

Introduction

Dans ce chapitre, nous présentons des notions de base de la simulation et les diagrammes utilisés du modèle UML, puis nous présentons une spécification en détail de la simulation des protocoles d'authentification RFID et nous allons modéliser l'aspect statique et dynamique de la simulation par le biais de la conception en diagramme de classe et en diagramme de séquence pour chaque protocole.

1. Simulation :

La simulation est un outil d'aide à la décision très utilisé par les concepteurs et les gestionnaires des systèmes complexes. Elle consiste à construire un modèle d'un système réel (physique, économique, humain ... etc.) et à conduire des expériences sur ce modèle afin de bien comprendre le comportement de ce système et d'en améliorer les performances. Peut être utilisé pour études, prédiction, réalisation ...etc. [Med]

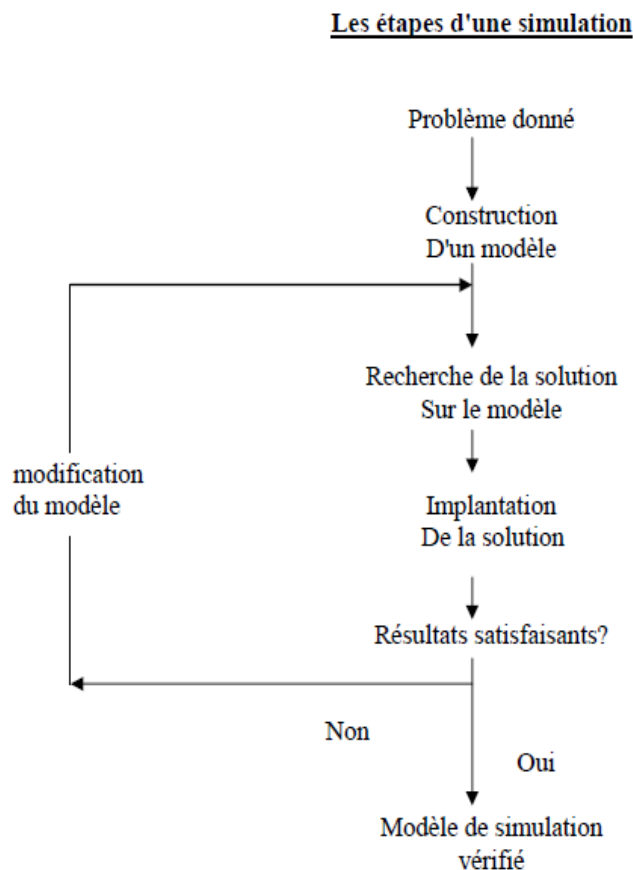


Figure N° III.1 : les étapes d'une simulation [Med]

1.1. Méthodes de simulation:

Il existe deux catégories de méthodes de simulation: la simulation continue et la simulation discrète.

- La Simulation continue : Si le modèle du système contient en grande partie des variables continues qui évoluent dans le temps, ce modèle conduit à un système d'équations différentielles. Fréquemment, il se trouve que le système d'équations ne peut pas être résolu analytiquement. On utilise alors la simulation dite continue, ou des combinaisons de méthodes numériques essaient d'aboutir à la solution du modèle.
- La simulation discrète, on l'utilise pour simuler le fonctionnement des systèmes discrets et stochastiques. C'est la méthode de simulation la plus développée jusqu'à présent et englobe trois stratégies: la simulation à événements discrets, la simulation des activités et la simulation des processus. Parmi stratégies de simulation discrète, nous intéressons par la simulation des événements.

1.1.1. La Simulation des événements

Le système est étudié à des instants discrets, non fixes; c'est-à-dire lorsque des événements surviennent et altèrent l'état du système. Ces changements de l'état du système peuvent générer à leur tour de nouveaux événements et ainsi de suite.[Med].

1.1.2. Le temps en simulation discrète:

Le temps est un paramètre essentiel dans tout modèle de simulation. L'écoulement du temps est mesuré par une horloge de simulation (indépendante de l'horloge du système). L'unité de temps doit être choisie en fonction des besoins de la simulation, elle peut varier de la nanoseconde à un siècle (évolution des rivières). Dans la simulation discrète, l'horloge avance par bonds irréguliers, c'est-à-dire elle saute d'un événement à un autre.

1.1.3. Les arrivées dans le système:

En général, la durée séparant deux arrivées successives de tag, et la durée de service pour un tag sont générées suivant une distribution de probabilités déterminée. Le choix de cette distribution se fait en tenant compte des phénomènes observés dans la réalité.

1.1.4. Simulation orientée clients:

Les clients sont considérés comme des entités autonomes, on peut calculer des statistiques pour chaque tag comme par exemple: le temps qu'il a passé dans la queue, dans le service ...etc.

1.2. Evaluation des performances

Evaluer un système, c'est mesurer son aptitude à résoudre l'ensemble des problèmes pour les quels on l'utilise ou on envisage de l'utiliser.

La performance est une ensemble des indications chiffrées caractérisant les possibilités optimales d'un system; comme par exemple: la vitesse de calcul, la rapidité de transfert ...etc. [Med]

De manière générale, l'évaluation de performances est consacrée à l'analyse de l'efficacité des systèmes qui traitent des flots de clients (les réseaux de télécommunication, ferroviaires, routiers, etc.), des machines qui effectuent des traitements complexes de tâches, etc.

L'évaluation des performances peut intervenir à deux niveaux:

- En conception (dimensionnement): Le système n'existe pas encore, il s'agit d'évaluer les performances de ce système avant que la réalisation soit engagée.
- En exploitation (modification ou test): Le système réel existe, est on souhaite le modifier pour améliorer son fonctionnement.

1.3. La simulation pour l'évaluation des performances

C'est une technique largement utilisée pour l'évaluation des performances. Elle présente l'avantage par rapport aux méthodes analytiques de traduire d'une manière plus réaliste le comportement du système à évaluer et nécessite moins de simplifications et quasiment pas d'hypothèses spécifiques.

La simulation numérique a été, pour la première fois, utilisée en 1950 dans la planification stratégique au niveau militaire. Elle n'a gagné sa popularité dans les domaines manufacturiers et de services qu'au début des années quatre-vingt.

1.4. La performance de protocole RFID :

L'évaluation de performances s'intéresse au calcul de quelques paramètres caractéristiques d'un système. Ces paramètres diffèrent d'un système à un autre. Dans les protocoles d'authentification des systèmes RFID étudiés, le temps de réponse est un

paramètre de performance important. Il mesure le temps qui sépare l'émission, de la réception par le destinataire (temps des fonctions (TAG ou READER...), temps d'attente au niveau des différents nœuds, temps de transmissions sur les lignes, temps de recomposition du message au nœud final).

1.5. Les hypothèses de simulation :

La communication entre le serveur et le lecteur est sécurisée, c'est-à-dire que le canal de communication est un canal privé. Contrairement à ceci, la communication entre le tag et le lecteur est quant à elle insécurisée et basée sur les ondes radiofréquences. Notre simulation particulière touche les transmissions sur le canal lecteur-tag seulement, car ce dernier est public, et peut subir des attaques par un adversaire.

