Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *système d'information et connaissance (SIC)*

Thème

Etude et analyse de la sécurité des transmissions Bluetooth

Réalisé par :

MERIAH Mohammed Yassine

Présenté le 25 Juin 2024 devant le jury composé de MM.

- *Mr BRIKCI NIGASSA Amine …….* *(Président)*
- *Mr BENAMAR Abdelkrim……..* *(Encadreur)*
- *Mme REMACI Zeyneb Yasmina ……* *(Examinateur)*

Année universitaire : 2023-2024

REMERCIEMENTS

Tout d'abord, je remercie ***le dieu le tout puissant*** *pour son aide et pour la volonté qui m'a donnée pour finir mon travail.*

Il est particulièrement agréable, avant de présenter notre travail, d'exprimer nos gratitudes envers les personnes qui ont contribué de près ou de loin à l'élaboration de ce travail.

J'adresse mes remerciements à ***Monsieur le Professeur BENAMAR Abdelkrim****, qui était chargé de fournir des conseils et des orientations à chaque point de ce mémoire.*

Mes remerciements vont également à ***Monsieur Brikci Nigassa Amine****, pour avoir accepté de présider le jury de ma soutenance.*

Je tiens aussi à remercier ***Madame Remaci Zeyneb Yasmina*** *pour avoir accepté d'examiner et participer au jury de ma soutenance.*

Merci à tous

DÉDICACES

Je tiens à dédier cet humble travail :

A ma tendre mère Salima et mon très cher père Sidi Mohammed

A mes frères : Abderrahmène et Ahmed El Amine

A Tous ceux qui m'aiment et que j 'aime

MERIAH Mohammed Yassine

Tables des Matières

LISTE D'ABREVIATIONS

Liste d’abréviations

A

AES : Advanced Encryption Standard
ASIC : Application Specific Integrated Circuit
ANSI : Agence de Normalisation et de standardisation internationale

B
BD_ADDR: Bluetooth Device ADDRess
BLE: Bluetooth Low Energy

C
CRC: Cyclic redundancy check

D

DES : Data Encryption Standard
DSS : Digital Signature Standard
DH : Protocole Diffie-Hellman

I

IdO: Internet des Objets

L
L2CAP : Logical Link Control & Adaptation Protocol

M
MAC: Message Authentication Code
MD: Message Digest

E

ECC: Elliptic Curve Cryptography

H
HCI : Host Controller Interface

O
OBEX: OBject EXchange

P

PIN: Personal Identification Number
PKI: Public Key Infrastructure
PDU: Protocol Data Unit

R

RSA: Rivest.R, Shamir.A, Adleman.L
RIPEMD: Ripe Message Digest

S

SHA: Secure Hash Algorithm
SDP: Service Discovery Protocol

T

TDES: Triple Data Encryption Standard
TDD : Time Division Duplex

U

USB: Universal Serial Bus

LISTE DES FIGURES

LISTE DES TABLEAUX

Chapitre 1 :

Chapitre 2 :

Chapitre 3 :

Introduction Générale

Introduction Générale

- Contexte de Travail

Le Bluetooth est aujourd'hui omniprésent dans notre quotidien, facilitant la communication sans fil entre une multitude d'appareils. Cette technologie de communication sans fil à courte portée permet aux dispositifs électroniques de se connecter, de communiquer et de partager des données sans besoin de câbles. Conçu pour simplifier la connectivité et faciliter le transfert de données, le Bluetooth, comme toute technologie sans fil, reste vulnérable à diverses attaques, soulignant la nécessité de renforcer la protection des données contre les menaces potentielles, d'où la necessité de faire appel à des outils et algorithmes de cryptographie.

La cryptographie, moyen de dissimuler l'information, a une histoire ancienne, utilisée initialement à des fins militaires et diplomatiques. Aujourd'hui, son champ d'application s'est élargi, devenant crucial pour la sécurité des transactions bancaires et la transmission de fichiers et bases de données électroniques. Depuis ses débuts avec des systèmes simples comme les chiffrements de César et de Vigenère jusqu'aux systèmes modernes comme la machine de ENIGMA, la cryptographie a évolué pour inclure des systèmes à clés publiques et privées. Jusqu'à récemment, les systèmes à clés secrètes, tels que le Triple DES et l'AES, étaient prédominants, malgré le problème de communication des clés secrètes. Les systèmes asymétriques, ou à clé publique, ont émergé pour résoudre ce problème, avec le RSA, inventé en 1978, étant le plus courant. Ce système utilise un calcul d'exponentiation modulaire de très grands nombres pour le cryptage et le décryptage des messages.

