

Cryptographie et protocoles cryptographiques

Jean Goubault-Larrecq

LSV/CNRS UMR 8643, ENS Cachan

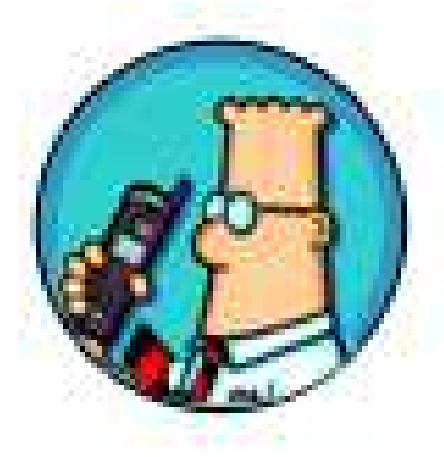
Plan

1. Comment envoyer des messages **secrets** ?
2. Un aperçu de quelques algorithmes de **chiffrement**.
3. Quelques autres propriétés de sécurité.
4. Comment tirer un nombre **au hasard** ?
5. Les **protocoles** cryptographiques ; SSL.

Alice veut envoyer un message à Bob...



solution : 2 ; 3 ; 1 ; 1



Alice veut envoyer un message à Bob...



solution : 2 ; 3 ; 1 ; 1



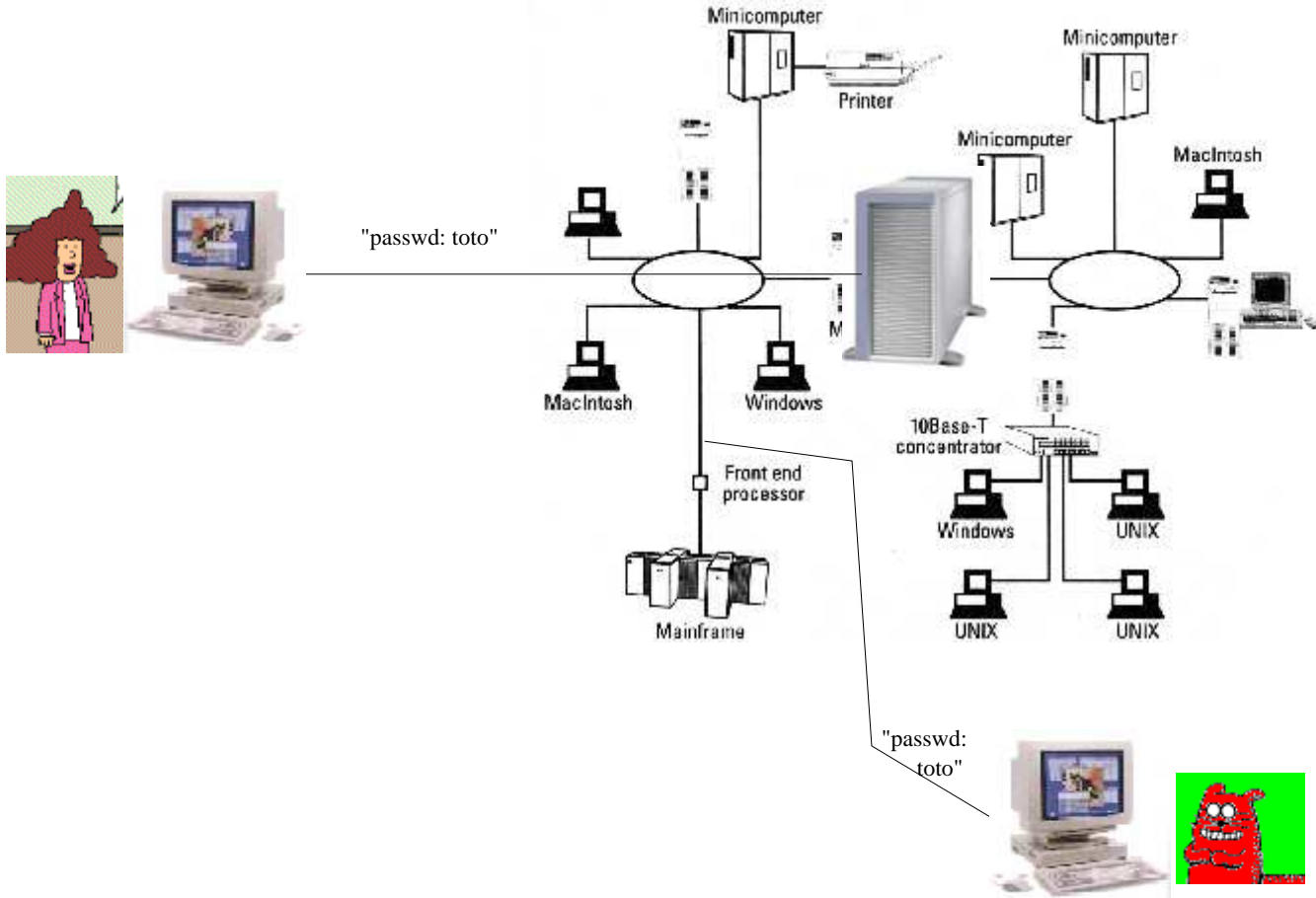
Mais Charlie peut espionner le message : pas de **secret**.

Concrètement



Tout ce qu'Alice tape circule en clair sur le câble qui relie son ordinateur (à gauche) au serveur (à droite).

N'importe qui peut espionner !



...même quand vous croyez que c'est sûr

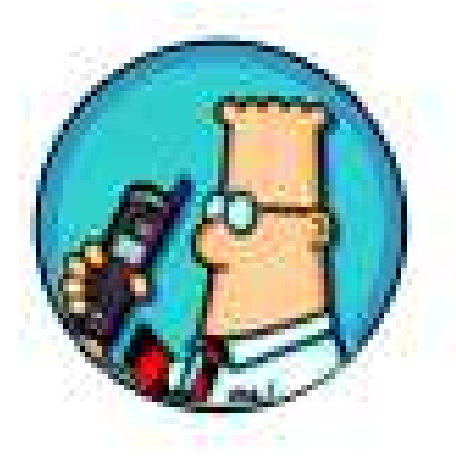
The screenshot shows a Mozilla browser window with the following elements:

- Address Bar:** `http://www.lsv.ens-cachan.fr/~goubault/SECI02/`
- Page Title:** LSV, CNRS & ENS de Cachan : Jean GOUBAULT-LARRECQ - Mozilla
- Navigation:** Back, Forward, Reload, Stop buttons.
- Bookmarks:** Home, Bookmarks, Members, WebMail, Connections, BizJournal, SmartUpdate, Mktplace.
- Sidebar:** What's Related, Search, Bookmarks (with sub-items like Biographie - Jean Goubault-Larrecq, Quelques liens, Official LSV Website, Google, Yahoo! France, People, Amiga, Seminaires, Transports, INRIA, etc.).
- Main Content:** "Laboratoire Specification et..." with a UK flag icon. Logos for ENS CACHAN and CNRS are visible. A navigation menu includes: Publications, Services, Événements, Accès, Département Informatique, Pages Privées. Contact information: 07, avenue du Président Wilson, 94235 CACHAN Cedex - France. Tél.: +33 (0)1 47 40 75 68, Fax: +33 (0)1 47 40 24 64, Secr.: +33 (0)1 47 40 75 20, Mél: goubault@lsv.ens-cachan.fr, Bureau: RH-B-111. A link for "Quelques publications" is present, with a sub-item "notes de cours".
- Portrait:** A photo of a man with glasses and a light blue turtleneck.
- Modal Dialog:** A "Prompt" dialog box with a question mark icon. Text: "Enter username and password for 'the SECI'02 PC pages' at www.lsv.ens-cachan.fr". Fields for "User Name:" and "Password:". A checkbox: "Use Password Manager to remember these values." Buttons: "OK", "Cancel".
- Status Bar:** "Sending request to www.lsv.ens-cachan.fr..."

La solution standard : le chiffrement



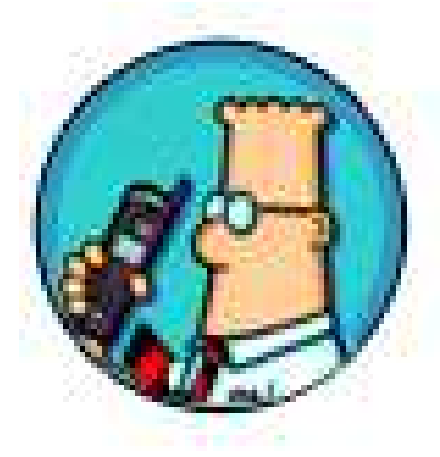
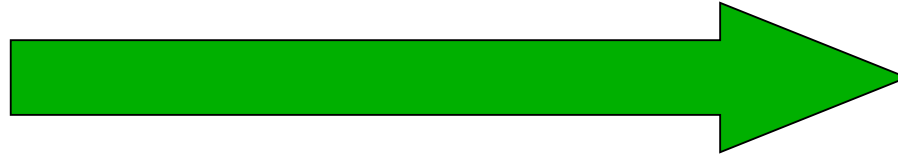
vroxwlrq = 5 > 6 > 4 > 4



La solution standard : le chiffrement



vroxwlrq = 5 > 6 > 4 > 4



Mais l'algorithme de chiffrement est trop simple...

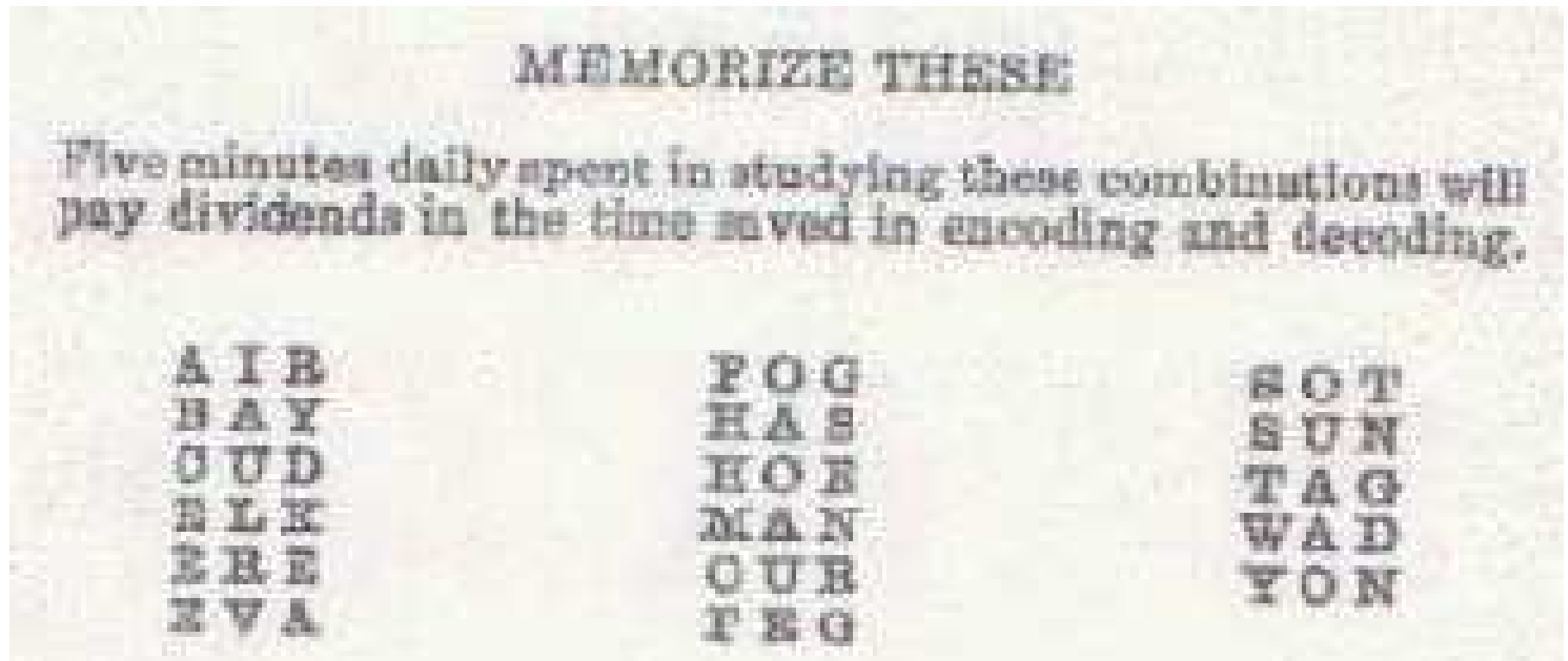
Plan

1. Comment envoyer des messages **secrets** ?
2. Un aperçu de quelques algorithmes de **chiffrement**.
3. Quelques autres propriétés de sécurité.
4. Comment tirer un nombre **au hasard** ?
5. Les **protocoles** cryptographiques ; SSL.