On suppose que les protocoles étudiés sont des protocoles sans collision. Ainsi, dans notre simulateur, le protocole utilise le service pendant l'exécution complète du protocole, c'est-à-dire le service reste occupé de début de protocole avec un tag jusqu'à la fin de l'authentification du tag et l'authentification du lecteur.

2. Modélisation avec UML :

2.1. Définition :

UML (*Unified Modeling Language*) est un langage ou formalisme de modélisation orienté objet qui représente un moyen de spécifier et représenter les composantes d'un système informatique. UML est un standard car à partir de 1997, il est devenu une norme de l'Object Management Group (OMG). Parmi les objectifs d'UML : être indépendant des langages de programmation et être adapté à toutes les phases du développement [Mah07].

2.2. Les objectifs d'UML :

- Représenter des systèmes entiers.
- Etablir un couplage explicite entre les concepts et les artefacts exécutables.
- Prendre en compte les facteurs d'échelle.
- Créer un langage de modélisation utilisable à la fois par les humains et les machines.
- Recherche d'un langage commun (utilisables par toutes les méthodes, adapté à toutes les phases de développement, compatible avec toutes les techniques de réalisation).

2.3. Présentation générale des diagrammes d'UML :

Un modèle est une représentation simplifiée d'un problème ou une abstraction de la réalité. UML permet d'exprimer les modèles objets à travers un ensemble de diagrammes, ces dernières sont des moyens de description des objets ainsi que des liens qui les relient.

Parmi les neufs diagrammes de l'UML, on utilise le diagramme de cas d'utilisation, le diagramme de classes, et le diagramme de Séquence.

Le diagramme de classe est le point central dans un développement orienté objet. En analyse, il a pour objectif de décrire la structure des entités manipulées par les utilisateurs [AmMy].

Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes, les cas d'utilisation, ayant un sens pour les acteurs. Les cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système, ils sont donc une vision orientée utilisateur de ce besoin au contraire d'une vision informatique [Uap09].

Le diagramme de séquence montre les interactions entre les objets selon un point de vue temporel [Mu97].

Pour la réalisation des diagrammes de notre modèle, on utilise l'application UML (voir la figure III.2).

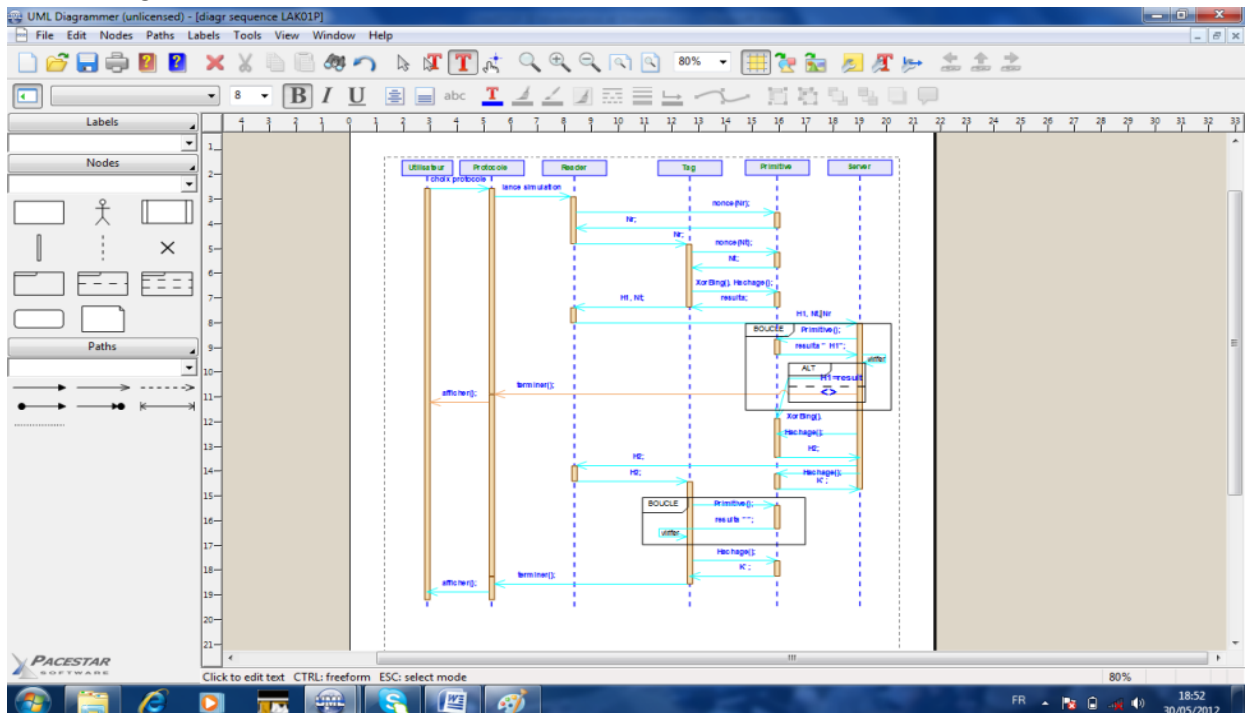


Figure N° III.2 : Représentation de langage de programmation UML2

3. Conception de simulation des protocoles d'Authentification RFID :

Dans cette section, nous présentons une spécification en détail de la simulation de chaque protocole d'authentification étudié utilisant les protocoles de fonction de hachage (RHLS, HMNB, RHLS, LAK, PAP), puis nous allons modéliser l'aspect statique et dynamique de la simulation le biais de la conception en diagramme de classe et en diagramme de séquence.

3.1. Spécification des besoins du simulateur

Pour la spécification, une représentation en diagramme de cas d'utilisation est nécessaire. Un seul acteur principal à partir de l'analyse des besoins, C'est l'utilisateur du simulateur. La (figure III.3) représente le diagramme de cas d'utilisation de notre système.

