

# IRM : RFID Authentication Resistant To Compromised Readers

Tania Martin

*Promoteur* : Pr. Gildas Avoine

Octobre 2008



**UCL**  
Université  
catholique  
de Louvain



# Plan de l'exposé

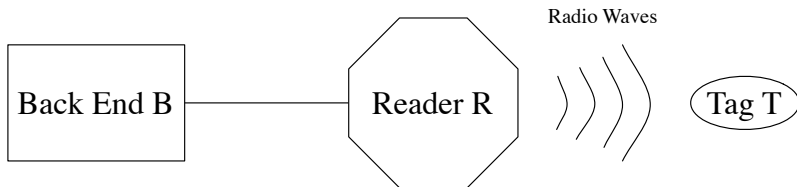
- 1 Le contexte
  - La RFID
  - Authentification
- 2 Quels sont les problèmes de la RFID ?
  - Introduction aux problèmes
- 3 Présentation de nos protocoles
  - Nos préliminaires
  - Notre proposition
  - Un exemple détaillé : Protocole 1 (chiffrement symétrique)
  - Comparaison
- 4 Et après ?
  - La thèse
  - Bilan du stage

# Définitions

**RFID** = Radio Frequency Identification

- Technologie sans contact
- Identifier et/ou authentifier des objets à distance à travers un canal radio
- Exploité dans de nombreux domaines

# Architecture



# Les tags


## ● Les **actifs**

- ▶ Source individuelle d'énergie → batterie
- ▶ Possibilité d'initier une communication
- ▶ Capacités cryptographiques → protocoles complexes
- ▶ Grande mémoire de stockage
- ▶ Portée de lecture = longue distance

## ● Les **passifs**

- ▶ Pas de batterie, alimentés par le lecteur
- ▶ Portée de lecture = 10 cm à quelques mètres
- ▶ Capacités cryptographiques en fonction du prix du tag

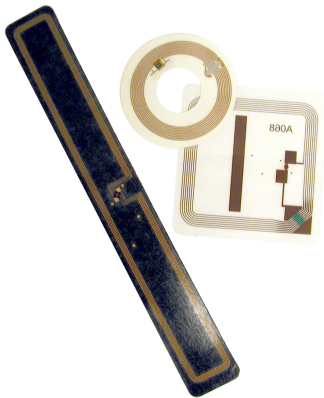
# Le prix des tags passifs



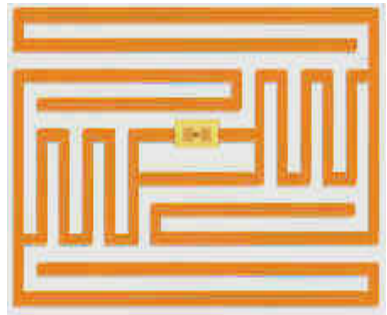
0.10 €	-->	XOR pas de cryptographie
0.80 €	-->	cryptographie symétrique avec - logique câblée - microprocesseur
2 €	-->	cryptographie asymétrique avec microprocesseur