Un meilleur système de chiffrement

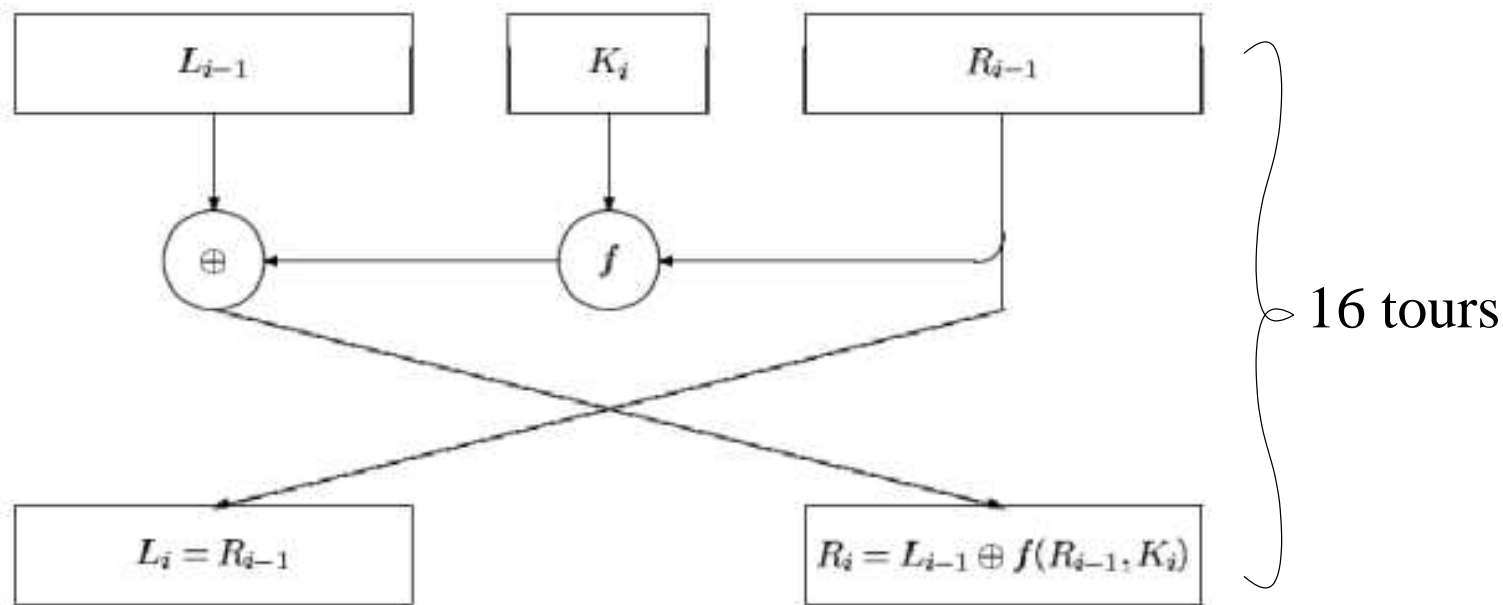
Envoyer k ème lettre \oplus k ème lettre d'un **masque jetable** (one-time pad), $1 \leq k \leq$ longueur :

(\oplus =ou exclusif, ou bien addition mod 26, etc.)



(Celui-ci est bien entendu ridicule...)

Et il y en a d'autres, le DES par exemple...



C'est un cas particulier de chiffrement **de Feistel**.

Un petit point de vocabulaire

En Français, on dit :

- **chiffrer**, pas “crypter”
- **déchiffrer**, pas “décrypter”

(to encrypt)

(to decrypt)

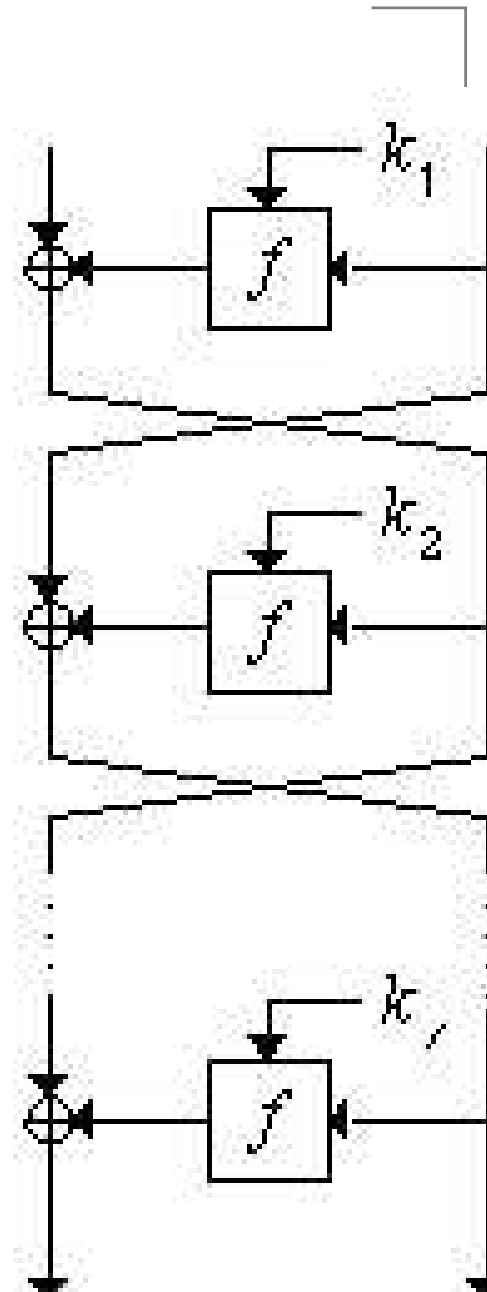
Un petit point de vocabulaire

En Français, on dit :

- **chiffrer**, pas “crypter” (to encrypt)
- **déchiffrer**, pas “décrypter” (to decrypt)
- **décrypter**... (to decrypt, to crack)

Les chiffrements de Feistel

```
proc Feistel (L, R : int)
  for i = 1 to N do
    R := L xor f (R, k[i]) ;
    L := R ;
  return (R, L) ;
```



Les chiffrements de Feistel

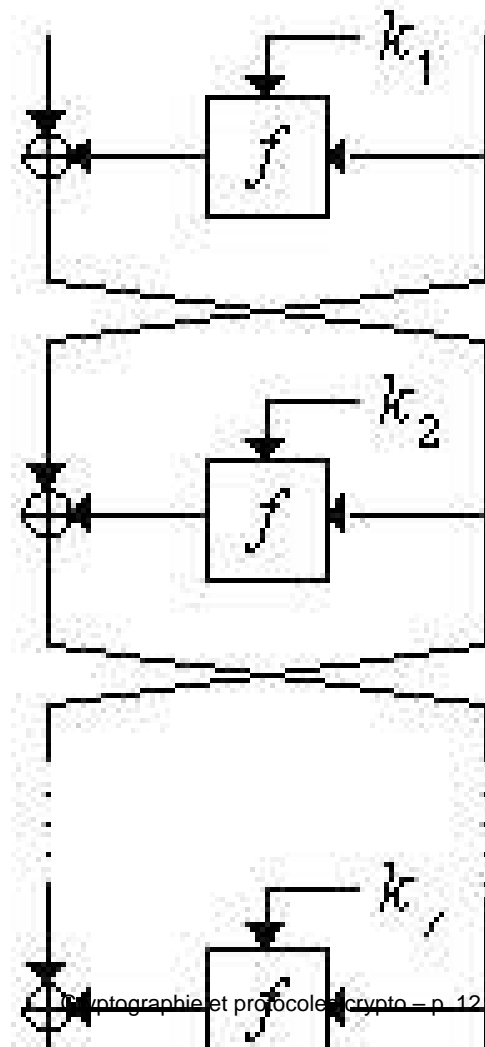
```
proc Feistel (L, R : int)
  for i = 1 to N do
    R := L xor f (R, k[i]) ;
    L := R ;
  return (R, L) ;
```

Pour le DES : $N = 16$, L et R sur 32 bits, f est calculée à l'aide de tables (les **S-box**).

Le tableau $k[i]$ est obtenu par un processus d'**expansion** à partir d'une clé de 64 bits (56 bits effectifs, 8 bits de checksum).

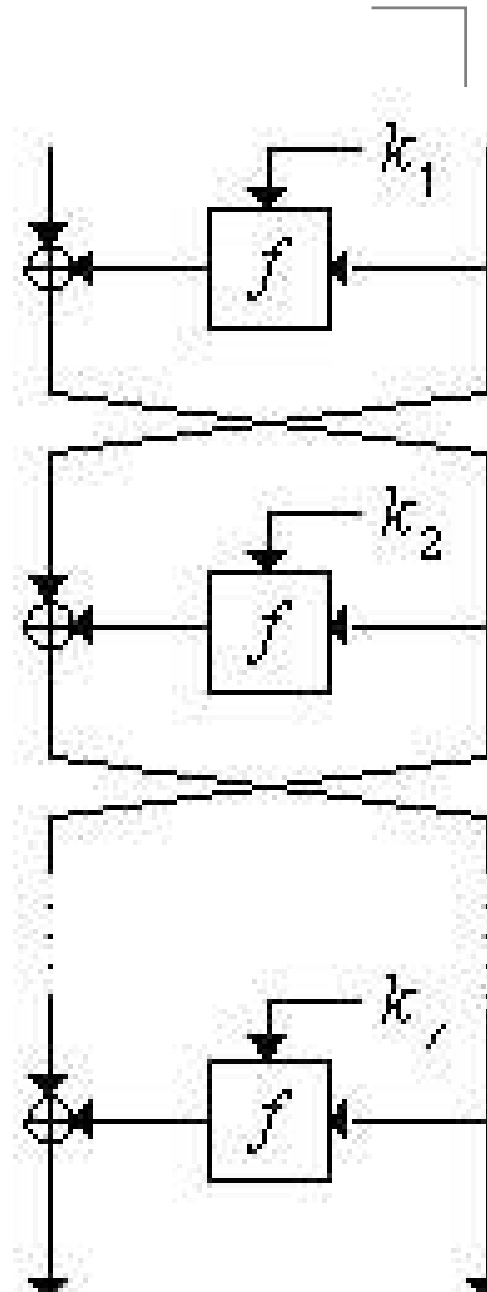
Les chiffrements de Feistel sont très **rapides** à calculer... surtout en hardware (circuits DES).