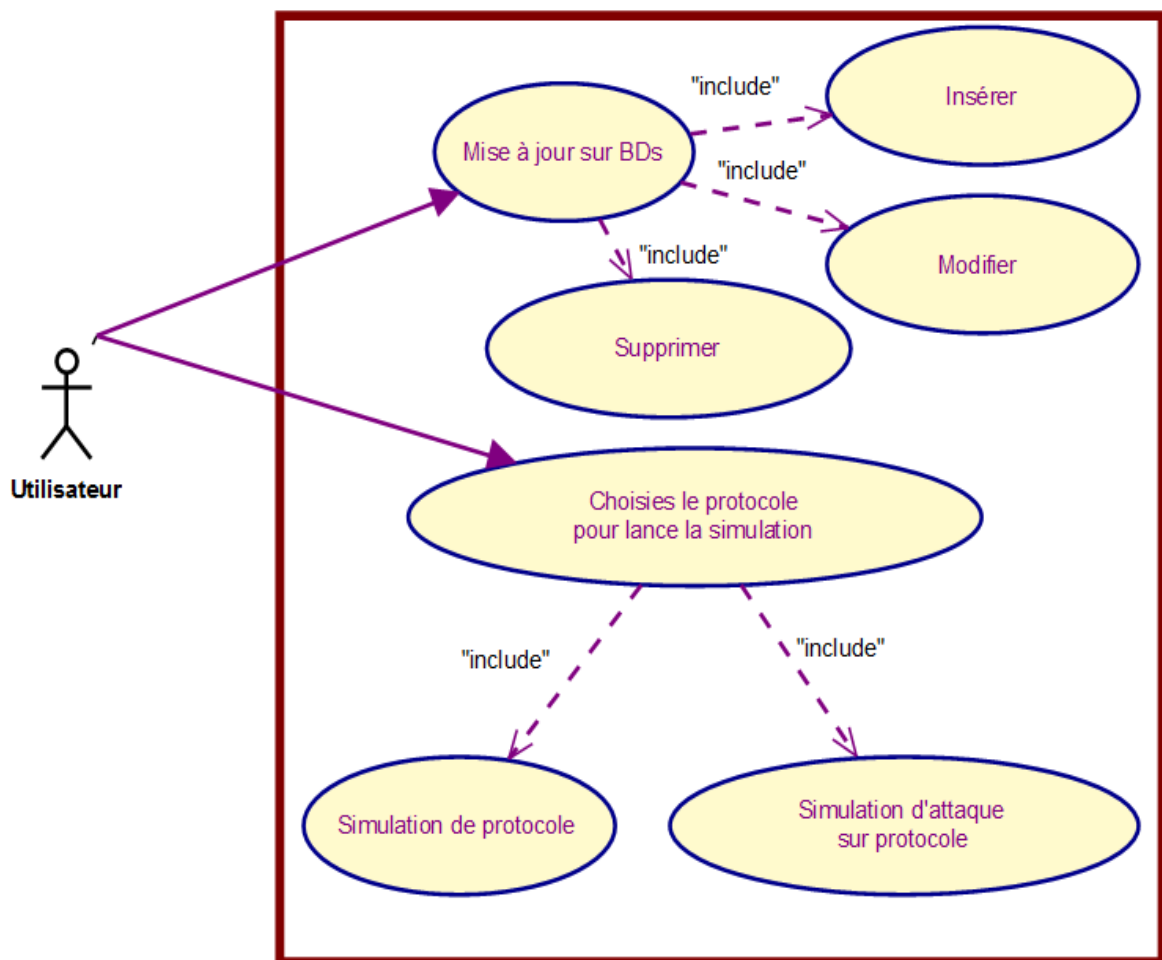


Figure N° III.3 : Diagramme de cas d'utilisation général.

3.2. Diagramme de classe du simulateur

(La figure III.4) représente le diagramme de classe. Ce diagramme décrit la structure des différentes classes nécessaires pour le déroulement de simulation. Il fait, il comporte six classes (même diagramme de classe pour les tous protocoles) :

Classe Protocole: cette classe représente la classe principale dans le système, que donnent les actions de système, et **cette** classe représente toutes les opérations sur la base de données, et représente les transmissions de les informations envoyés enter l'émetteur et le récepteur. Elle contient différentes méthode qui modélisent l'échange de données nécessaires.

Classe TAG : cette classe représente la carte que représente une personne, avez-vous le droit ou non, cette classe fait appel aux les méthodes (*nonce()*, *Xor()*, *Hachage()*)« calcule le hachage», ...) de la classe PRIMITIVE pour fait les opération de ce classe.

Classe READER : cette classe représente le lecteur de identifier du TAG (récepteur), cette classe fait appel aux les méthodes (*nonce()* « nonce une valeur aléatoire») de la classe PRIMITIVE pour fait les opération de ce classe.

Classe SERVER : cette classe représente le vérificateur de l'authentification du TAG. cette classe fait appel aux les méthodes (*Xor()*, *Hachage ()*)« calcule le hachage», ...) de la classe PRIMITIVE pour fait les opération de ce classe.

Classe INTRUS : cette classe représente l'intrus (attaquant). Elle contient différentes méthodes pour qu'un intrus les envoyés par l'émetteur sur le canal, Il change les informations envoyés entre le TAG et le READER.

Classe PRIMITIVE : cette classe représente les opérations utilisées dans le système.