# Où trouve-t-on des tags ?

## Les bibliothèques

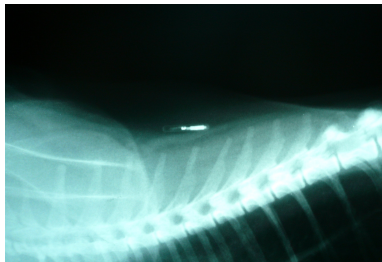


## La grande distribution : ici Wal-Mart

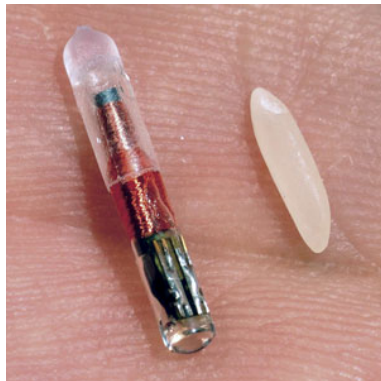


## Où trouve-t-on des tags ?

Les implants d'animaux



Un tag RFID aussi grand qu'un grain de riz



# Où trouve-t-on des tags ?

## Les transports de Bruxelles



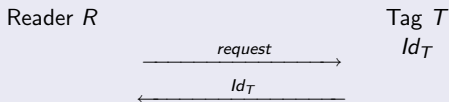
## Le contrôle d'accès



# Authentication vs. Identification

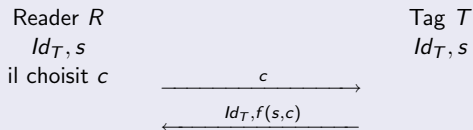
## Identification

Le lecteur demande au tag son identité.



## Authentication

Le tag doit **prouver** son identité au lecteur.



# Quatre problèmes majeurs en authentification

- 1 Cryptographie à bas coût
- 2 Privacy
- 3 Attaques par relais
- 4 Lecteurs compromis

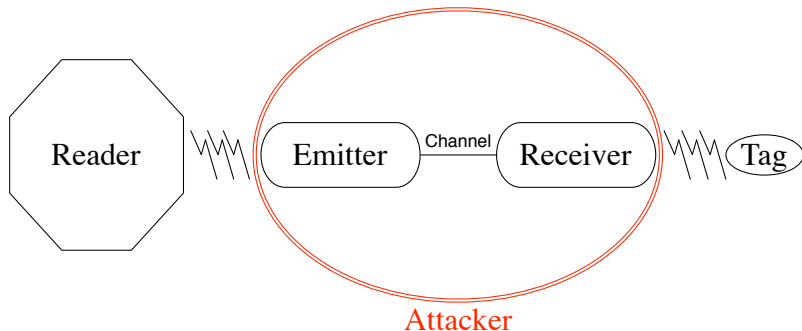
# Cryptographie à bas coût

- Clés de chiffrement trop courtes
  - ▶ Dans la plupart des contrôles d'accès, **une clé unique** est partagée entre les lecteurs et les cartes
- Algorithmes de chiffrement non publics
  - ▶ Reverse engineering = cassage du système entier
    - Mifare → CRYPTO1
    - Texas Instrument → DST40

# Privacy

- Protection des données personnelles
  - ▶ Peuvent-elles être révélées ?
  
- Traçabilité malicieuse
  - ▶ Peut-on reconnaître un tag à partir des messages d'une communication ?

# Attaques par relais



Difficile à prendre en compte dans un protocole cryptographique

## Lecteurs compromis

- L'attaquant s'est emparé des données contenues dans un lecteur
- Problème pas pris en compte à l'heure actuelle
- Lecteurs faciles d'accès, lors d'un déploiement à grande échelle

## Nos propriétés, hypothèse et but

### But

**Rétablir la sécurité d'un système sans avoir besoin de changer physiquement les tags**

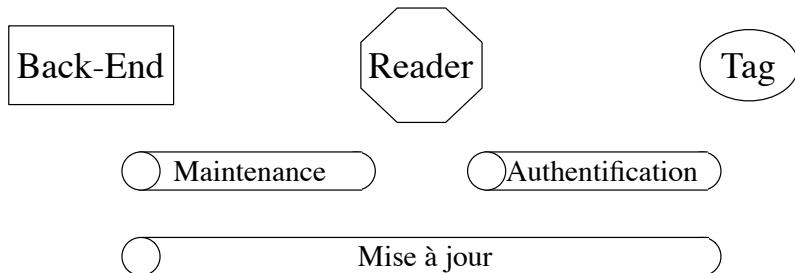
### Propriétés

- Protocole implémentable en pratique
- Les lecteurs se connectent sporadiquement à la back-end
  - ▶ Cette communication se fait sur un canal sécurisé
- Déploiement dans de grands systèmes

### Hypothèse de départ

- Le système détecte un lecteur compromis

# L'idée générale



## Nos trois protocoles

- Protocole 1 : Chiffrement symétrique
- Protocole 2 : Chiffrement asymétrique + hachage + signature
- Protocole 3 : Authentification + chiffrement symétrique

# Initialisation

**Back-end  $B$**

$Id_T, S, K, k$

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$

- $Id_T$  : identifiant de  $T$ .