- Problématique

Notre travail se situe dans le contexte de sécurisation des communications Bluetooth. Bien que le Bluetooth soit pratique et largement utilisé, ses vulnérabilités aux attaques nécessitent des solutions robustes pour protéger les données échangées. La problématique centrale de notre étude est de concevoir et de mettre en œuvre une méthode sécurisée de communication Bluetooth entre deux cartes Arduino, en utilisant des modules Bluetooth HC05 en mode maître-esclave. Les défis comprennent la garantie de la confidentialité, de l'intégrité et de l'authenticité des données transmises, tout en maintenant une efficacité acceptable en termes de temps d'exécution et de ressources matérielles limitées des cartes Arduino.

- Solution Proposée

Nous proposons une solution hybride combinant les algorithmes RSA et TDES pour sécuriser la communication Bluetooth entre deux cartes Arduino. Cette approche vise à tirer parti des avantages de chacun : la robustesse de l'algorithme RSA pour l'échange de clés et la rapidité du TDES pour le chiffrement des données. Notre mise en œuvre inclut les étapes suivantes :

1. Conception et Implémentation : Utilisation de deux modules Bluetooth HC05 en mode maître-esclave pour établir une connexion sécurisée entre les deux cartes Arduino Uno. La clé publique RSA est utilisée pour échanger une clé de session temporaire, qui est ensuite utilisée par le TDES pour chiffrer les données.

2. Mesure de Temps d'Exécution : Évaluation de l'efficacité de notre solution en mesurant le temps d'exécution des opérations de cryptage et de décryptage, ainsi que l'impact sur les performances des Arduino Uno.

3. Implémentation Logicielle : Développement et déploiement du logiciel sur les cartes Arduino Uno, intégrant les algorithmes de cryptographie et les protocoles de communication nécessaires.

4. Contribution : Développement d'un système de communication Bluetooth et intégration de divers algorithmes de sécurité tels algorithme TDES et l'algorithme RSA. Aussi, résoudre le problème d'authentification, en utilisant l'algorithme SHA256. Cela permet de garantir l'intégrité et l'authenticité des messages échangés.

- Structure du manuscrit

Notre manuscrit est structuré en trois chapitres:

Le premier chapitre introduit les notions de base de la sécurité, son historique, ses techniques et les standards de chiffrement.

Le second chapitre aborde la communication Bluetooth, expliquant son fonctionnement et détaillant le protocole Bluetooth.

Le troisième chapitre présente les implémentations possibles des algorithmes TDES et RSA sur les cartes Arduino, en utilisant les modules Bluetooth HC05 en mode maître-esclave, et introduit la signature SHA256 pour l'authentification.

Chapitre 1 : Notions de base de S*écurité*

1.1 Introduction :

La sécurité des systèmes informatiques et télécommunications vise à protéger l'accès et la manipulation des données et des ressources d'un système par des mécanismes d'authentification, d'autorisation et de contrôle d'accès, etc.

Néanmoins, avec l'ouverture et l'interconnexion des systèmes informatiques, des attaques exploitant les failles de ces systèmes et contournant leurs mécanismes de sécurité sont toujours possibles. Il n'est donc pas suffisant d'agir préventivement, c'est-à-dire de définir une politique de sécurité (en termes de confidentialité, d'intégrité et de disponibilité des données et ressources du système à protéger) et de mettre en œuvre des mécanismes implantant cette politique. Il faut aussi être capable de détecter toute tentative de violation de la politique de sécurité, c'est-à-dire toute intrusion, tels que :

- ✓ **La confidentialité :** La confidentialité consiste à rendre l'information inintelligible à d'autres personnes que les seuls acteurs de la transaction.

- ✓ **L'intégrité :** Vérifier l'intégrité des données consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle).

- ✓ **La disponibilité :** L'objectif de la disponibilité est de garantir l'accès à un service ou à des ressources.

- ✓ **La non-répudiation :** La non répudiation de l'information est la garantie qu'aucun des correspondants ne pourra nier la transaction.

- ✓ **L'identification et l'authentification :** L'authentification consiste à assurer l'identité d'un utilisateur, c'est-à-dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il croit être. Un contrôle d'accès peut permettre (par exemple par le moyen d'un mot de passe qui devra être crypté) l'accès à des ressources uniquement aux personnes autorisées.

- ✓ **Contrôle d'accès :** Mécanisme permettant de prévenir contre l'utilisation non appropriée ou non autorisée d'une ressource (service, système programme, données). La détermination des niveaux d'accès aux ressources dépend de l'évaluation de leurs niveaux de sensibilité. Cela est basé le plus souvent, sur une classification des données, des personnes et des flux informationnels.