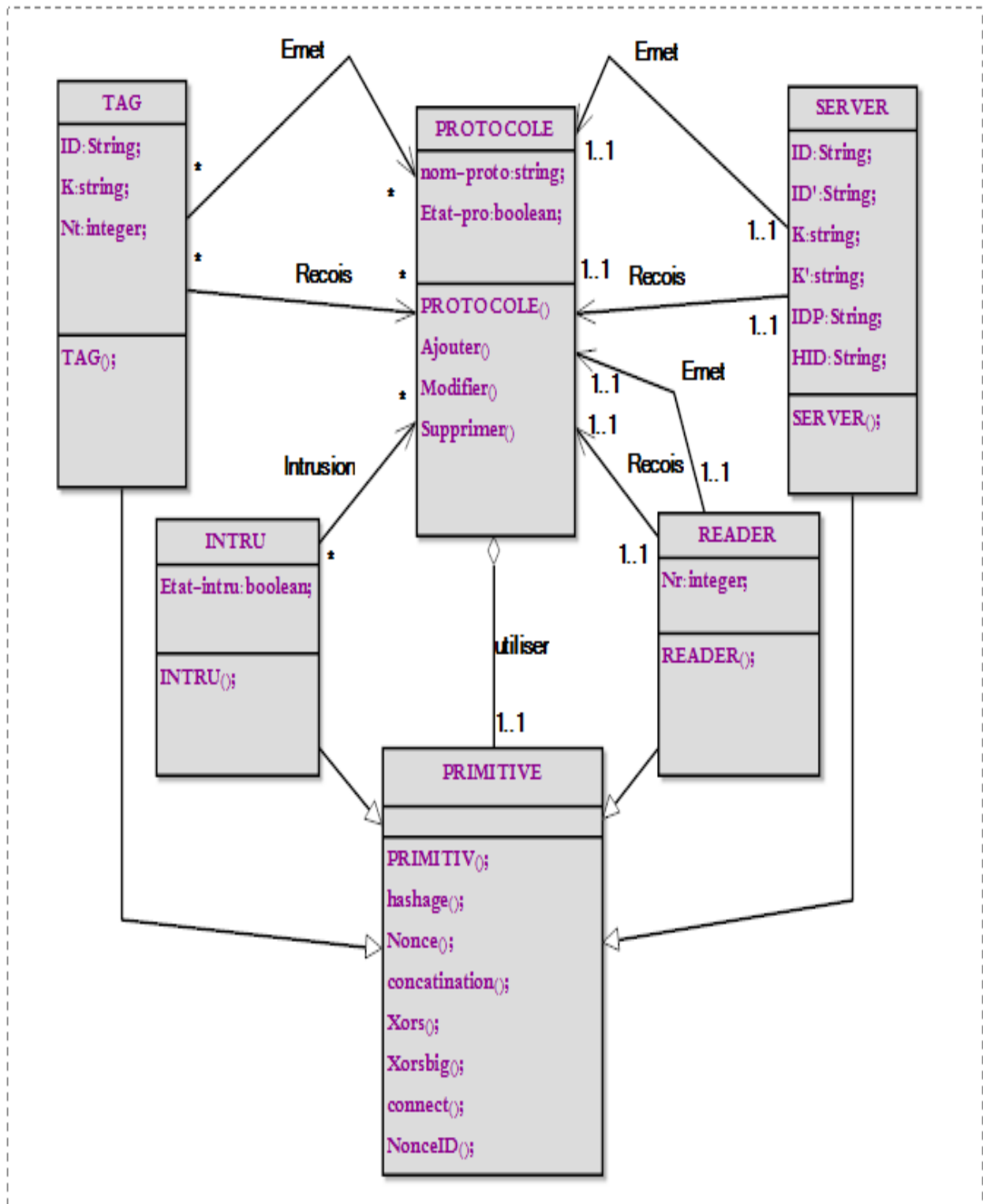


Figure N° III.4 :Diagramme de classe générale.

➤ **Les attributs :**

ID : stringe; ID' : stringe; IDP : stringe; HID : stringe;

K : stringe; K' : stringe; Nt :integer; Nr :integer;

Nom-proto:string; Etat-pro: Boolean; Etat-intru :string;

➤ Les méthodes :

TAG();

READER();

SERVER();

INTRU();

PROTOCOLE();

Ajouter();

Modofier() ;

Supprimer() ;

String hashage(String msg,String type) ;

int Nonce() ;

String NonceID() ;

String concatination(String x1,String x2) ;

int Xors(int x1,int x2) ;

BigInteger Xorsbig(BigInteger x1,BigInteger x2) ;

Connection connect(Connection con) ;

3.3. Diagramme de séquence des protocoles étudiés:

3.3.1. Protocole LAK :

Le diagramme de séquence donné par la (figure III.5) montre les interactions entre les objets des différentes classes selon la tempe dans le cas où l'utilisateur choisit de faire une simulation avec affichage numérique. Selon un point de vue temporel.

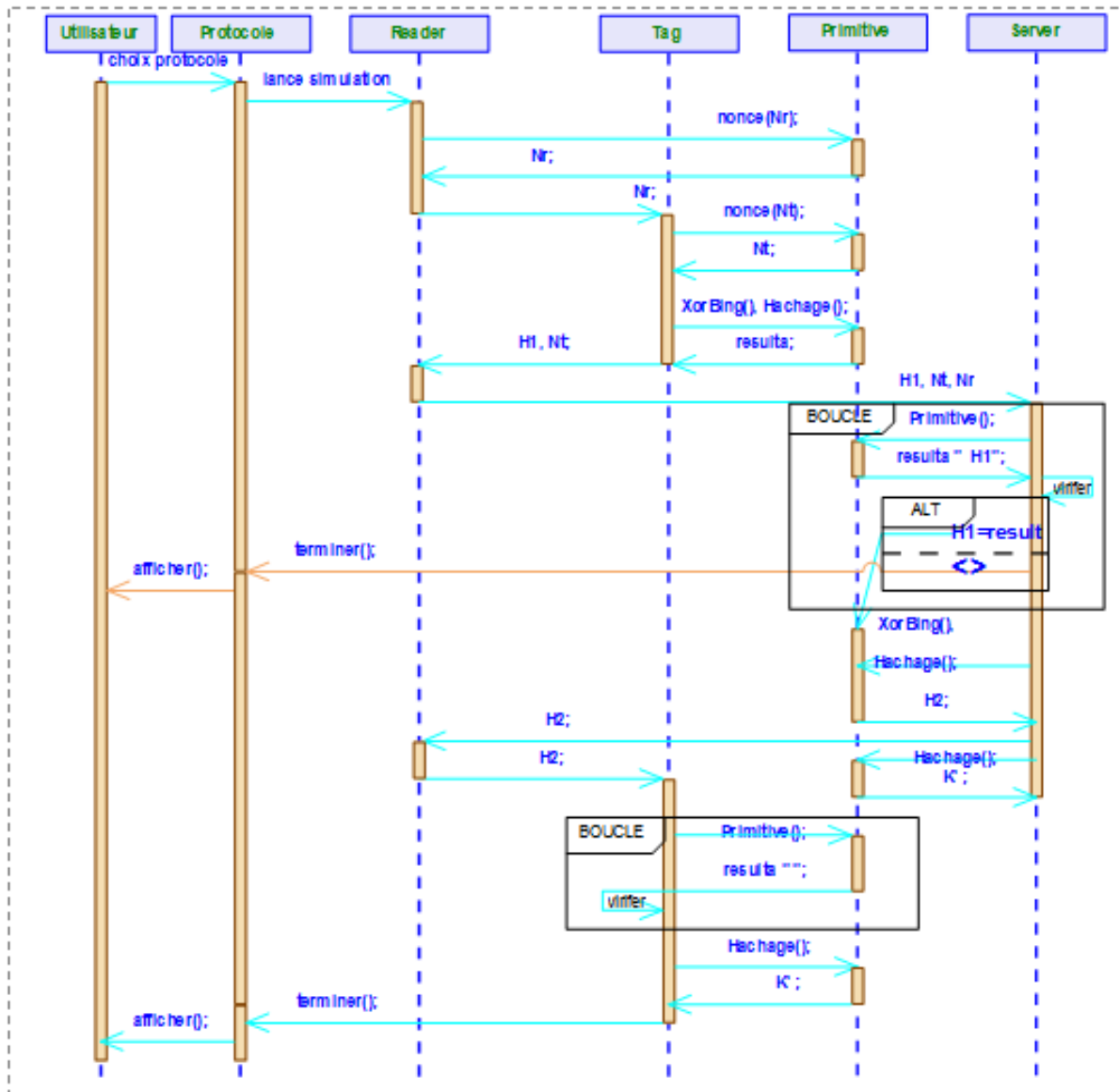


Figure N° III.5: Diagramme de séquence du protocole LAK.

Description :

✓ READER

READER produit d'une chaîne hexadécimale aléatoire Nr et l'envoie au tag,

✓ TAG

TAG reçu la valeur Nr par READER, les informations transmises par le tag à chaque fois qu'elle est interrogée se compose d'une valeur aléatoire Nt et $H1 = \text{valeur du hachage } H(Nr \oplus Nt \oplus K)$ où K est la clé qui est l'identifiant statique du tag, puis envoie " Nt et H1 " à le READER.

✓ READER

READER reçu les valeurs " Nt et H1 " et l'envoies au SERVER.

✓ SERVER

SERVER vérifie H1 « Si valeur $H1 = \text{valeur du hachage } H(Nr \oplus Nt \oplus K)$ alors (l'authentification du TAG est succès) », et la valeur $H2 = \text{valeur du hachage } H(H(Nr \oplus Nt \oplus K) \oplus K \oplus Nr)$, et l'envoi au READER.