# Initialisation

**Back-end  $B$**

$Id_T, S, K, k$

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$

- $Id_T$  : identifiant de  $T$ .
- $S$  : état courant de  $T$ .

# Initialisation

**Back-end  $B$**

$Id_T, S, K, k$

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$

- $Id_T$  : identifiant de  $T$ .
- $S$  : état courant de  $T$ .
- $K$  : clé symétrique partagée entre  $B$  et  $T$ .

## Initialisation

**Back-end  $B$**

$Id_T, S, K, k$

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$

- $Id_T$  : identifiant de  $T$ .
- $S$  : état courant de  $T$ .
- $K$  : clé symétrique partagée entre  $B$  et  $T$ .
- $k$  : clé symétrique partagée entre  $R$  et  $T$ .

## Initialisation

**Back-end  $B$**

$Id_T, S, K, k$

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$

- $Id_T$  : identifiant de  $T$ .
- $S$  : état courant de  $T$ .
- $K$  : clé symétrique partagée entre  $B$  et  $T$ .
- $k$  : clé symétrique partagée entre  $R$  et  $T$ .
- $M$  : donnée de stockage pour les mises à jour.

## Initialisation

**Back-end  $B$**

$Id_T, S, K, k$

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$

- $Id_T$  : identifiant de  $T$ .
- $S$  : état courant de  $T$ .
- $K$  : clé symétrique partagée entre  $B$  et  $T$ .
- $k$  : clé symétrique partagée entre  $R$  et  $T$ .
- $M$  : donnée de stockage pour les mises à jour.

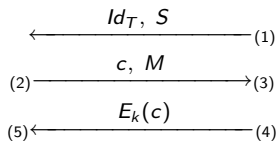
# Authentication

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$



- ①  $T$  envoie son identifiant + son état courant.

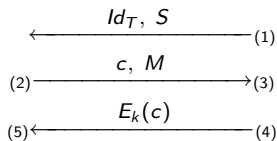
# Authentification

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$



- 1  $T$  envoie son identifiant + son état courant.
- 2  $R$  contrôle l'état de  $T$ , et envoie :
  - ▶ un challenge  $c$ ,
  - ▶ la donnée  $M$  (l'update), si  $T$  nécessite une mise à jour.

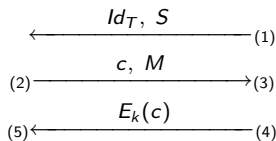
# Authentification

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$



- 1  $T$  envoie son identifiant + son état courant.
- 2  $R$  contrôle l'état de  $T$ , et envoie :
  - ▶ un challenge  $c$ ,
  - ▶ la donnée  $M$  (l'update), si  $T$  nécessite une mise à jour.
- 3 Si  $T$  reçoit  $M$  alors il met à jour  $S$  et  $k$ .

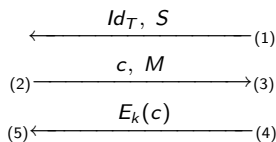
# Authentification

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$



- 1  $T$  envoie son identifiant + son état courant.
- 2  $R$  contrôle l'état de  $T$ , et envoie :
  - ▶ un challenge  $c$ ,
  - ▶ la donnée  $M$  (l'update), si  $T$  nécessite une mise à jour.
- 3 Si  $T$  reçoit  $M$  alors il met à jour  $S$  et  $k$ .
- 4  $T$  chiffre  $c$  avec la clé secrète  $k$ , et l'envoie à  $R$ .