1.2 L'internet des objets

1.2.1 Définition

Le concept de l'Internet des Objets (IdO) remonte aux années 1980, lorsque Ken Sakamura, un informaticien de l'université de Tokyo, a créé le premier système d'exploitation libre [1]. Ce succès a conduit Sakamura à imaginer un monde où des ordinateurs miniaturisés seraient intégrés dans divers objets, introduisant ainsi la notion de "computing everywhere", qu'il a popularisée à l'époque. Par la suite, il a parlé de "ubiquitous computing" et a démontré l'intérêt d'incorporer des puces électroniques dans des objets physiques [2]. En 1988, Mark Weiser, alors chef du laboratoire Ubicomp au Xerox PARC, a posé les bases théoriques de l'informatique ubiquitaire, énonçant des principes tels que l'assistance de l'ordinateur dans l'activité humaine, son caractère non intrusif, et son rôle comme prolongement de notre inconscient. Il a particulièrement souligné que l'informatique ubiquitaire repose sur une architecture distribuée avec un noyau central minimal et des périphériques largement autonomes [3].

Le terme "internet of things" n'a émergé que plus d'une décennie plus tard. En 1999, Kevin Ashton, gestionnaire de marque chez Procter & Gamble, a utilisé pour la première fois la notion d'IdO lors d'une présentation visant à améliorer l'efficacité de la chaîne d'approvisionnement. Ce n'est qu'en 2012 que l'Union internationale des télécommunications a fourni une définition formelle, décrivant l'IdO comme une "infrastructure mondiale pour la société de l'information, permettant de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution".

l'Internet des Objets constitue une révolution aux applications multiples. Ses domaines d'application sont vastes, englobant aussi bien les services à la personne (comme les cannes intelligentes pour malvoyants) que la sécurité routière (avec les voitures autonomes), ou encore l'économie d'énergie (comme les bâtiments intelligents).

1.2.2 Arrière-plan technique

Un objet connecté intègre généralement un ou plusieurs capteurs. Ceux-ci peuvent varier d'un simple bouton détectant une pression à des dispositifs beaucoup plus complexes capables de percevoir des variations environnementales telles que la luminosité, la température ou la pression. Lors de la fabrication d'un objet connecté, ces capteurs sont soudés sur une carte électronique équipée d'un processeur, qui traite

les données et les transfère vers le composant émetteur, introduisant ainsi la notion de connectivité.

Les besoins en énergie et en débit étant très diversifiés, il existe une multitude de technologies de connectivité pour l'Internet des Objets (IdO). Pour répondre aux différents cas d'usage, on considère généralement trois grandes catégories de réseaux, illustrées par la figure 1.1 selon divers critères.

La première catégorie englobe les réseaux locaux à courte portée, couramment utilisés aujourd'hui, tels que le WiFi et le Bluetooth. Ces technologies, ayant une portée inférieure à 100 mètres, ont été initialement conçues pour échanger des données multimédias comme de la musique ou des vidéos. Leur principal inconvénient est la nécessité d'une passerelle entre l'objet et Internet (par exemple, un routeur WiFi ou un smartphone équipé de Bluetooth).

Les objets étant accessibles à distance grâce à leur connectivité, ils peuvent être vulnérables aux attaques de personnes malintentionnées. Par exemple, dans une ville équipée de lampadaires connectés, une attaque contre ces dispositifs pourrait semer la peur et l'insécurité parmi la population. Pour répondre aux exigences de sécurité de l'IdO, il est donc nécessaire d'utiliser divers outils, notamment des algorithmes cryptographiques.

1.3 La cryptologie

La cryptologie, provenant du grec "kruptos" signifiant "caché", est l'étude du secret. Elle se divise en deux disciplines principales :

- La cryptographie, qui regroupe toutes les méthodes de protection de l'information. Son rôle principal est de garantir la confidentialité des données lors de leur transmission ou de leur stockage en utilisant des techniques de chiffrement. Elle vise également d'autres objectifs de sécurité, tels que l'intégrité et l'authentification.

- La cryptanalyse, qui consiste à analyser les messages chiffrés et à tenter de contourner les protections cryptographiques pour retrouver l'information originale sans disposer de la clé de déchiffrement. Cette compétence spécialisée est rare dans le monde professionnel. Dans ce cours, vous explorerez quelques-unes des attaques utilisées en cryptanalyse.

1.3.1 Historique

Les algorithmes de chiffrement sont essentiels pour sécuriser des informations confidentielles. Un exemple antique notable est le chiffre de César, un chiffrement par décalage où chaque lettre du texte clair est remplacée par une autre décalée dans

l'alphabet, suivant une clé définie par le décalage. Cependant, ce type de chiffrement est vulnérable à une attaque par force brute, où toutes les clés possibles sont testées.

Auguste Kerckhoffs introduit les principes fondamentaux de la cryptographie moderne en 1988 [4]. L'un de ces principes stipule que la sécurité d'un système de chiffrement doit reposer uniquement sur la clé, non sur la confidentialité de l'algorithme lui-même. L'avènement de l'informatique au XXe siècle transforme la cryptanalyse en permettant une automatisation, illustrée par Claude Shannon en 1949 qui établit les bases théoriques de la cryptographie moderne, incluant le concept de chiffrement parfait et l'importance de l'entropie linguistique [5].

