérification efficace : combinaison de protocoles
  - Affaiblissement de Dolev-Yao (Crypto bas-coût)