La mise à jour est faite après la dernière transition (fin de transmission) au niveau du SERVER, dans ce cas K'_0 égal la valeur initiale de K avant sa mise à jour, et le nouveau K' égal $H(K)$.

✓ READER

READER reçu la valeur H2 et l'envoie au TAG.

✓ TAG

TAG vérifie H2 « Si la valeur de $H2 = \text{valeur du hachage } H(H(Nr \oplus Nt \oplus K) \oplus K \oplus Nr)$ alors l'authentification du READER est succès ».

.Enfin, Le TAG fait la mise à jour, dans ce cas le nouveau K' égal $H(K)$.

3.3.2. Protocole LAK (Avec attaque) :

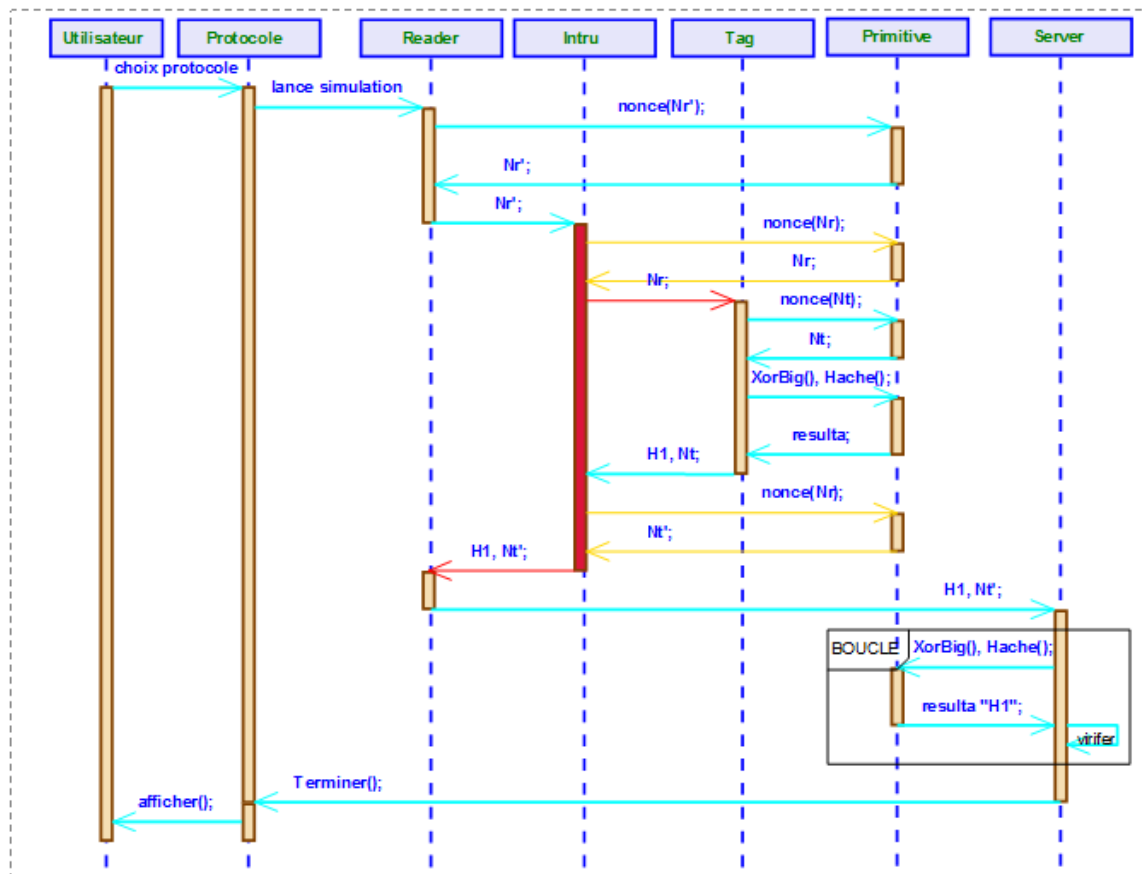


Figure N° III.6: Diagramme de séquence du protocole LAK « avec attaque ».

Description:✓ **READER**

READER produit d'une chaîne hexadécimale aléatoire Nr' , et envoie ces états au TAG,

✓ **INTRU**

Il prend la valeur Nr , puis produit d'une autre chaîne hexadécimale aléatoire Nr , et l'envoie au tag,

✓ **TAG**

TAG reçu la valeur Nr à travers de l'INTRUS, les informations transmises par le tag à chaque fois qu'elle est interrogée se compose d'une valeur aléatoire Nt et la valeur $H1 = \text{valeur du hachage } H(Nr \oplus Nt \oplus K)$ où K est la clé qui est l'identifiant statique du tag, puis envoie " Nt et $H1$ " au READER.

✓ **INTRU**

Il prendre les valeurs N_t et H_1 et changer la valeur N_t par $(nr' \oplus nr \oplus nt)$, et envoie ces valeurs au READER,

✓ **READER**

READER reçu les valeurs N_t et H_1 et l'envoie au SERVER.

✓ **SERVER**

SERVER vérifie H_1 « Si la valeur H_1 = valeur du hachage $H(Nr \oplus N_t \oplus K)$ Alors (l'authentification de l'Intrus est succès) ».