Le Data Encryption Standard (DES) est standardisé en 1977, malgré des critiques concernant la taille relativement petite de sa clé effective (56 bits sur 64). À la fin des années 1990, des attaques distribuées montrent la vulnérabilité du DES en trouvant sa clé en moins de 24 heures grâce à des réseaux d'ordinateurs connectés [6-7] .

En réponse aux faiblesses du DES, l'Advanced Encryption Standard (AES) est proposé en 2001, offrant des clés de 128, 192 et 256 bits. L'AES est aujourd'hui largement considéré comme sécurisé et demeure l'algorithme de chiffrement symétrique prédominant à l'échelle mondiale. Contrairement aux chiffrements à flot, qui opèrent bit par bit, l'AES et le DES sont des chiffrements par blocs, manipulant des blocs de données en entrée et en sortie [8,9] .

Outre le chiffrement, la cryptographie inclut les fonctions de hachage qui produisent des empreintes fixes à partir de données de taille variable, assurant l'intégrité des données. Pour garantir l'authenticité des données, il est nécessaire d'utiliser des codes d'authentification de message (MAC) générés par des fonctions de hachage ou des chiffrements par bloc comme HMAC ou CBC-MAC [11,12].

La cryptographie symétrique nécessite un échange préalable de clés pour sécuriser la communication, une contrainte surmontée par la cryptographie asymétrique [13] , introduite par Diffie et Hellman en 1976. Cette méthode repose sur des paires de clés publiques et privées, permettant de chiffrer et déchiffrer des messages sans partager une clé de chiffrement commune.

L’algorithme RSA est publié comme le premier système de chiffrement asymétrique en 1978 [14] , basé sur la difficulté de factoriser de grands nombres premiers. Les systèmes utilisant des courbes elliptiques (ECC) sont ensuite développés, offrant une sécurité équivalente à des clés plus courtes qu'en RSA, grâce à la complexité accrue des calculs mathématiques nécessaires pour résoudre le problème du logarithme discret.

TABLEAU 1 – Comparatif des tailles de clefs (en bits) par niveau de sécurité [17]

Bits de sécurité	Taille de module RSA	Taille de clef ECC
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

1.3.2 Définition et terminologies

La cryptologie, étymologiquement la science du secret, englobe deux champs d'étude. Tout d'abord la *cryptographie*, qui à l'aide d'un secret, a pour but de garantir les propriétés de confidentialité (l'information n'est accessible qu'à ceux dont l'accès est autorisé), d'intégrité (l'information n'a pas été altérée) et d'authenticité (l'information est intègre atteste sa provenance). Ensuite la *cryptanalyse*, qui consiste à mettre en échec les algorithmes cryptographiques afin d'extraire ou d'altérer les informations protégées, sans avoir accès au secret.

Afin de garantir la confidentialité d'une information, l'expéditeur *chiffre* le *texte clair M* à l'aide d'une fonction E paramétrée par une *clef K*, et obtient ainsi un *texte chiffré C*. Le destinataire *déchiffre C* à l'aide d'une fonction *D* paramétrée par une clef *K '* afin d'obtenir *M*. L'action de retrouver le texte clair à partir du chiffré sans la connaissance de la clef est appelée *décryptage*. Les procédures de chiffrement et de déchiffrement peuvent être exprimées mathématiquement comme suit : E*K* (*M*) = *C et* D*K'*(*C*) = *M* .

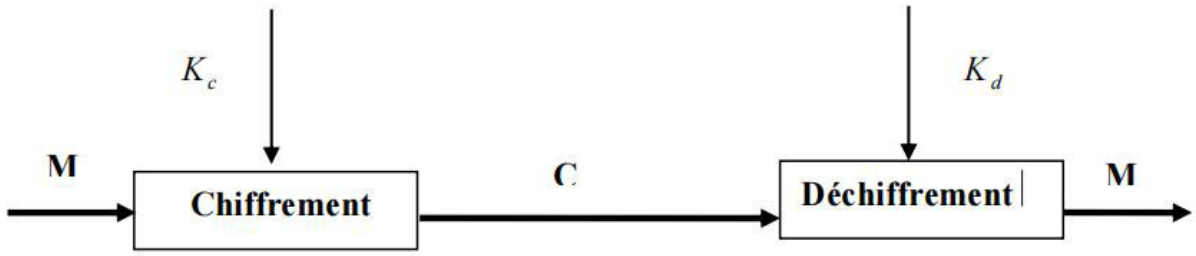

Figure 1.1 Système de cryptographie.

Dans le cas où la même clef est utilisée pour chiffrer et déchiffrer (*i.e.*, $K = K'$), on parle de chiffrement *symétrique*. Dans le cas contraire, on parle de chiffrement *asymétrique*.