Et déchiffrer est très facile !



Les chiffrements de Feistel

```
proc Feistel (L, R : int)
  for i = 1 to N do
    R := L xor f (R, k[i]) ;
    L := R ;
  return (R, L) ;
proc Feistel-inv (R, L : int)
  for i = 1 to N do
    R := L xor f (R, k[N+1-i]) ;
    L := R ;
  return (L, R) ;
```



Un chiffrement voisin : RC5 (RFC2040)

```
proc RC5 (L, R : int)
  L := L + k[0] ; R := R + k[1] ;
  for i = 1 to N do
    L := rotl (L xor R, R) + k[2*i] ;
    R := rotl (R xor L, L) + k[2*i+1] ;
  return (L, R) ;
```

Typiquement $N = 12$ (pour une clé de 64 bits), $N = 16$ (clé de 128 bits).

L et R sont sur 32 bits.

Tableau $k[i]$ calculé par expansion de clé 64 ou 128 bits.

Toujours très rapide ! ... et moins sensible aux attaques que DES (la cryptanalyse différentielle de Shamir).

IDEA (Intl. Data Encryption Algo.)

```
proc IDEA (p : int16[4])
  for i = 1 to 8 do
    d0 :=p[0]*k[6*i-6] ; d1 :=p[1]+k[6*i-5] ;
    d2 :=p[2]+k[6*i-4] ; d3 :=p[3]*k[6*i-3] ;
    d4 :=d0 xor d2 ; d5 :=d1 xor d3 ;
    d6 :=d4*k[6*i-2] ; d7 :=d5+d6 ;
    d8 :=d7*k[6*i-1] ; d9 :=d6+d8 ;
    p[0] :=d0 xor d8 ; p[2] :=d2 xor d8 ;
    p[1] :=d1 xor d9 ; p[3] :=d3 xor d9 ;
  p[0] :=p[0]*k[48] ; p[1] :=p[1]+k[49] ;
  p[2] :=p[2]+k[50] ; p[3] :=p[3]*k[51] ;
  return p ;
```

(où l'addition est mod 2^{16} , et la multiplication mod $2^{16} + 1...$)

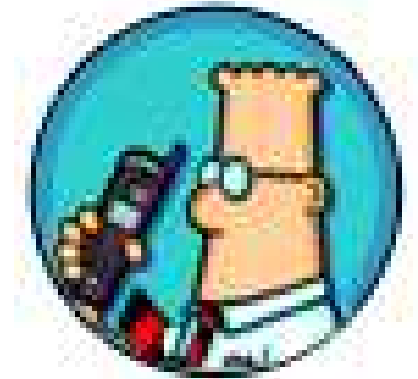
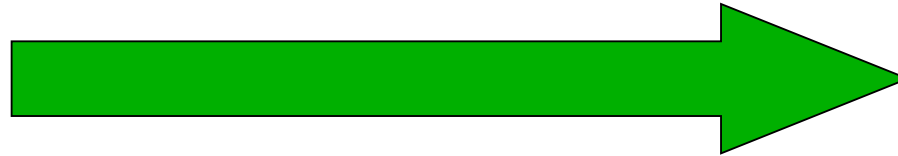
Plan

1. Comment envoyer des messages **secrets** ?
2. Un aperçu de quelques algorithmes de **chiffrement**.
3. Quelques autres propriétés de sécurité.
4. Comment tirer un nombre **au hasard** ?
5. Les **protocoles** cryptographiques ; SSL.

Avec un chiffrement amélioré...



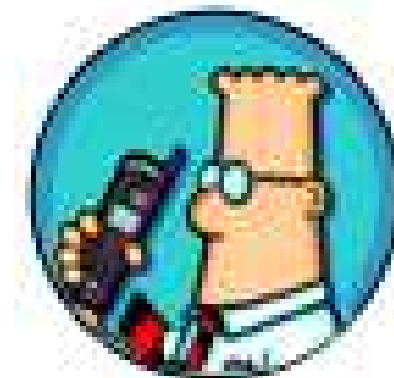
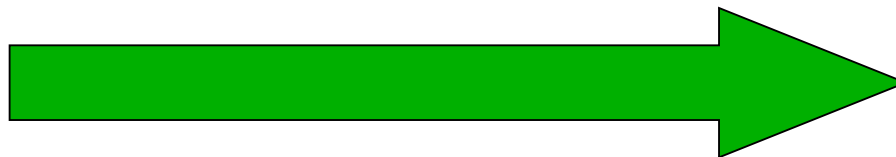
sw2z7o61 ; JB EM \geq 4
= {solution : 2 ; 3 ; 1 ; 1} $\}_K$



... Charlie peut encore nuire !



sw2z7o61 ; IB DM @ = 4
= {solution : 1 ; 2 ; 3 ; 1} K



L'intégrité du message est compromise.

⇒ éviter le chiffrement par blocs

Par ex., CBC : Découper le message en blocs (de 64 bits pour DES par ex.) :

B_1	B_2	...	B_n
-------	-------	-----	-------

Envoyer :

	$\{IV_0 \oplus B_1\}_K$	$\{IV_1 \oplus B_2\}_K$...	$\{IV_{n-1} \oplus B_n\}_K$
IV_0	$= IV_1$	$= IV_2$		$= IV_n$

Empêche de tronquer ou remplacer des bouts.

⇒ éviter le chiffrement par blocs

Par ex., CBC : Découper le message en blocs (de 64 bits pour DES par ex.) :

B_1	B_2	...	B_n
-------	-------	-----	-------

Envoyer :

	$\{IV_0 \oplus B_1\}_K$	$\{IV_1 \oplus B_2\}_K$...	$\{IV_{n-1} \oplus B_n\}_K$
IV_0	$= IV_1$	$= IV_2$		$= IV_n$

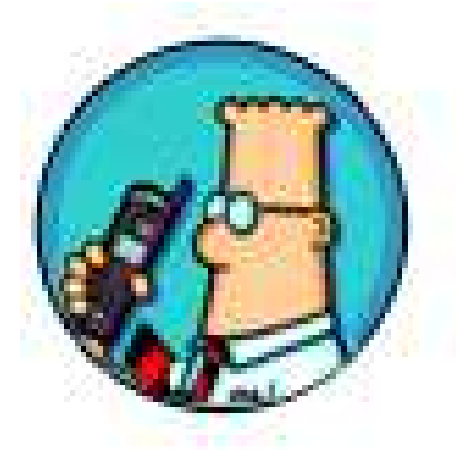
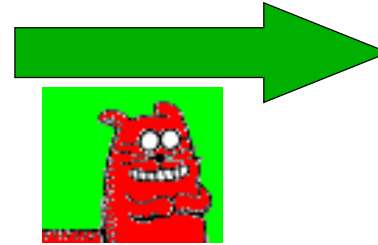
Empêche de tronquer ou remplacer des bouts.

Intégrité : inclure des sommes de contrôle (fonctions de hachage ; la plus connue : MD5).

On n'arrête pas Charlie !



jm ss :o1t =hs
= {je ne sais pas}_K

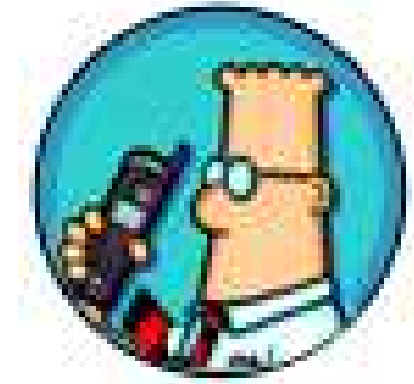
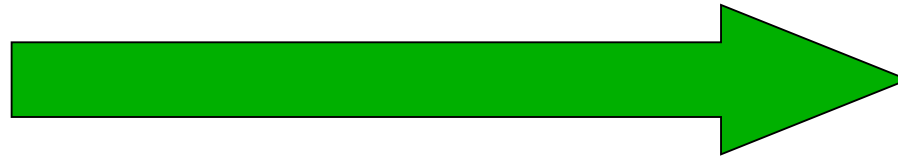


Charlie peut rejouer un vieux message, même sans connaître la clé K .

Alice inclut un temps dans son message...



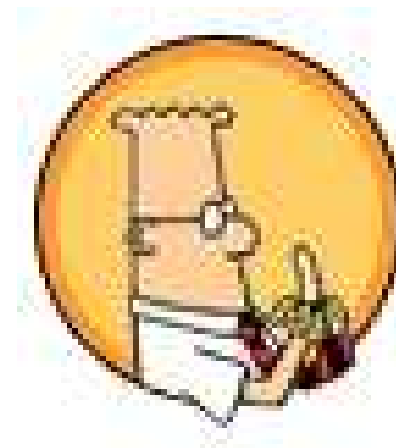
sw2z7o61 ; JB EM >= 4 - C9@5=
= {solution : 2 ; 3 ; 1 ; 1 - 09 :12} _K