3.3.3 Protocole RHLS :

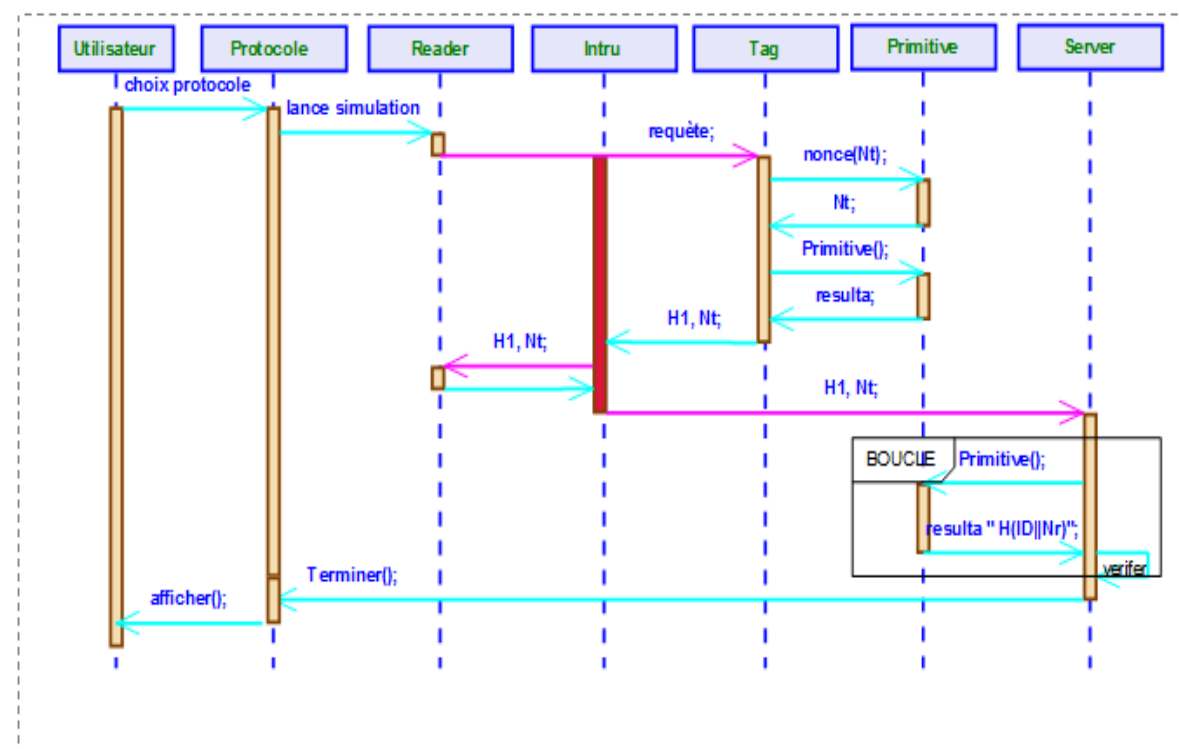


Figure N° III.6:Diagramme de séquence détaillé de protocole RHLS.

Description :

✓ **TAG**

TAG est composé d'une valeur aléatoire N_t et utilise l'opération concaténation (\parallel) pour mesurer cet état et la valeur H_1 = valeur du hachage $H(ID \parallel N_t)$ où ID qui est l'identifiant statique du tag, puis envoie " N_t et H_1 " au READER.

Le tag a besoin d'un générateur pseudo-aléatoire et un objet incorporé une fonction du hachage irréversible mais seulement stocke son identifiant.

✓ **READER**

READER reçu les valeurs “ Nt et H1 ” et l’envoie au SERVER.

✓ **SERVER**

SERVER vérifie H1 (si l’égalité alors l’authentification du TAG est succès) »,

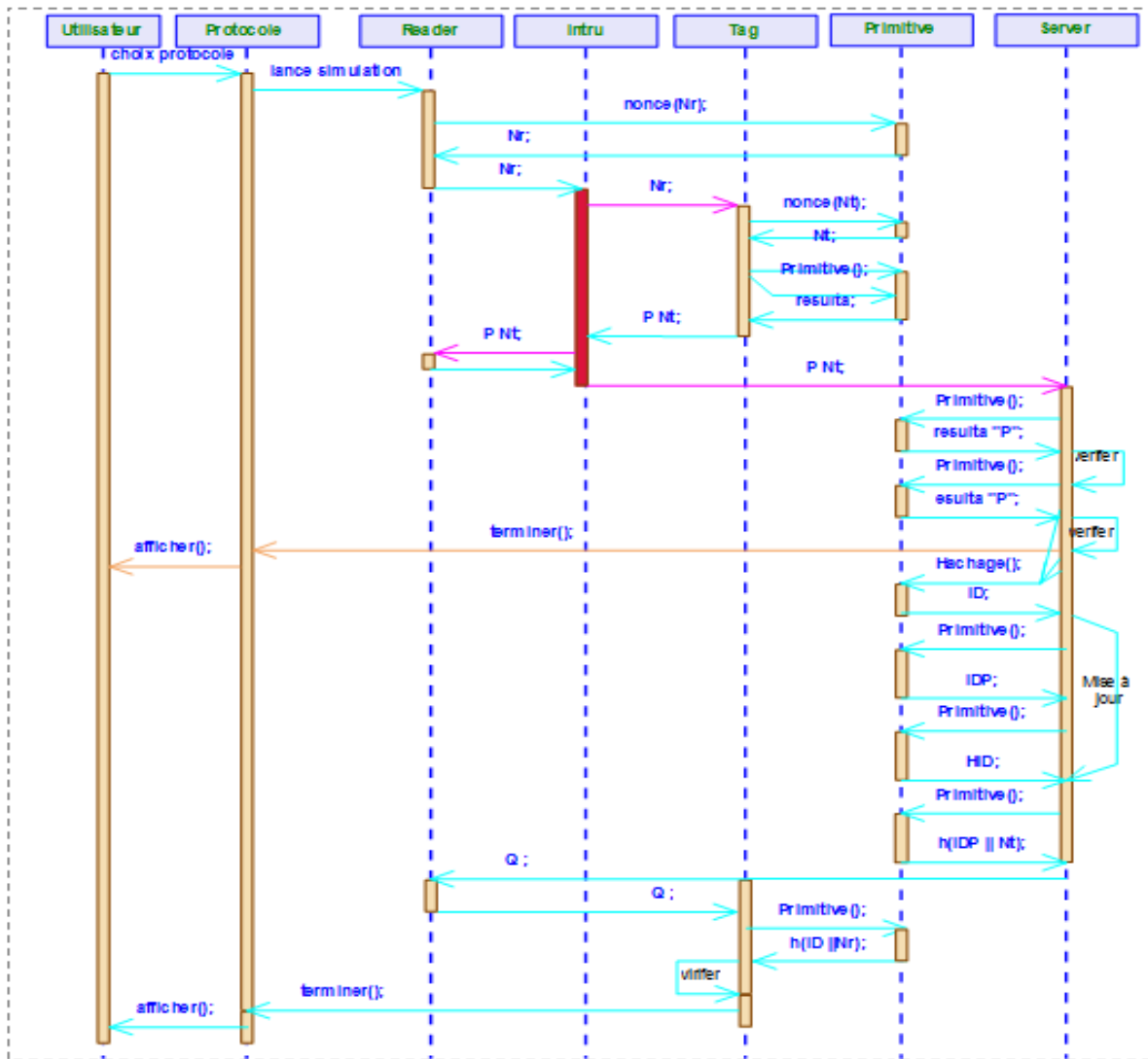
3.3.4. Protocole HMNB :

Figure N° III.6:Diagramme de séquence détaillé de protocole HMNB.

Description :✓ **READER**

READER produit d’une chaîne hexadécimale aléatoire Nr, pour présenter chaque Bit (générateur pseudo-aléatoire), et envoie ce nombre au tag,

✓ TAG

TAG reçu la valeur Nr par READER, les informations transmises par le tag à chaque fois qu'elle est interrogée se compose d'une valeur aléatoire Nt, la réponse du tag dépend de la valeur de S, si S=0 alors $P=h(ID)$ sinon utilise l'opération concaténation (||) pour mesurer cette état et la valeur $P = \text{valeur du hachage } H (ID || Nt || Nr)$ où ID est l'identifiant statique du tag, et avant l'envoi " Nt et P " à le READER la valeur S =1.

✓ READER

READER reçu les valeurs " Nt et P " et l'envoi au SERVER.

✓ SERVER

SERVER vérifie P « P, Nt », si l'égalité, l'authentification du TAG est succès.