1.4 La cryptographie moderne

Si le but de la cryptographie classique est d'élaborer des méthodes permettant de transmettre des données de manière confidentielle, la cryptographie moderne s'intéresse en fait plus généralement aux problèmes de sécurité des communications [18]. Pour cela, on utilise un certain nombre de mécanismes basés sur des algorithmes cryptographiques [19].

La sécurité des données chiffrées dépend des éléments suivants [20]:

La robustesse de l'algorithme cryptographique (difficile à casser).

La confidentialité de la clé.

Les techniques de cryptographie moderne se composent de grandes parties comme le montre la figure 1.2 [20-21] :

La cryptographie à clés secrètes ou cryptographie symétrique.

La cryptographie à clés publiques ou cryptographie asymétrique.

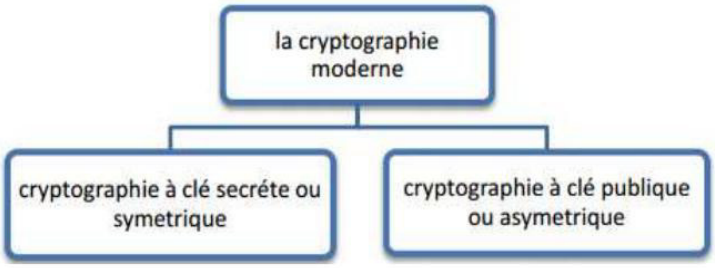

Figure 1.2: les méthodes de la cryptographie moderne [9].

Ils ont tous les deux leurs avantages et leurs inconvénients. La différence qui existe entre ces deux types se situe au niveau de la clé [9].

1.4.1 Cryptographie symétrique

Le cryptage à clé privée ou symétrique est basé sur une clé (ou algorithme) partagée entre les deux parties communicantes. Cette même clé sert à crypter et décrypter les messages. Lorsqu'on utilise la cryptographie à clé symétrique, nous devons considérer certains principes qui constituent les critères de base pour implémenter une sécurité fiable et efficace. Les algorithmes de chiffrement les plus connus sont : Kerberos, DES (Data Encryption Standard).

La difficulté engendrée par la génération, le stockage et la transmission des clés (on appelle l'ensemble de ces trois processus le management des clés : Key management) limite le système des clés privées surtout sur Internet (Figure 1.3).

Le principal inconvénient de la cryptographie à clé symétrique est qu'il faut partager la clé secrète d'une manière ou d'une autre.

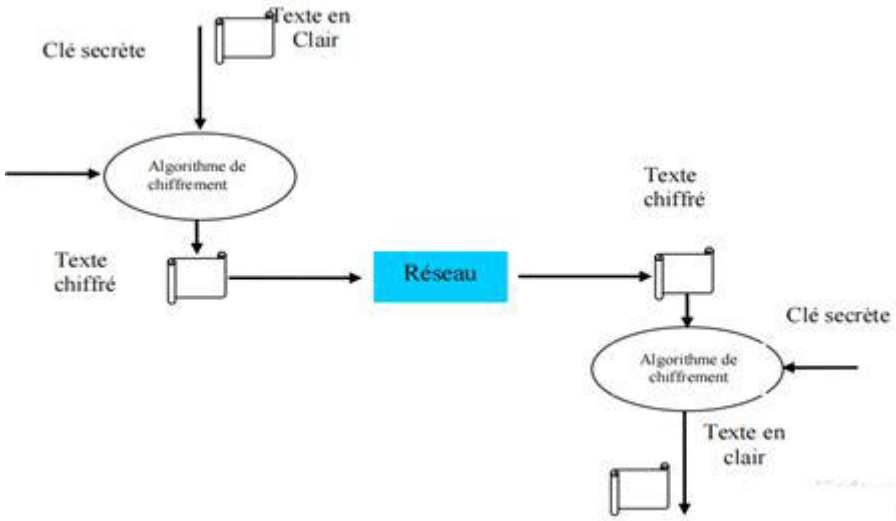

Figure 1.3 Algorithme de chiffrement symétrique.

1.4.2 Cryptographie asymétrique

Ce système de cryptage utilise deux clés différentes pour chaque utilisateur : L'une est privée et n'est connue que de l'utilisateur ; l'autre est publique et donc accessible à tout le monde.

Les clés, publique et privée, sont mathématiquement liées par l'algorithme de décryptage de telle manière qu'un message crypté avec une clé publique ne puisse être décrypté qu'avec la clé privée correspondante. Une clé est donc utilisée pour le cryptage et l'autre pour le décryptage (Figure 1.4).

La messagerie électronique du destinataire vérifiera si la clé privée correspond à la clé publique, puis invitera l'utilisateur à saisir la phrase d'authentification pour déchiffrer le message.