Charlie ne peut plus rejouer de messages

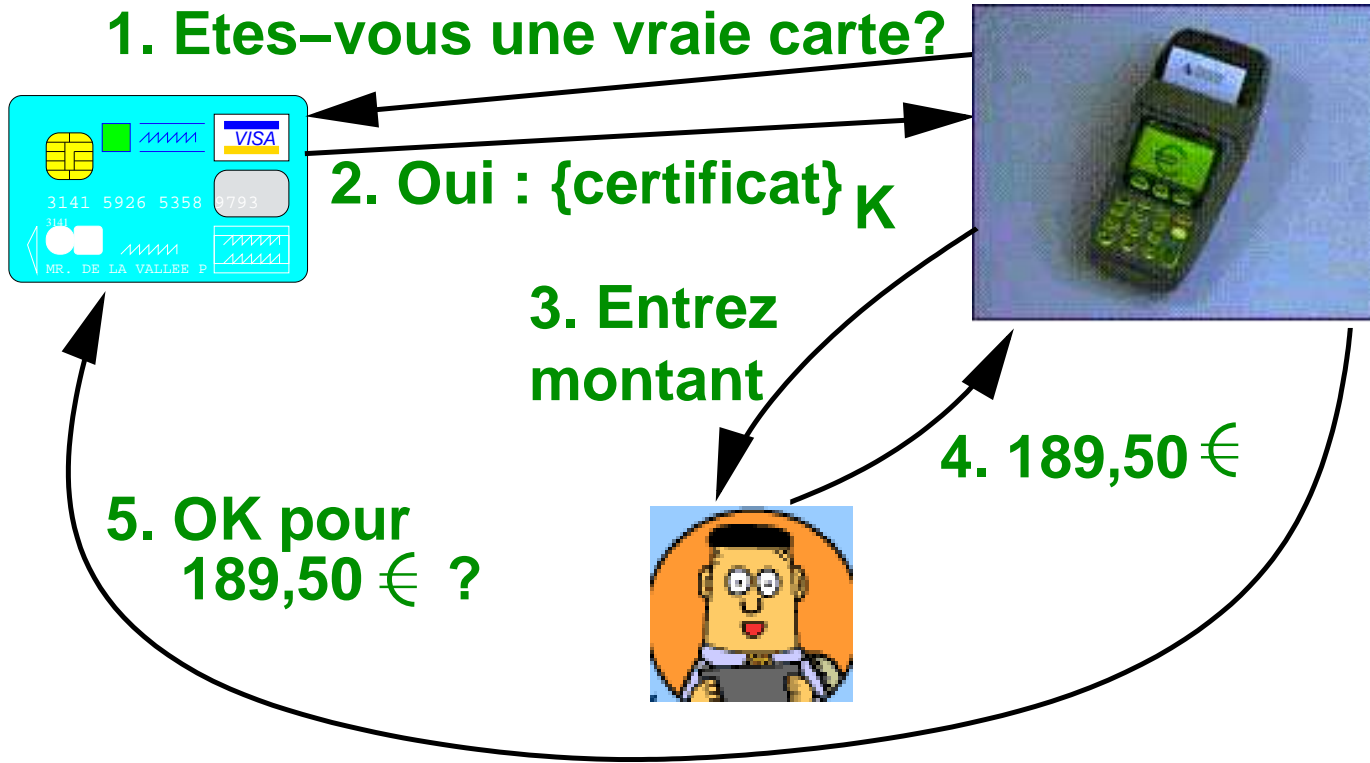


sw2z7o61 ; IB DM @ = 4 - C8@4@
= {solution : 1 ; 2 ; 3 ; 1 - 08 :05} K

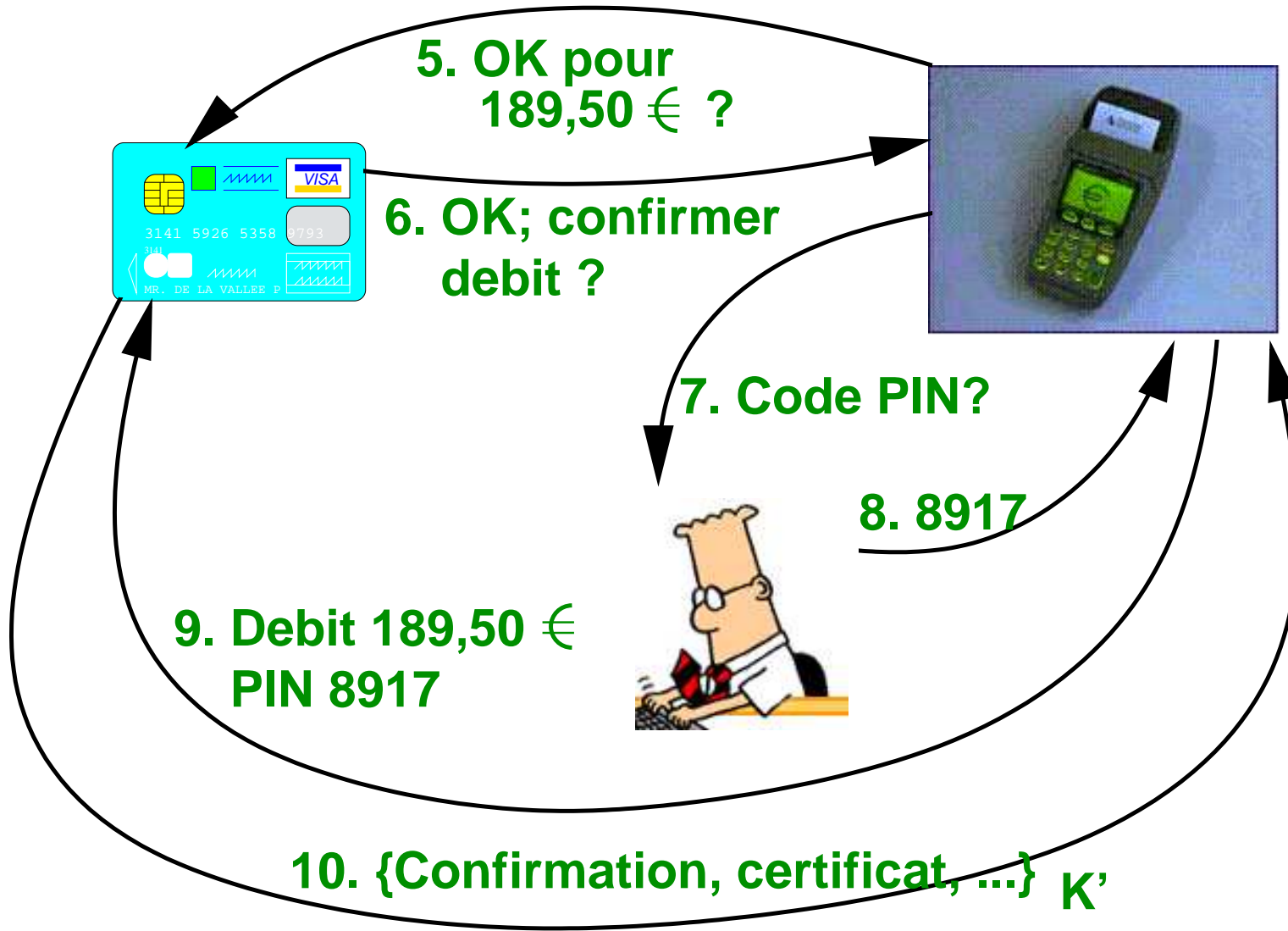


Ceci demande cependant des horloges synchronisés, donc un protocole de **synchronisation** (à la ntp) **sécurisé**.
On verra des solutions plus simples à base de **nonces**.

Un exemple concret : les cartes à puce



Les cartes à puce, suite



En résumé

Alice a dû utiliser :

- un algorithme de **chiffrement fort** **secret** ;
- des **sommes de contrôle** **intégrité** ;
- des **estampilles** temporelles **fraîcheur** ;
- plus quelques autres précautions de codage (CBC, types explicites, ...) ;
- plus divers **authenticité, unicité, ...**

Avec tout ça, on devrait être parés ?

Plan

1. Comment envoyer des messages **secrets** ?
2. Un aperçu de quelques algorithmes de **chiffrement**.
3. Quelques autres propriétés de sécurité.
4. Comment tirer un nombre **au hasard** ?
5. Les **protocoles** cryptographiques ; SSL.

Nonces

En fait, on a souvent besoin de fabriquer des données **nouvelles** (jamais vues auparavant, et imprédictibles).

Nonces

En fait, on a souvent besoin de fabriquer des données **nouvelles** (jamais vues auparavant, et imprédictibles).

- Utile pour fabriquer des **clés nouvelles**.

Nonces

En fait, on a souvent besoin de fabriquer des données **nouvelles** (jamais vues auparavant, et imprédictibles).

- Utile pour fabriquer des **clés nouvelles**.
- Peut servir à garantir la **fraîcheur** :

new N_a

1. $A \rightarrow B : N_a, \dots$

2. $B \rightarrow A : \{N_a, \dots\}_K$

garantit que le message 2 est au moins aussi récent que le message 1.

Nombres (pseudo-)aléatoires

Pour fabriquer un nombre nouveau (disons sur 128 bits), on le tire **au hasard**.

Nombres (pseudo-)aléatoires

Pour fabriquer un nombre nouveau (disons sur 128 bits), on le tire **au hasard**.

198 089 357 218 068 216 517 260 685 773 624 326 765

Nombres (pseudo-)aléatoires

Pour fabriquer un nombre nouveau (disons sur 128 bits), on le tire **au hasard**.

198 089 357 218 068 216 517 260 685 773 624 326 765
324 073 163 420 648 383 635 925 543 613 609 217 851

Nombres (pseudo-)aléatoires

Pour fabriquer un nombre nouveau (disons sur 128 bits), on le tire **au hasard**.

198 089 357 218 068 216 517 260 685 773 624 326 765
324 073 163 420 648 383 635 925 543 613 609 217 851
232 992 305 167 558 465 512 230 572 597 182 844 496

Nombres (pseudo-)aléatoires

Pour fabriquer un nombre nouveau (disons sur 128 bits), on le tire **au hasard**.

198 089 357 218 068 216 517 260 685 773 624 326 765
324 073 163 420 648 383 635 925 543 613 609 217 851
232 992 305 167 558 465 512 230 572 597 182 844 496
285 921 562 481 663 339 762 468 133 626 338 094 559

Nombres (pseudo-)aléatoires

Pour fabriquer un nombre nouveau (disons sur 128 bits), on le tire **au hasard**.

198 089 357 218 068 216 517 260 685 773 624 326 765
324 073 163 420 648 383 635 925 543 613 609 217 851
232 992 305 167 558 465 512 230 572 597 182 844 496
285 921 562 481 663 339 762 468 133 626 338 094 559
311 193 852 387 359 001 535 265 578 684 927 465 138

Nombres (pseudo-)aléatoires

Pour fabriquer un nombre nouveau (disons sur 128 bits), on le tire **au hasard**.

198 089 357 218 068 216 517 260 685 773 624 326 765

324 073 163 420 648 383 635 925 543 613 609 217 851

232 992 305 167 558 465 512 230 572 597 182 844 496

285 921 562 481 663 339 762 468 133 626 338 094 559

311 193 852 387 359 001 535 265 578 684 927 465 138

...

Le risque de tomber sur un nombre déjà fabriqué est infime...

Nombre pseudo-aléatoires et fraîcheur

Âge de l'univers : 15 milliards d'années $\sim 4,73 \cdot 10^{17}$ s.

Nombre pseudo-aléatoires et fraîcheur

Âge de l'univers : 15 milliards d'années $\sim 4,73 \cdot 10^{17}$ s.

Population totale depuis le début de l'univers $\sim 10^{10}$.

Nombre pseudo-aléatoires et fraîcheur

Âge de l'univers : 15 milliards d'années $\sim 4,73 \cdot 10^{17}$ s.

Population totale depuis le début de l'univers $\sim 10^{10}$.

Supposons que chaque humain invente un nombre chaque seconde.

\Rightarrow au plus $5 \cdot 10^{27}$ nombres sont non frais.

Nombre pseudo-aléatoires et fraîcheur

Âge de l'univers : 15 milliards d'années $\sim 4,73 \cdot 10^{17}$ s.

Population totale depuis le début de l'univers $\sim 10^{10}$.

Supposons que chaque humain invente un nombre chaque seconde.

\Rightarrow au plus $5 \cdot 10^{27}$ nombres sont non frais.

Il y a $3,4 \cdot 10^{38}$ nombres de 128 bits.

Nombre pseudo-aléatoires et fraîcheur

Âge de l'univers : 15 milliards d'années $\sim 4,73 \cdot 10^{17}$ s.