Le SERVER a fait la mise à jour de l'ID, HID et ID', tel que :

si $P=h(ID)$ dans ce cas « $ID'=ID$; $ID=h(ID||nr)$; $HID=h(ID)$ »

si $P=h(ID || Nt || Nr)$ dans ce cas « $ID'=ID$; $ID=h(ID||nr)$; $HID=h(ID)$ »

si $P=h(ID' || Nt || Nr)$ dans ce cas « $ID=h(ID' || nr)$; $HID=h(ID)$ »

La valeur Q = valeur du hachage $h(IDP||Nt)$, et envoie à READER.

✓ READER

READER reçu la valeur de Q et l'envoi au TAG.

✓ TAG

TAG vérifie Q « Si la valeur Q = valeur du hachage $h(ID||Nt)$; alors l'authentification du READER est succès et le TAG fait la mise à jour tel que: $ID=h(ID||Nr)$; et S=0 ».

3.3.5. Protocole CRAP:

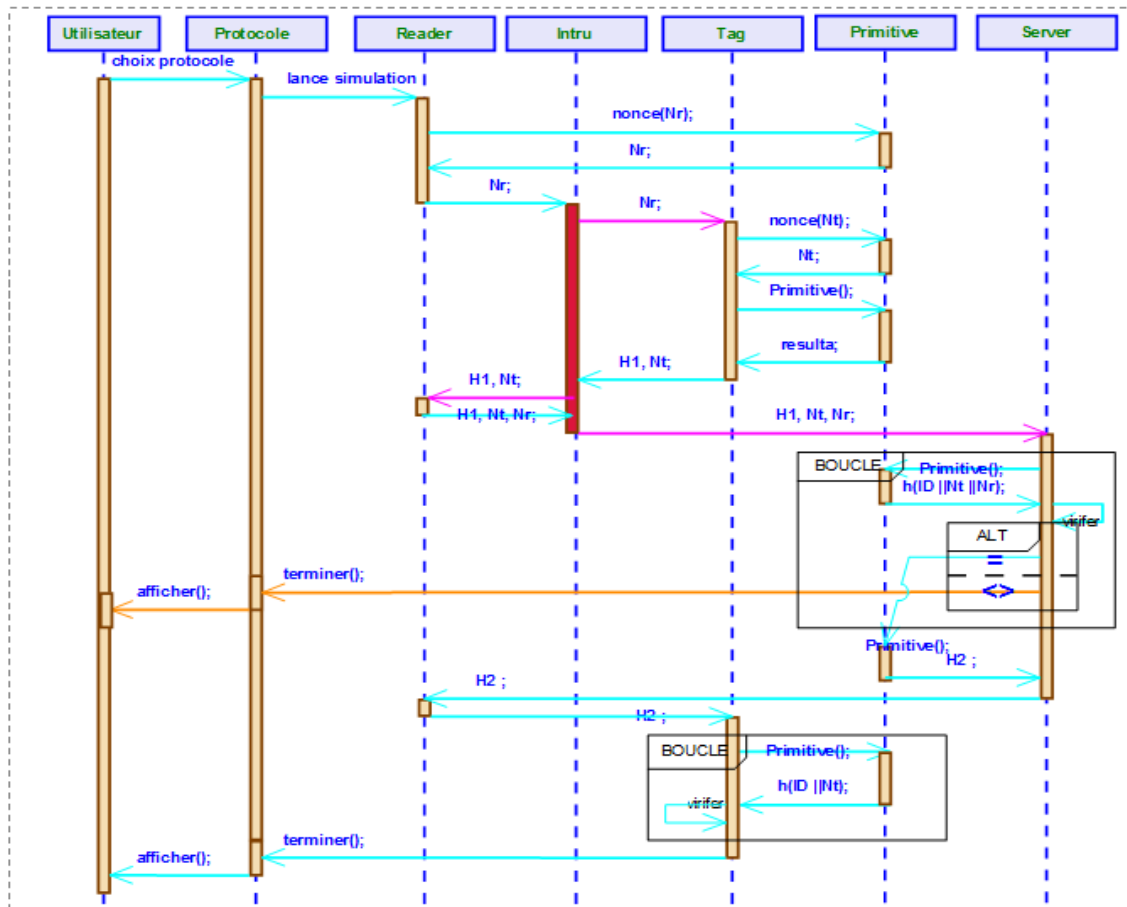


Figure N° III.7:Diagramme de séquence détaillé de protocole CRAP.

Description :

✓ **READER**

READER produit d'une chaîne hexadécimale aléatoire Nr, par le générateur pseudo-aléatoire, l'envoi au tag,

✓ TAG

Un TAG répond au READER en lui envoyant une fonction de hachage qui contient le nombre Nr reçu du READER et son propre nombre aléatoire Nt ainsi que l'identificateur ID, et utilise l'opération concaténation (||) pour mesurer cette état et la valeur H1 = valeur du hachage H (ID || Nt || Nr), puis envoie " Nt et H1 " au READER.

✓ **READER**

READER est reçois les valeurs “ Nt, H1 et Nr “ et l’envoie à SERVER.

✓ **SERVER**

SERVER vérifie H1 « Si la valeur de H1 = valeur du hachage H (ID || Nt || Nr) alors l'authentification du TAG est succès, et la valeur H2 = valeur du hachage H(ID || Nt), et envoie à READER.

READER

READER reçu la valeur de H2 et l'envoie au TAG.

✓ **TAG**

TAG vérifie H2 pour l'authentification du READER.