Ce cryptage présente l'avantage de permettre le placement des signatures numériques dans le message et ainsi permettre l'authentification de l'émetteur. Le principal avantage du cryptage à clé publique est de résoudre le problème de l'envoi de clé privée sur un réseau non sécurisé. Bien que plus lent que la plupart des cryptages à clé privée il reste préférable pour les raisons suivantes :

- Plus évolutif pour les systèmes possédant des millions d'utilisateurs.

- Authentification plus flexible.

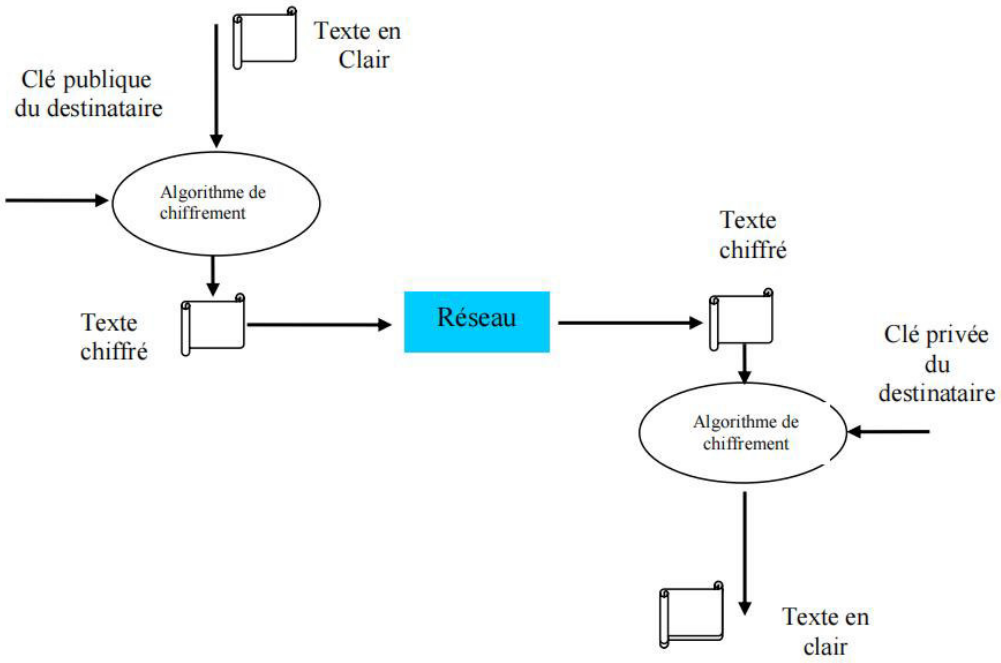

Figure 1.4 Algorithme de chiffrement asymétrique.

- Quelques bonnes pratiques pour le chiffrement asymétrique :

1. Utilisez des clés de 2048 bits ou plus
2. Stockez votre clé privée localement, afin de ne pas l'oublier
3. Ne partagez pas votre clé privée avec qui que ce soit.

La création de clés fortes est la base du chiffrement asymétrique. Une bonne pratique de chiffrement consisterait à utiliser plusieurs méthodes de chiffrement au lieu d'une seule. Tout le monde ne sait pas utiliser le chiffrement à clé publique, il peut donc arriver que vous deviez utiliser différentes méthodes de chiffrement. Mailfence utilise un chiffrement asymétrique basé sur l'algorithme RSA pour les clés basées sur OpenPGP. L'algorithme ECC (courbe 25519) est également pris en charge pour les clés basées sur OpenPGP.

- Avantages et inconvénients du chiffrement asymétrique :

Le chiffrement asymétrique présente également des avantages et des inconvénients :

1. Les avantages :

- Les données ne peuvent être déchiffrées qu'à l'aide de la clé privée détenue par le propriétaire.
- En cas de perte ou de vol de la clé publique, les données ne sont pas compromises.
- Permet l'authentification et la non-répudiation en plus de la confidentialité.

2. Inconvénients :

- Il est plus lent que le chiffrement symétrique.
- Utilise plus de ressources.
- En cas de perte de la clé privée, il n'existe aucun moyen de la récupérer.

1.5 La différence entre le chiffrement symétrique et asymétrique

Le chiffrement symétrique utilise une clé privée pour chiffrer et déchiffrer un message chiffré. Le chiffrement asymétrique utilise la clé publique du destinataire pour chiffrer le message. Ensuite, si le destinataire veut déchiffrer le message, il devra utiliser sa clé privée pour le déchiffrer. Si les clés correspondent, le message est déchiffré. Le tableau 1.2 présente une comparaison des avantages et inconvénients des deux méthodes de chiffrement.