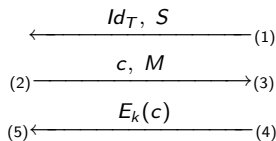
# Authentification

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$



- 1  $T$  envoie son identifiant + son état courant.
- 2  $R$  contrôle l'état de  $T$ , et envoie :
  - ▶ un challenge  $c$ ,
  - ▶ la donnée  $M$  (l'update), si  $T$  nécessite une mise à jour.
- 3 Si  $T$  reçoit  $M$  alors il met à jour  $S$  et  $k$ .
- 4  $T$  chiffre  $c$  avec la clé secrète  $k$ , et l'envoie à  $R$ .
- 5  $R$  déchiffre  $E_k(c)$  avec  $k$  et vérifie la validité de  $c$ .

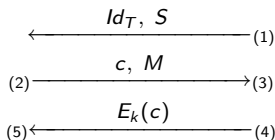
# Authentification

**Reader  $R$**

$Id_T, S, k, M$

**Tag  $T$**

$Id_T, S, K, k$



- 1  $T$  envoie son identifiant + son état courant.
- 2  $R$  contrôle l'état de  $T$ , et envoie :
  - ▶ un challenge  $c$ ,
  - ▶ la donnée  $M$  (l'update), si  $T$  nécessite une mise à jour.
- 3 Si  $T$  reçoit  $M$  alors il met à jour  $S$  et  $k$ .
- 4  $T$  chiffre  $c$  avec la clé secrète  $k$ , et l'envoie à  $R$ .
- 5  $R$  déchiffre  $E_k(c)$  avec  $k$  et vérifie la validité de  $c$ .

# Update

**Back-End  $B$**

$Id_T, S, K, k$

**Reader  $R$**

$Id_T, S, k, M$

(1)  $\xrightarrow{Id_T, S_{up}, k_{up}, E_K(S_{up}, k_{up})}$  (2)

①  $B$  envoie à  $R$

- ▶ les mises à jour de l'état de  $T$  et de la clé secrète  $k$  (partagée entre  $R$  et  $T$ ),
- ▶ un chiffrement de celles-ci avec la clé secrète  $K$  (partagée entre  $B$  et  $T$ ).

# Update

**Back-End  $B$**

$Id_T, S, K, k$

**Reader  $R$**

$Id_T, S, k, M$

(1)  $\xrightarrow{Id_T, S_{up}, k_{up}, E_K(S_{up}, k_{up})}$  (2)

- ①  $B$  envoie à  $R$ 
  - ▶ les mises à jour de l'état de  $T$  et de la clé secrète  $k$  (partagée entre  $R$  et  $T$ ),
  - ▶ un chiffrement de celles-ci avec la clé secrète  $K$  (partagée entre  $B$  et  $T$ ).
- ②  $R$  met à jour  $S$  et  $k$ , et stocke dans  $M$  le chiffrement  $E_K(S_{up}, k_{up})$ .

# Update

**Back-End  $B$**

$Id_T, S, K, k$

**Reader  $R$**

$Id_T, S, k, M$

(1)  $\xrightarrow{Id_T, S_{up}, k_{up}, E_K(S_{up}, k_{up})}$  (2)

- ①  $B$  envoie à  $R$ 
  - ▶ les mises à jour de l'état de  $T$  et de la clé secrète  $k$  (partagée entre  $R$  et  $T$ ),
  - ▶ un chiffrement de celles-ci avec la clé secrète  $K$  (partagée entre  $B$  et  $T$ ).
- ②  $R$  met à jour  $S$  et  $k$ , et stocke dans  $M$  le chiffrement  $E_K(S_{up}, k_{up})$ .

## Nos trois protocoles

Objectifs : Quels sont les résultats en terme de

- Porte logiques ?
- Communication ?
- Consommation de puissance ?

P1	Symétrique	⇒	AES	=	3500 GE	[CHES'04]
			DES Variant	=	2300 GE	[FSE'07]
P2	Asymétrique	⇒	RSA	=	————	
			NtruEncrypt	=	3000 GE	[ESAS'04]
	Hachage	⇒	SHA-1	=	5500 GE	[RFIDSec'08]
			Autres	≥	7000 GE	[CHES'08]
P3	Authentification	⇒	GPS	=	1700 GE	[CT-RSA'07]
	Symétrique	⇒	idem			

## Qu'est ce que GPS ?

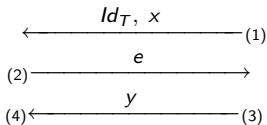
**GPS** = Protocole d'authentification Zero-Knowledge

**Reader  $R$**

$Id_T, l$

**Tag  $T$**

$Id_T, s, l, coupons$



- 1  $T$  choisit un coupon ( $r, x = g^r \pmod n$ ), et envoie à  $R$  son identifiant et  $x$ .