Population totale depuis le début de l'univers $\sim 10^{10}$.

Supposons que chaque humain invente un nombre chaque seconde.

\Rightarrow au plus $5 \cdot 10^{27}$ nombres sont non frais.

Il y a $3,4 \cdot 10^{38}$ nombres de 128 bits.

On n'a donc qu'une chance sur 68 milliards (au plus !) de tirer au hasard un nombre non frais.

Aléa et imprédictibilité

Un générateur aléatoire est **cryptographiquement sûr** si et seulement si, étant données les n premières valeurs produites

$$x_1, x_2, \dots, x_n$$

on ne peut pas produire la suivante x_{n+1} par calcul [en temps raisonnable].

... sinon un intrus pourrait produire le nonce N_a avant A !

Aléa et imprédictibilité

Un générateur pseudo-aléatoire courant : le générateur **linéaire congruentiel**.

$$x_{n+1} = ax_n + b \text{ mod } M$$

avec a et M premiers entre eux. (M peut être supposé connu.)

Aléa et imprédictibilité

Un générateur pseudo-aléatoire courant : le générateur **linéaire congruentiel**.

$$x_{n+1} = ax_n + b \pmod{M}$$

avec a et M premiers entre eux. (M peut être supposé connu.)

Extrêmement facile à prédire :

Connaissant x_1, x_2 , et x_3 ,

$$a = \frac{x_2 - x_1}{x_1 - x_0} \quad b = x_2 - ax_1$$

Maintenant l'intrus connaît x_4, x_5 , etc.

Aléa et imprédictibilité

Un générateur pseudo-aléatoire courant : le générateur **linéaire congruentiel**.

$$x_{n+1} = ax_n + b \pmod{M}$$

avec a et M premiers entre eux. (M peut être supposé connu.)

Extrêmement facile à prédire :

Connaissant x_1, x_2 , et x_3 ,

$$a = \frac{x_2 - x_1}{x_1 - x_0} \quad b = x_2 - ax_1$$

Maintenant l'intrus connaît x_4, x_5 , etc.

Malheureusement, de nombreux générateurs pseudo-aléatoires sont **prédictibles**. N'utilisez jamais

`random()` pour de la cryptographie !

Le générateur de Blum-Micali

Soit $N = pq$, produit de deux grands nombres premiers (disons 128 bits). Soit n entier assez grand.

À partir d'un nombre aléatoire k , poser

$$x_n = k \quad x_{i-1} = x_i^2 \bmod N \quad (i = N..2)$$

On tire ensuite x_1, x_2, x_3, \dots , dans l'ordre inverse de leur génération.

Le générateur de Blum-Micali

Soit $N = pq$, produit de deux grands nombres premiers (disons 128 bits). Soit n entier assez grand.

À partir d'un nombre aléatoire k , poser

$$x_n = k \quad x_{i-1} = x_i^2 \bmod N \quad (i = N..2)$$

On tire ensuite x_1, x_2, x_3, \dots , dans l'ordre inverse de leur génération.

On peut montrer que si l'on savait prédire ce générateur, on saurait factoriser N ; ce qui est supposé infaisable.

Le générateur de Blum-Micali

Soit $N = pq$, produit de deux grands nombres premiers (disons 128 bits). Soit n entier assez grand.

À partir d'un nombre aléatoire k , poser

$$x_n = k \quad x_{i-1} = x_i^2 \bmod N \quad (i = N..2)$$

On tire ensuite x_1, x_2, x_3, \dots , dans l'ordre inverse de leur génération.

On peut montrer que si l'on savait prédire ce générateur, on saurait factoriser N ; ce qui est supposé infaisable.

Comment tirer k ? PGP utilise un comptage des nombres de millisecondes entre deux frappes de touche au clavier. Sous

Linux, utiliser `/dev/random/`, ou `RAND_egd()`.

Plan

1. Comment envoyer des messages **secrets** ?
2. Un aperçu de quelques algorithmes de **chiffrement**.
3. Quelques autres propriétés de sécurité.
4. Comment tirer un nombre **au hasard** ?
5. Les **protocoles** cryptographiques ; SSL.

Un exemple [NeedhamSchroeder78]



A



B



S

new Na
write A, B, Na

read A, B, Na
new sym key Kab
write {Na, B, Kab, {Kab, A | Kbs} | Kas}

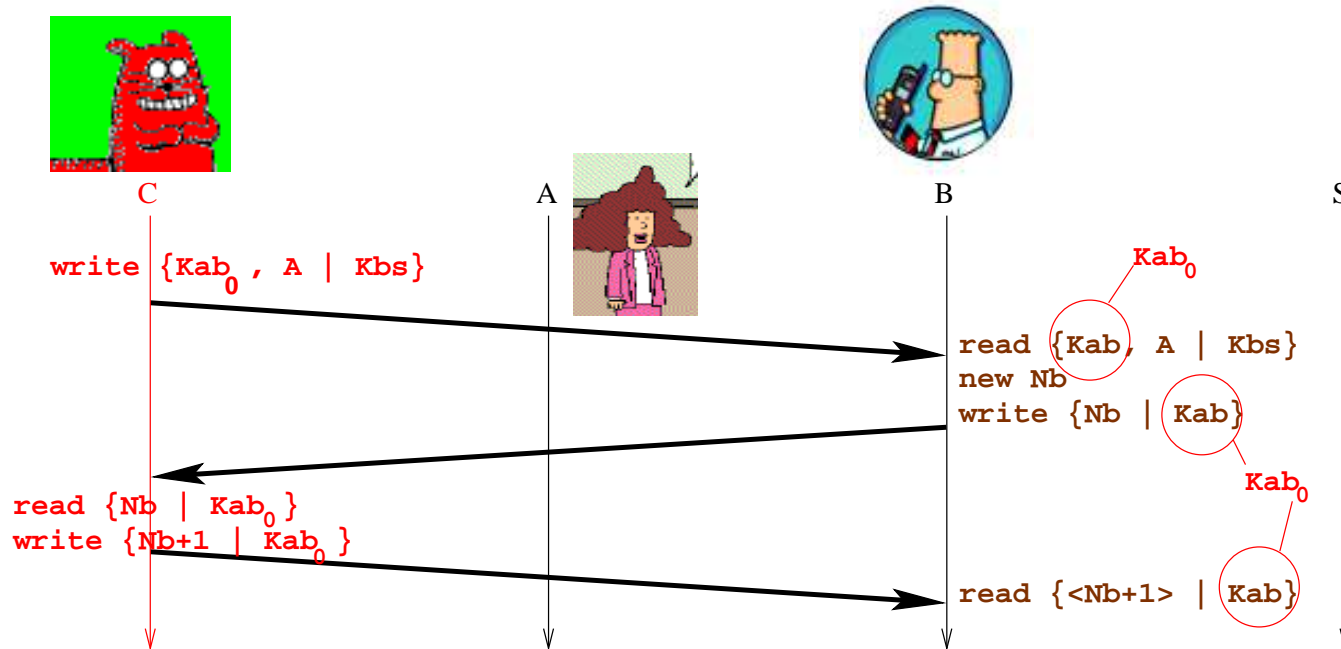
read {<Na>, , Kab, message | Kas}
write message

read {Kab, A | Kbs}
new Nb
write {Nb | Kab}

read {Nb | Kab}
write {Nb+1 | Kab}

read {<Nb+1> | Kab}

Une attaque (secret, authentication)



Miracle : la cryptographie à clé publique

A fabrique K_a, K_a^{-1} , publie K_a , garde K_a^{-1} .

Pour **chiffrer** M , B utilise la clé **publique** K_a :

$$M' = \{M\}_{K_a}.$$

A **déchiffre** avec sa clé **privée** :

$$M = \{M'\}_{K_a^{-1}}.$$

Miracle : la cryptographie à clé publique

A fabrique K_a, K_a^{-1} , publie K_a , garde K_a^{-1} .

Pour **chiffrer** M , B utilise la clé **publique** K_a :

$$M' = \{M\}_{K_a}.$$

A **déchiffre** avec sa clé **privée** :

$$M = \{M'\}_{K_a^{-1}}.$$

Arg : M' n'est pas forcément **authentique** (K_a est publique...) \implies B **signe** avec sa clé **privée** K_b^{-1} .

Miracle : la cryptographie à clé publique

A fabrique K_a, K_a^{-1} , publie K_a , garde K_a^{-1} .

Pour **chiffrer** M , B utilise la clé **publique** K_a :

$$M' = \{M\}_{K_a}.$$

A **déchiffre** avec sa clé **privée** :

$$M = \{M'\}_{K_a^{-1}}.$$

Arg : M' n'est pas forcément **authentique** (K_a est publique...) \implies B **signe** avec sa clé **privée** K_b^{-1} .

Secret+authenticité : B envoie $\{M, \underbrace{\{h(M)\}_{K_b^{-1}}}_{M'}\}_{K_a}$

A vérifie ensuite

$$h(M) = \{M'\}_{K_b}.$$

algorithme RSA [RivestShamirAdleman7

Génération de clés : tirer deux grands nombres premiers $p \neq q$. $N = pq$, $\phi(N) = (p - 1)(q - 1)$. Tirer $d \in [2, \phi(N)[$ premier avec $\phi(N)$. Calculer e tel que $de = 1 \pmod{\phi(N)}$ (Bezout).

Clé **publique** : (N, e)

privée : (N, d) .

Chiffrement : $\{M\}_{(N,e)} = M^e \pmod{N}$.

Déchiffrement : $\{M'\}_{(N,d)} = M'^d \pmod{N}$.

(Exercice : ces fonctions sont inverses.)

RSA sur un exemple

$p = 11\ 508\ 288\ 272\ 609\ 080\ 559$, $q = 16\ 540\ 558\ 193\ 182\ 354\ 801$
(64 bits).

$N = 190\ 353\ 511\ 877\ 008\ 536\ 544\ 166\ 732\ 523\ 129\ 413\ 759$

$\phi(N) = 190\ 353\ 511\ 877\ 008\ 536\ 516\ 117\ 886\ 057\ 337\ 978\ 400$

$d = 68\ 551\ 770\ 307\ 036\ 473\ 070\ 080\ 515\ 976\ 000\ 236\ 897$,

$e = 71\ 313\ 093\ 839\ 083\ 143\ 987\ 818\ 093\ 247\ 265\ 904\ 033$.

RSA sur un exemple

$p = 11\ 508\ 288\ 272\ 609\ 080\ 559$, $q = 16\ 540\ 558\ 193\ 182\ 354\ 801$
(64 bits).

$N = 190\ 353\ 511\ 877\ 008\ 536\ 544\ 166\ 732\ 523\ 129\ 413\ 759$

$\phi(N) = 190\ 353\ 511\ 877\ 008\ 536\ 516\ 117\ 886\ 057\ 337\ 978\ 400$

$d = 68\ 551\ 770\ 307\ 036\ 473\ 070\ 080\ 515\ 976\ 000\ 236\ 897$,

$e = 71\ 313\ 093\ 839\ 083\ 143\ 987\ 818\ 093\ 247\ 265\ 904\ 033$.

Message en clair : $M = \text{“sol :1 ;2 ;1 ; 08 :05”} =$
153 439 759 577 269 640 773 274 743 596 375 486 517.