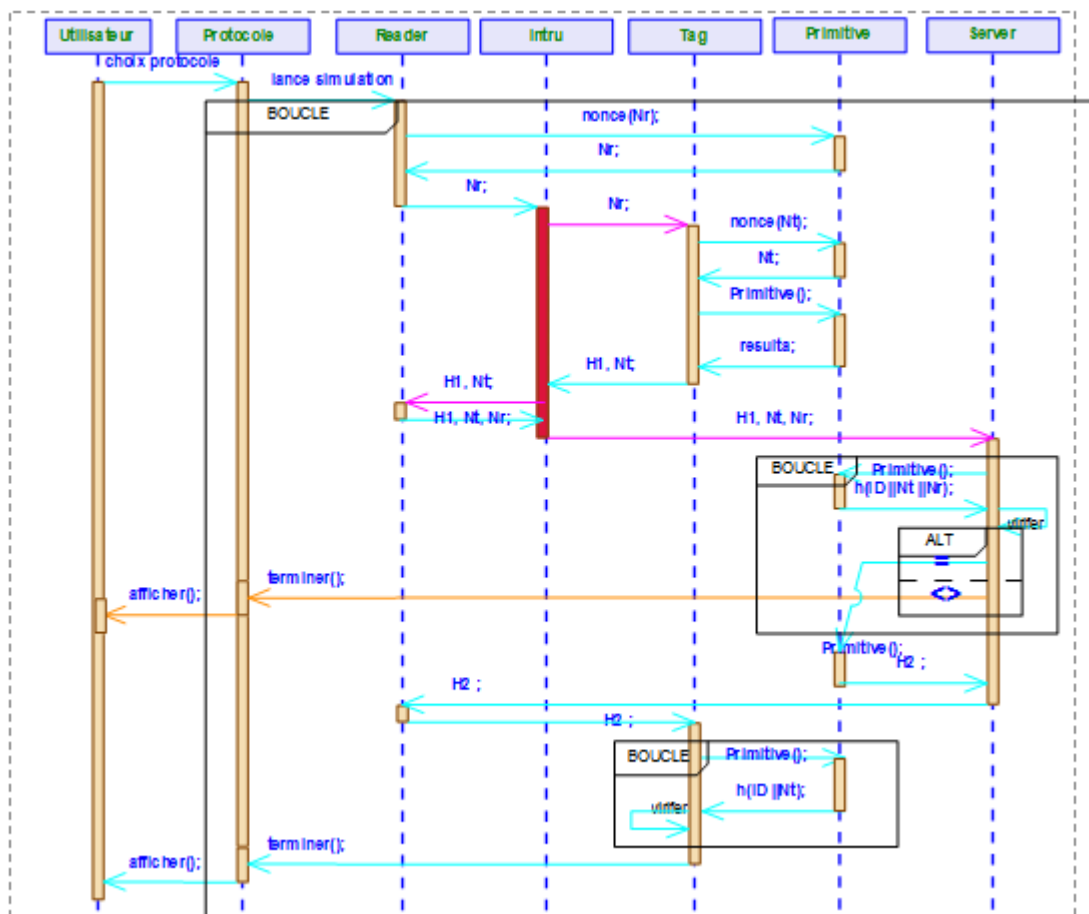


Figure N° III.8:Diagramme de séquence protocole CRAP (plusieurs TAGs arrivés).

La figure (III.8) représente le diagramme de séquence du protocole CRAP avec plusieurs tags arrivés. L'objectif de cette représentation est évalué la performance du protocole CRAP dans le cas d'arriver plusieurs tags avec un seul lecteur et un seul serveur.

3.3.6. Protocole PAP :

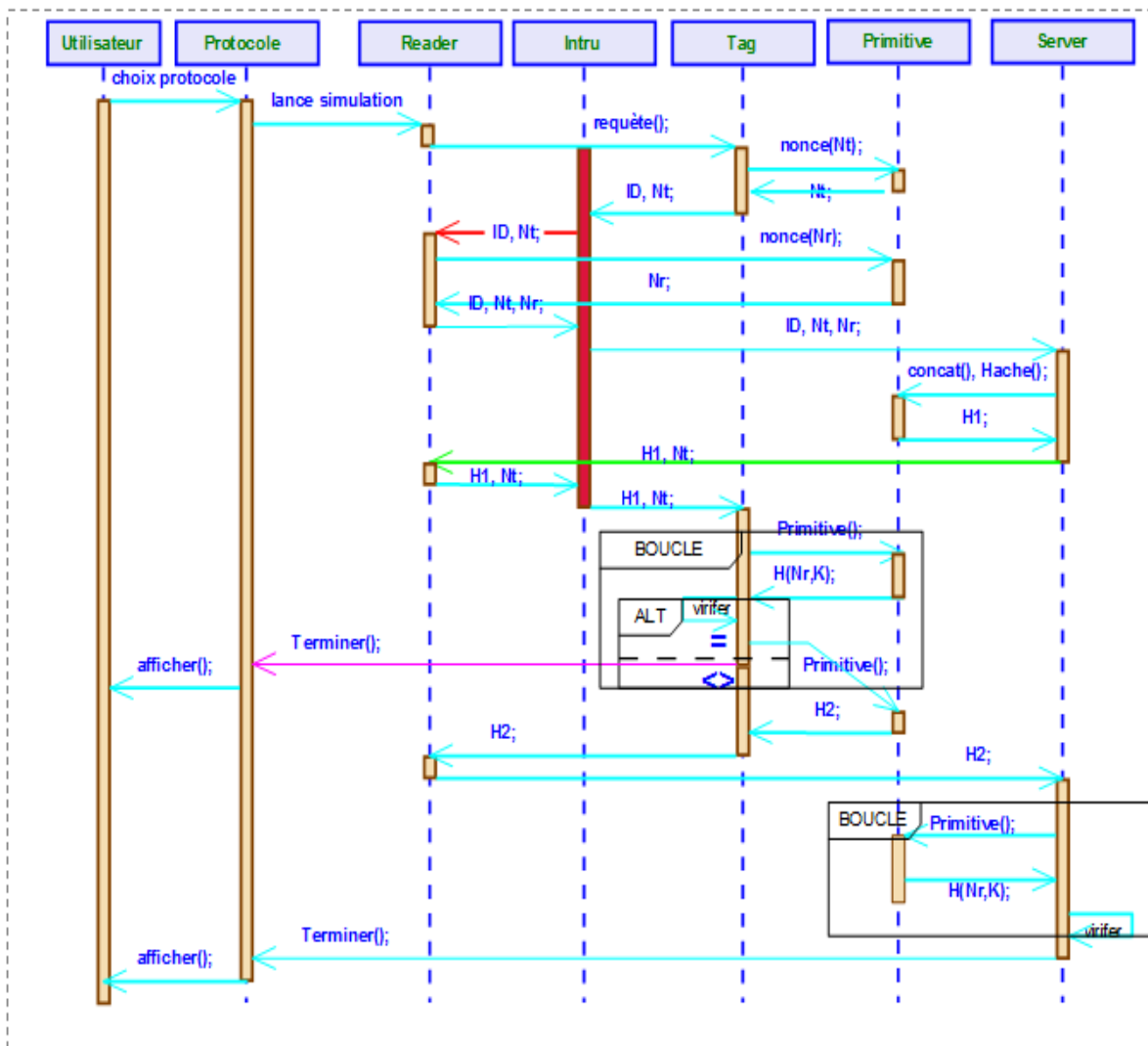


Figure N°III.9:Diagramme de séquence détaillé de protocole PAP.

Description :

✓ **READER**

READER envoie une requête au TAG.

✓ **TAG**

TAG envoie “ Nt et ID “ au READER, où ID qui est l'identifiant statique du tag.

✓ **READER**

READER reçu les valeurs de Nt et ID. READER produit d'une chaîne hexadécimale aléatoire Nr et l'envoie au SERVER .

✓ SERVER

SERVER reçu les valeurs “ Nt, ID et Nr “, et utilise l’opération concaténation (||) pour mesurer cette état et la valeur $H1 = \text{valeur du hachage } H(Nt || K)$, puis envoie “ H1 “ au READER.

✓ READER

READER reçu les valeurs “H1 “ et l’envoie avec Nr à TAG.

✓ TAG

TAG vérifie la valeur de H1 « Si l’égalité, l’authentification de READER est succès ; le tag calcule $H2 = \text{valeur du hachage } H(Nr || K)$, puis envoie l’envoie H2 au READER».

✓ READER

READER reçu la valeur de H2 et l’envoie au SERVER.

✓ SERVER

SERVER vérifie H2, si l’égalité alors l’authentification du TAG est succès.