TABLEAU 2: Avantages / Inconvénients

Algorithme	Avantages	Inconvénients
Symétrique	• Facilité d'intégration • Plus performant	• Moins sécurisé (Par le fait que la clé secrète est facilement transmissible)
Asymétrique	• Clé privée connue que d'un seul acteur • (RSA demande une clé minimale de 1024/2048 bits) • (Les courbes elliptiquesne nécessitent qu'une clé de 128 bits) • Plus sécurisé	• Moins performant (Coûteux en ressource,temps de calcul plus élevé) • Complexité à gérer (Utilisation d'une PKI)
Hybride	• Plus performant • Plus sécurisé	• Échange de deux informations (clé symétrique chiffré et message chiffré)

1.6 La cryptanalyse

La cryptanalyse est la science qui a pour but de mettre en péril un algorithme cryptographique sans avoir accès au secret. La méthode la plus naïve qui soit, appelée *attaque par force brute*, consiste à tester chaque valeur possible de la clef. Pour tout

algorithme respectant les principes de Kerckhoffs, cette attaque est impossible à mener en pratique étant donné que l'ensemble des valeurs possibles étant trop important. Le tableau 1.2 fournit quelques ordre de grandeurs significatifs, qui illustrent la complexité de considérer les 2^n hypothèses de clef pour différentes valeurs de *n*.

TABLEAU 3 – Ordres de grandeur significatifs

Nombre de grains de sable dans le Sahara	$10^{23} \approx 2^{77}$
Nombres d'atomes constituant la Terre	$10^{51} \approx 2^{170}$
Nombres d'atomes constituant l'Univers	$10^{77} \approx 2^{265}$

La cryptanalyse ne désigne pas nécessairement un moyen pratique de casser un algorithme cryptographique, mais toute méthode qui exploite une faiblesse dans la conception ce dernier afin d'offrir une complexité inférieure à l'attaque par force brute. Par exemple, la meilleure attaque publiée à ce jour à l'encontre du chiffrement AES-128 requiert 2126 opérations [23].

1.7 La clé :

On appelle clé une valeur utilisée dans un algorithme de cryptographie, afin de chiffrer un texte. Il s´agit en fait d´un nombre complexe dont la taille se mesure en bits. On peut imaginer que la valeur numérique correspondant à 1024 bits est absolument gigantesque. Plus la clé est grande, plus elle contribue à élever la sécurité à la solution.Toutefois, c´est la combinaison d´algorithmes complexes et de clés importantes qui seront la garantie d´une solution bien sécurisée. Les clés doivent être stockées de manière sécurisée et de manière à ce que seul leur propriétaire soit en mesure de les atteindre et de les utiliser.

1.7.1 Gestion des clés :

La mise en œuvre d'un système de cryptographie doit passer par la diffusion des clés aux correspondants concernés. Dans le cas de la cryptographie à clé secrète, ceci se traduit par le choix d'une clé secrète commune et son envoi aux deux correspondants doit se faire en utilisant un canal sûr (envoi postal, téléphonie, porteur). Dans le cas de la cryptographie à clé publique, les correspondants doivent s'échanger leurs clés publiques, l'entité qui s'occupe de la conservation et de la divulgation des clés publiques est appelée gestionnaire des clés. Le gestionnaire peut être, par exemple, un serveur ou les clés publiques peuvent être consultées. La distribution des clés constitue ainsi un problème majeur en matière de cryptographie.

1.8 Signature numérique

L'un des principaux avantages de la cryptographie de clé publique est qu'elle offre une méthode d'utilisation des signatures numériques. Celles-ci permettent au destinataire de vérifier leur authenticité, leur origine, mais également de s'assurer qu'elles sont intactes. Ainsi, les signatures numériques de clé fournissent également une fonctionnalité de non répudiation, afin d'éviter que l'expéditeur ne prétende qu'il n'a pas envoyé les informations. Ces fonctions jouent un rôle tout aussi important pour la cryptographie que pour la confidentialité.

La signature numérique d'un document se fait en deux temps. Tout d'abord, il est appliqué une fonction mathématique dite ''de hachage'', qui va à partir d'un texte générer une suite, de nombres binaires, de taille fixe, et ce quelque soit la taille du texte de départ. Cela constitue l'empreinte du texte. Si un caractère du texte change (''a'' devient ''A'') l'empreinte ne sera pas la même. Ensuite l'empreinte est chiffrée avec la clé privée de l'utilisateur, puis attachée au texte original et enfin le message est envoyé.

1.9 La fonction de hachage

Une fonction de hachage est aussi appelée fonction de hachage à sens unique ou "one-way hash function" en anglais [23]. Ce type de fonction est très utilisé en cryptographie, principalement dans le but de réduire la taille des données à traiter par l'algorithme de chiffrement. En effet, la caractéristique principale d'une fonction de hachage est de produire un haché des données, c'est-à-dire un condensé de ces données. Ce condensé est de taille fixe, dont la valeur diffère suivant la fonction utilisée. Fonctions de hachage usuelles :

– **MD4 et MD5** (Message Digest) furent développées par Ron Rivest. MD5 produit des hachés de 128 bits en travaillant les données originales par blocs de 512 bits.