RSA sur un exemple

$p = 11\ 508\ 288\ 272\ 609\ 080\ 559$, $q = 16\ 540\ 558\ 193\ 182\ 354\ 801$
(64 bits).

$N = 190\ 353\ 511\ 877\ 008\ 536\ 544\ 166\ 732\ 523\ 129\ 413\ 759$

$\phi(N) = 190\ 353\ 511\ 877\ 008\ 536\ 516\ 117\ 886\ 057\ 337\ 978\ 400$

$d = 68\ 551\ 770\ 307\ 036\ 473\ 070\ 080\ 515\ 976\ 000\ 236\ 897$,

$e = 71\ 313\ 093\ 839\ 083\ 143\ 987\ 818\ 093\ 247\ 265\ 904\ 033$.

Message en clair : $M = \text{"sol :1 ;2 ;1 ; 08 :05"} =$
 $153\ 439\ 759\ 577\ 269\ 640\ 773\ 274\ 743\ 596\ 375\ 486\ 517$.

On chiffre : $M' = M^e \bmod N$

$= 56\ 605\ 557\ 572\ 603\ 317\ 510\ 437\ 246\ 813\ 714\ 386\ 816$

$= \text{"*\149\tilde{v}2 ;\frac{3}{4}\130]f5\ ^R\tilde{o}\hat{O}\128"}$.

RSA sur un exemple

$p = 11\ 508\ 288\ 272\ 609\ 080\ 559$, $q = 16\ 540\ 558\ 193\ 182\ 354\ 801$
(64 bits).

$N = 190\ 353\ 511\ 877\ 008\ 536\ 544\ 166\ 732\ 523\ 129\ 413\ 759$

$\phi(N) = 190\ 353\ 511\ 877\ 008\ 536\ 516\ 117\ 886\ 057\ 337\ 978\ 400$

$d = 68\ 551\ 770\ 307\ 036\ 473\ 070\ 080\ 515\ 976\ 000\ 236\ 897$,

$e = 71\ 313\ 093\ 839\ 083\ 143\ 987\ 818\ 093\ 247\ 265\ 904\ 033$.

Message en clair : $M = \text{"sol :1 ;2 ;1 ; 08 :05"} =$
 $153\ 439\ 759\ 577\ 269\ 640\ 773\ 274\ 743\ 596\ 375\ 486\ 517$.

On chiffre : $M' = M^e \bmod N$

$= 56\ 605\ 557\ 572\ 603\ 317\ 510\ 437\ 246\ 813\ 714\ 386\ 816$

$= \text{"*\149\tilde{v}2 ;\frac{3}{4}\130]f5\^R\hat{O}\128"}$.

On déchiffre : $M'^d \bmod N =$

$153\ 439\ 759\ 577\ 269\ 640\ 773\ 274\ 743\ 596\ 375\ 486\ 517 = M$.

RSA sur un exemple

$p = 11\ 508\ 288\ 272\ 609\ 080\ 559$, $q = 16\ 540\ 558\ 193\ 182\ 354\ 801$
(64 bits).

$N = 190\ 353\ 511\ 877\ 008\ 536\ 544\ 166\ 732\ 523\ 129\ 413\ 759$

$\phi(N) = 190\ 353\ 511\ 877\ 008\ 536\ 516\ 117\ 886\ 057\ 337\ 978\ 400$

$d = 68\ 551\ 770\ 307\ 036\ 473\ 070\ 080\ 515\ 976\ 000\ 236\ 897$,

$e = 71\ 313\ 093\ 839\ 083\ 143\ 987\ 818\ 093\ 247\ 265\ 904\ 033$.

Message en clair : $M = \text{"sol :1 ; 2 ; 1 ; 08 :05"} =$
 $153\ 439\ 759\ 577\ 269\ 640\ 773\ 274\ 743\ 596\ 375\ 486\ 517$.

RSA sur un exemple

$p = 11\ 508\ 288\ 272\ 609\ 080\ 559$, $q = 16\ 540\ 558\ 193\ 182\ 354\ 801$
(64 bits).

$N = 190\ 353\ 511\ 877\ 008\ 536\ 544\ 166\ 732\ 523\ 129\ 413\ 759$

$\phi(N) = 190\ 353\ 511\ 877\ 008\ 536\ 516\ 117\ 886\ 057\ 337\ 978\ 400$

$d = 68\ 551\ 770\ 307\ 036\ 473\ 070\ 080\ 515\ 976\ 000\ 236\ 897$,

$e = 71\ 313\ 093\ 839\ 083\ 143\ 987\ 818\ 093\ 247\ 265\ 904\ 033$.

Message en clair : $M = \text{“sol :1 ;2 ;1 ; 08 :05”} =$
 $153\ 439\ 759\ 577\ 269\ 640\ 773\ 274\ 743\ 596\ 375\ 486\ 517$.

L'intrus modifie M' légèrement :

$M' = 56\ 605\ 557\ 572\ 603\ 317\ 510\ 437\ 246\ 813\ 714\ 386\ 817$.

RSA sur un exemple

$p = 11\ 508\ 288\ 272\ 609\ 080\ 559$, $q = 16\ 540\ 558\ 193\ 182\ 354\ 801$
(64 bits).

$N = 190\ 353\ 511\ 877\ 008\ 536\ 544\ 166\ 732\ 523\ 129\ 413\ 759$

$\phi(N) = 190\ 353\ 511\ 877\ 008\ 536\ 516\ 117\ 886\ 057\ 337\ 978\ 400$

$d = 68\ 551\ 770\ 307\ 036\ 473\ 070\ 080\ 515\ 976\ 000\ 236\ 897$,

$e = 71\ 313\ 093\ 839\ 083\ 143\ 987\ 818\ 093\ 247\ 265\ 904\ 033$.

Message en clair : $M = \text{"sol :1 ; 2 ; 1 ; 08 :05"} =$
 $153\ 439\ 759\ 577\ 269\ 640\ 773\ 274\ 743\ 596\ 375\ 486\ 517$.

L'intrus modifie M' légèrement :

$M' = 56\ 605\ 557\ 572\ 603\ 317\ 510\ 437\ 246\ 813\ 714\ 386\ 817$.

On déchiffre : $M'^d \bmod N =$

$95\ 309\ 921\ 257\ 222\ 517\ 132\ 517\ 799\ 055\ 788\ 481\ 754$

$= \text{"G' \^Ei&ú' Àâ \^_ \t \265 \^_ \376 \^ \Ú"} (\rightarrow \text{poubelle})$.

La cryptographie à clé publique

C'est bien mais :

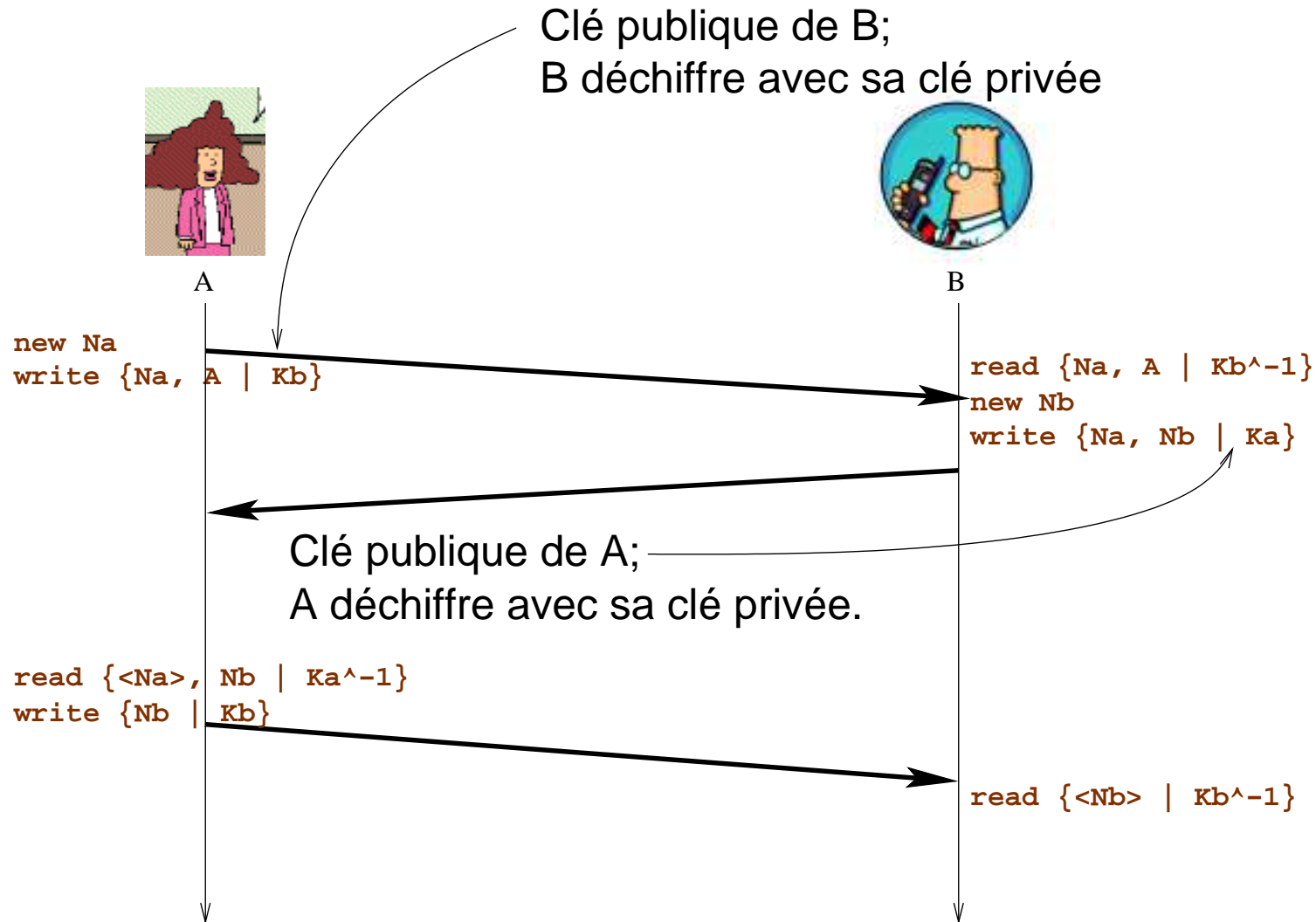
- C'est lourd et **lent** (RSA est typiquement 1000 fois plus lent que DES).