## Qu'est ce que GPS ?

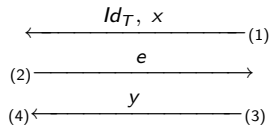
**GPS** = Protocole d'authentification Zero-Knowledge

**Reader  $R$**

$Id_T, l$

**Tag  $T$**

$Id_T, s, l, coupons$



- 1  $T$  choisit un coupon ( $r, x = g^r \pmod n$ ), et envoie à  $R$  son identifiant et  $x$ .
- 2  $R$  envoie un challenge  $e$  à  $T$ .

## Qu'est ce que GPS ?

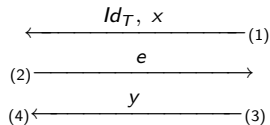
**GPS** = Protocole d'authentification Zero-Knowledge

**Reader  $R$**

$Id_T, l$

**Tag  $T$**

$Id_T, s, l, coupons$



- 1  $T$  choisit un coupon ( $r, x = g^r \pmod n$ ), et envoie à  $R$  son identifiant et  $x$ .
- 2  $R$  envoie un challenge  $e$  à  $T$ .
- 3  $T$  répond  $y = r + e \times s$ .

## Qu'est ce que GPS ?

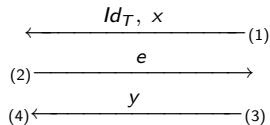
**GPS** = Protocole d'authentification Zero-Knowledge

**Reader R**

$Id_T, l$

**Tag T**

$Id_T, s, l, coupons$



- 1  $T$  choisit un coupon ( $r, x = g^r \pmod n$ ), et envoie à  $R$  son identifiant et  $x$ .
- 2  $R$  envoie un challenge  $e$  à  $T$ .
- 3  $T$  répond  $y = r + e \times s$ .
- 4  $R$  vérifie :
  - ▶  $g^y \times l^e \pmod n = x$
  - ▶  $y \in [0, A + \Phi[$

## Qu'est ce que GPS ?

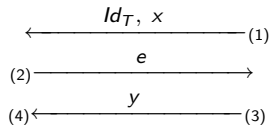
**GPS** = Protocole d'authentification Zero-Knowledge

**Reader R**

$Id_T, l$

**Tag T**

$Id_T, s, l, coupons$



- 1  $T$  choisit un coupon ( $r, x = g^r \pmod n$ ), et envoie à  $R$  son identifiant et  $x$ .
- 2  $R$  envoie un challenge  $e$  à  $T$ .
- 3  $T$  répond  $y = r + e \times s$ .
- 4  $R$  vérifie :
  - ▶  $g^y \times l^e \pmod n = x$
  - ▶  $y \in [0, A + \Phi[$

## Avantages et inconvénients de GPS

- ⊕ Protocole très léger → peu de calculs pour le tag
- ⊕ Complexité faible pour le lecteur
- ⊖ Pas de privacy pour le tag
- ⊖ Problème des coupons → déni de service

## Objectifs à court terme

- Définir des critères de sélection
- Etude de nos solutions en pratique
- Implémentation

## Résumé du travail effectué

- Etat de l'art des protocoles Challenge/Réponse et GPS
- Conception de 3 protocoles
- La Mifare (cartes à puce sans contact, produite par NXP)
  - ▶ Vendue à plus d'un milliard d'exemplaires (billetterie, etc.)
  - ▶ Failles dans la conception  
CRYPTO1 → 4 attaques publiées en 2008
  - ▶ Article pour MISC Magazine (Hors-Série Octobre 2008)