La cryptographie à clé publique

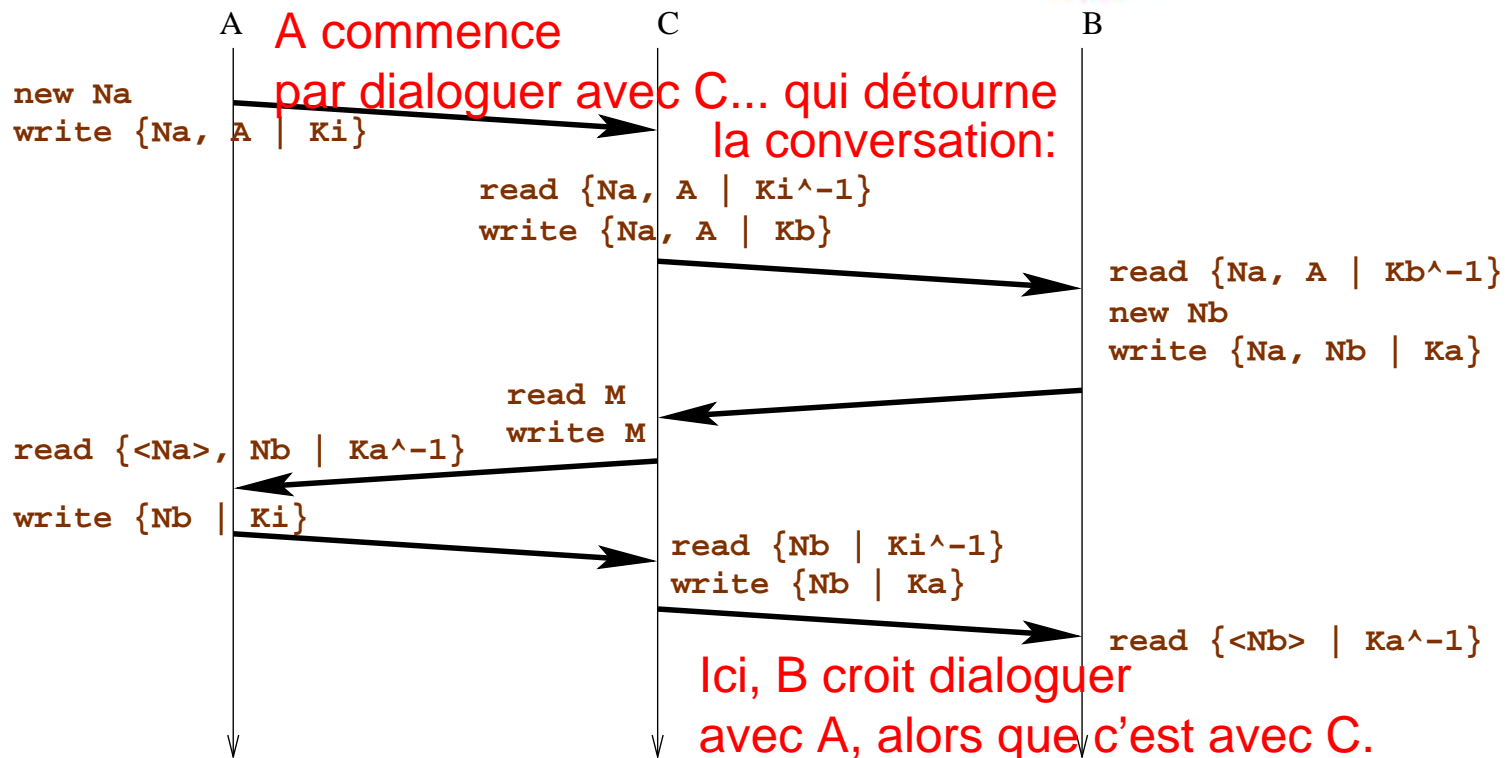
C'est bien mais :

- C'est lourd et **lent** (RSA est typiquement 1000 fois plus lent que DES).
- Ça ne change rien au fait que les protocoles cryptographiques contiennent des bugs **purement logiques** !

Un classique [NeedhamSchroeder78]



Caramba, encore raté !



Comment ça se passe, un protocole crypto

Un exemple : la commande d'un billet de train sur `www.sncf.com` sous Mozilla.

Plan

1. Comment envoyer des messages **secrets** ?
2. Un aperçu de quelques algorithmes de **chiffrement**.
3. Quelques autres propriétés de sécurité.
4. Comment tirer un nombre **au hasard** ?
5. Les **protocoles** cryptographiques ; SSL.

J'ai défini ma commande : payons !

Commande - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop

http://www.voyages-sncf.com/dynamic/_SvTermAtos?_TMS=1051529169341&_DLG=SvTermAt Search Print

Home Bookmarks Members WebMail Connections BizJournal SmartUpdate Mktplace

Sidebar

What's Related

Search

Bookmarks

Add... Manage Search...

Name

- Bande dessinée: Albums, dessins ...
- Kid Comics - Le club des fans de...
- http://www.canalj.fr/
- Rapid Phase Cartoons
- La couvertures du Ouébe Fluide ...
- Comics.com | Site Map
- France 5
- Fininfo: Informations Boursières
- Childrens Software, Paint Program...
- Plan du site "Algorithmique et Pro...
- Riven Help -- Animals
- Welcome to uComics --The Best C...
- AS DE PIQUE: Annuaire des Sites...
- Toute la magie avec Magies.com - ...
- Bananalotto.fr - Gagnez 1 million d...
- Bingopoly : 100% jeux, 500% gain...
- HoaxBuster - Première ressource f...
- Jargon File Resources
- Agora de l'Aquariophilie d'eau dou...
- The Bubulle Red - Redirect by uli...

Jobs

Math

History

Document Done (4.141 secs)







voyages-sncf.com

accueil week-ends séjours trains avions hôtels voitures promos voya

Plan du site Aide

PAIEMENT EN LIGNE

Choisissez un moyen de paiement ci-dessous

Si vous avez choisi un retrait en Billetterie Automatique ou au guichet : vous devrez utiliser la même carte de paiement et le code confidentiel associé (avant la date de fin de validité de votre carte de paiement).

Après avoir choisi votre moyen de paiement, votre navigateur va passer en mode sécurisé. Veuillez à respecter **IMPÉRATIVEMENT** les étapes suivantes:

- Sur la page suivante, cliquez sur le bouton **VALIDER** après avoir saisi votre numéro de carte et sa date de fin de validité.
- Sur la page de confirmation de la transaction, cliquez **OBLIGATOIREMENT** sur le bouton **CONTINUER** pour obtenir votre référence de dossier.

En remplissant ces conditions, vous obtiendrez votre référence SNCF de dossier voyage sur 8 lettres qui vous permettra de retirer vos titres en France (si vous n'avez pas demandé l'envoi gratuit par courrier ou le Billet Imprimé®) et de suivre votre commande.

Le site n'accepte pas la carte "e-carte bleue".

ANNULER

Une fois qu'on a cliqué...

Le client Mozilla C engage le “**SSL handshake**” avec le serveur S de la SNCF :

1. $C \rightarrow S$: Hello ;
2. $S \rightarrow C$: certificat signé CA ;
3. C vérifie que le signataire du CA est dans une liste d'autorités de certification de confiance ;
4. C fabrique une clé K de session, l'envoie à S chiffrée par la clé publique de S (qu'il a reçue dans le CA) ;
5. S obtient K par déchiffrement.

Si tout a bien fonctionné, C et S peuvent maintenant dialoguer en utilisant la clé K de session.

SSL Handshake, plus précisément

1. ClientHello : C -> S : Vc, N1, Id, KEA, SA

2. ServerHello : S -> C : Vs, N2, Id, KEA, SA

(* si KEA="RSA", SA="RSA" par exemple: *)

CA = X509s, N3, sign, Issuer, T1, T2, S, PK(S)

3. ServerCert :

S -> C : CA, {sign (CA)}_{SK(Issuer)}

(* Le client génère la nouvelle clé K: *)

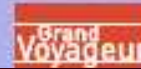
4. ClientKeyExchange :

C -> S : { Vc, K }_{PK(S)}

Mozilla passe en mode sécurisé

PAIEMENT EN LIGNE

Choisissez un moyen de paiement ci-dessous



Le site n'accepte pas la carte "e-carte bleue".

← ANNULER

La page est maintenant chiffrée

SIPS - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop <https://payment.sips-atos.com:443/cgis-payment/prod/callpayment> Search Print

Home Bookmarks Members WebMail Connections BizJournal SmartUpdate Mktpace

Sidebar Tabs x

- What's Related
- Search
- Bookmarks

Add... Manage Search...

Name

- Bande dessinée: Albums, dessins ...
- Kid Comics - Le club des fans de...
- http://www.canalj.fr/
- Rapid Phase Cartoons
- La couvertures du Ouëbe Fluide ...
- Comics.com | Site Map
- France 5
- Fininfo: Informations Boursières
- Childrens Software, Paint Program...
- Plan du site "Algorithmique et Pro...
- Riven Help -- Animals
- Welcome to uComics --The Best C...
- AS DE PIQUE: Annuaire des Sites...
- Toute la magie avec Magies.com - ...
- Bananalotto.fr - Gagnez 1 million d...
- Bingopoly : 100% jeux, 500% gain...
- HoaxBuster - Première ressource f...
- Jargon File Resources
- Agora de l'Aquariophilie d'eau dou...
- The Bubulle Red - Redirect by uli...

Jobs

Math

History

Document: Done (92.975 secs)

voyages-sncf.com

Identifiant commerçant 055204944722232

Référence de la transaction 995741

Montant de la transaction 192,70 Euro

Les symboles indiquent que votre transaction est sécurisée, vous pouvez remplir votre formulaire en toute confiance.

N° de carte : Expire fin : 01-Janvier / 2003

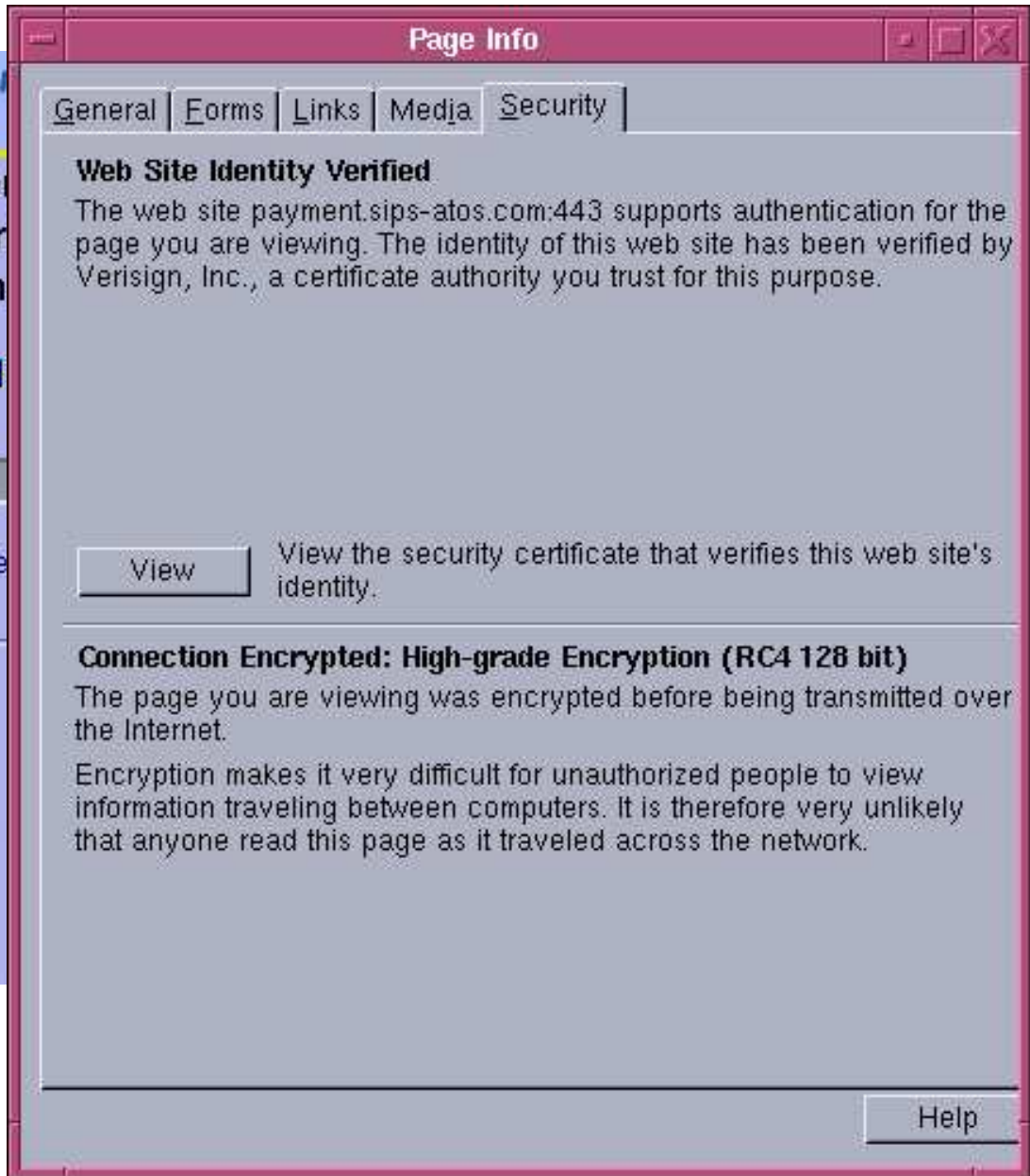
Vous avez complété correctement le formulaire, vous pouvez **VALIDER**

ANNULER

Copyright © 2003, all rights reserved

VISA

Vérifions les informations de sécurité



The image shows a screenshot of a web browser's "Page Info" dialog box, specifically the "Security" tab. The dialog box is overlaid on a webpage from "voyages-sips.com". The webpage content includes a shopping cart icon, the text "Identifiant comme", "Référence de la tr", "Montant de la tran", "Les symboles confiance.", "N° de carte :", and "Vous avez complété corre". The "Page Info" dialog box has tabs for "General", "Forms", "Links", "Media", and "Security". The "Security" tab is active and displays the following information:

Web Site Identity Verified
The web site payment.sips-atos.com:443 supports authentication for the page you are viewing. The identity of this web site has been verified by Verisign, Inc., a certificate authority you trust for this purpose.

View the security certificate that verifies this web site's identity.

Connection Encrypted: High-grade Encryption (RC4 128 bit)
The page you are viewing was encrypted before being transmitted over the Internet.
Encryption makes it very difficult for unauthorized people to view information traveling between computers. It is therefore very unlikely that anyone read this page as it traveled across the network.

The background webpage also features a "VISA" logo and the text "tre formulaire en toute".

... et notamment le certificat



The screenshot shows a 'Certificate Viewer' window for the domain 'payment.sips-atos.com'. The window has two tabs: 'General' and 'Details'. The 'General' tab is active, displaying the following information:

This certificate has been verified for the following uses:

- SSL Server Certificate
- Email Signer Certificate
- Email Recipient Certificate

Issued To

Common Name (CN)	payment.sips-atos.com
Organization (O)	ATOS ORIGIN
Organizational Unit (OU)	Services
Serial Number	6D:80:1D:66:8E:E9:83:C4:C7:E9:02:44:E3:03:4A:F5

Issued By

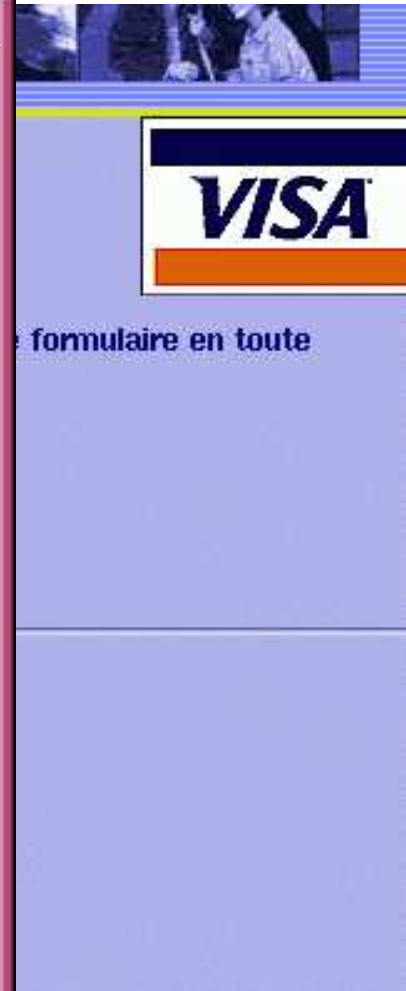
Common Name (CN)	<Not Part Of Certificate>
Organization (O)	RSA Data Security, Inc.
Organizational Unit (OU)	Secure Server Certification Authority

Validity

Issued On	01/09/2003
Expires On	02/05/2004

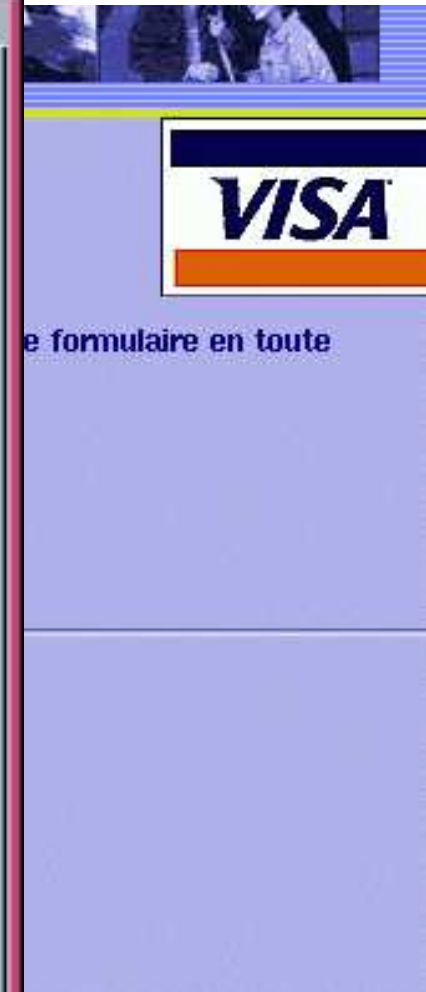
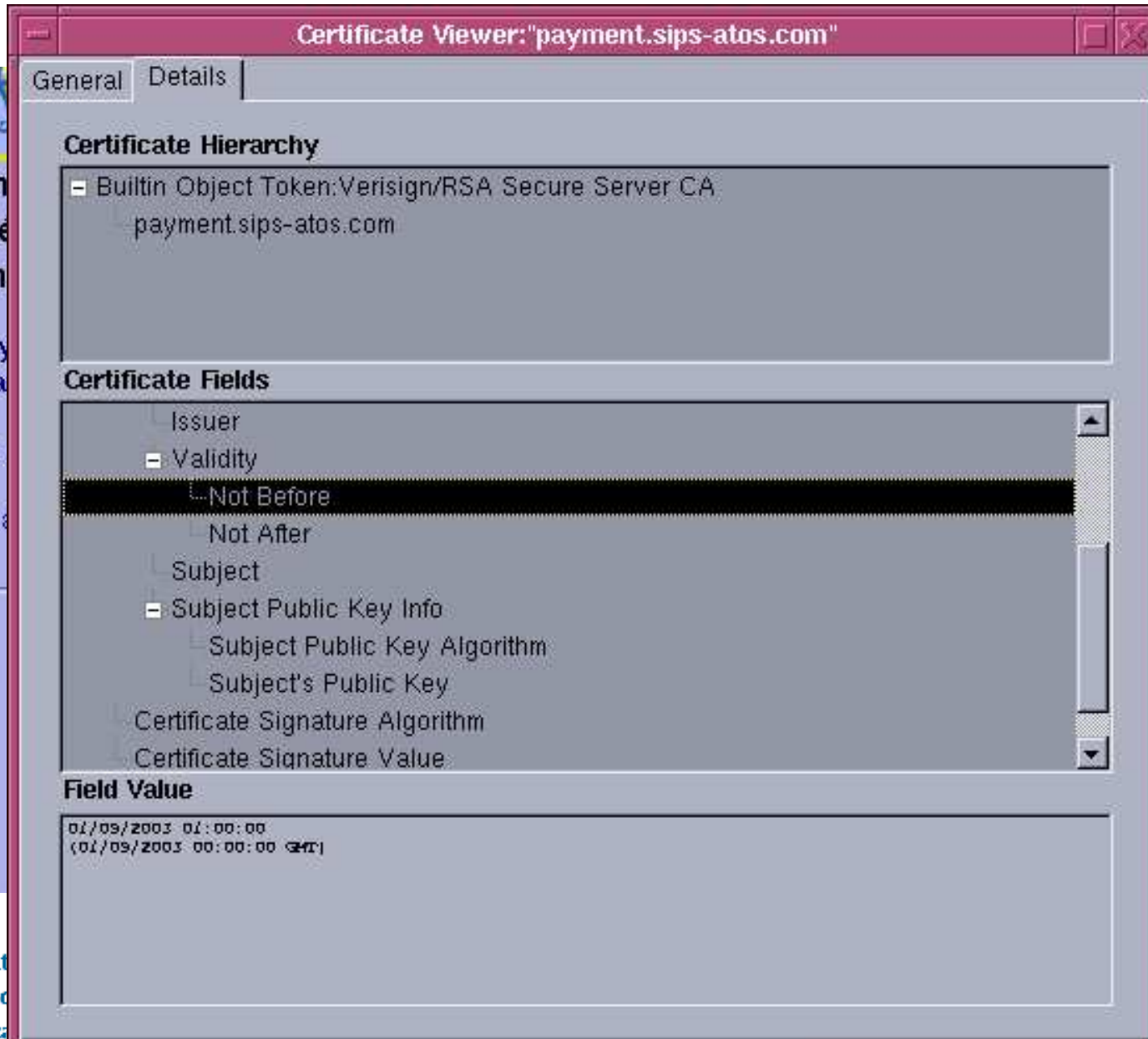
Fingerprints

SHA1 Fingerprint	DA:30:0E:1C:08:15:FB:D1:68:89:95:E9:A1:F1:A6:42:13:39:EC:27
MD5 Fingerprint	95:72:C2:12:23:AA:9B:A2:FE:1C:AB:23:05:B1:F2:FA



The image shows a partial view of a Visa website. At the top, there is a small photograph of a person. Below it is the prominent Visa logo, consisting of the word 'VISA' in a bold, italicized font with a horizontal line through it, set against a blue and orange background. Below the logo, the text 'formulaire en toute' is visible, suggesting a form or document related to the Visa service.

... et même les détails !



Quelques références

- L'incontournable :
Bruce Schneier.
Applied Cryptography.
John Wiley and Sons, 1996.
- Pour les cryptologues en herbe :
Shafi Goldwasser et Mihir Bellare.
Lecture Notes on Cryptography.
www-cse.ucsd.edu/~mihir/papers/gb.ps.gz.
Août 1999.