– SHA-1 (Secure Hash Algorithm 1), comme MD5, est basé sur MD4. Il fonctionne également à partir de blocs de 512 bits de données et produit par contre des condensés de 160 bits en sortie. Il nécessite donc plus de ressources que MD5.

– **SHA-2** (Secure Hash Algorithm 2) a été publié récemment et est destiné à remplacer SHA-1. Les différences principales résident dans les tailles de hachés possibles : 256, 384 ou 512 bits. Il sera bientôt la nouvelle référence en termes de fonction de hachage.

– **RIPEMD-160** (Ripe Message Digest) est la dernière version de l'algorithme RIPEMD. La version précédente produisait des condensés de 128 bits mais présentait des failles de sécurité importantes. La version actuelle reste pour l'instant sûre; elle produit comme son nom l'indique des condensés de 160 bits.

Un dernier point la concernant est sa relative gourmandise en termes de ressources et en comparaison avec SHA-1 qui est son principal concurrent.

– **Tiger** : Tiger est une fonction de hachage cryptographique conçue par Ross Anderson et Eli Biham en 1996. Tiger fournit une empreinte sur 192 bits mais des versions sur 128 et 160 bits existent aussi. Ces versions raccourcies prennent simplement les premiers bits de la signature de 192 bits.

1.10 Certification des clés

Afin d'éviter toute tentative de déguisement, il faut mettre en place un dispositif de certification permettant de vérifier que la clé publique est bien associée au détenteur légitime et d'authentifier le gestionnaire de clés. Un certificat est un document émis par une institution reconnue et attestant de la propriété d'une clé publique par un correspondant. L'infrastructure qui assure la gestion et la distribution des clés la plus connue est **PKI** (Public Key Infrastructure) (figure 1.5).

Les certificats doivent être infalsifiables, pouvoir être obtenus en toute sûreté et crées de telle façon que personne d'autre que leur destinataire légitime ne puisse les utiliser. Un certificat comprend en général les éléments suivants : la clé publique, le nom du propriétaire, la date d'expiration de la clé, le nom du responsable du certificat et le numéro de série de celle-ci.

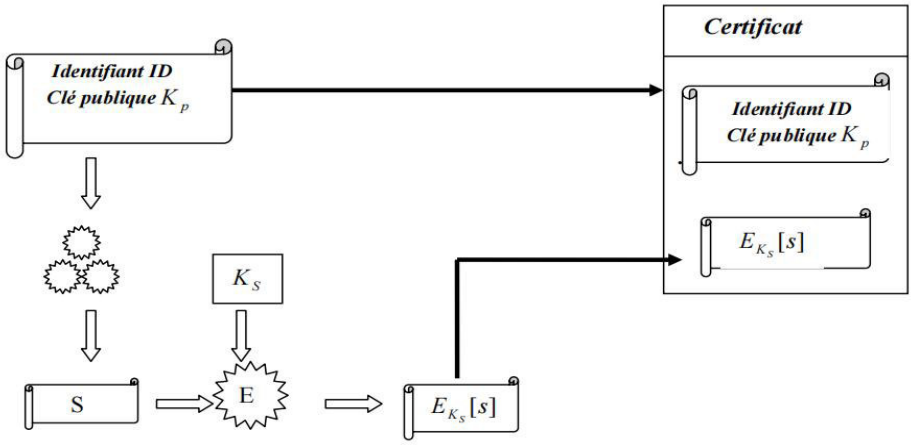

Figure 1.5 Procédé de certification de clé.

1.11 Conclusion :

Dans ce chapitre, nous avons donné un aperçu sur la sécurité Bluetooth. Nous avons débuté par un historique et définition de la cryptographie, ainsi que la classification des crypto-systèmes de chiffrement et de déchiffrement. Nous avons aussi abordé des exemples d'algorithmes de cryptage symétrique et asymétrique citons quelques avantages et inconvénients de chaque cryptage. Dans le chapitre suivant, nous allons donner un aperçu sur la communication Bluetooth, et présenté le protocole Bluetooth, ainsi que les différents piles composant ce protocole.

Chapitre 2 : Communication Bluetooth

2.1 Introduction :

Bluetooth est un standard de communication dont le but est d'établir des connexions rapidement et sans fil. Différentes normes composent ce standard, afin de donner aux constructeurs de périphériques les informations nécessaires au bon fonctionnement de Bluetooth. Certaines de ces normes définissent de quelle manière la sécurité doit être implémentée. Cela est vrai tant pour le périphérique en lui-même que pour les protocoles de communication à utiliser. Elles indiquent également comment les clefs permettant l'authentification et le maintien des connexions doivent être générées et stockées.

Dans ce chapitre, nous allons donner quelques définitions et présenter le protocole Bluetooth. Nous allons aussi abordé la sécurité dans le Bluetooth, ainsi que les mécanismes de sécurité au niveau liaison des données.

2.2 Definition de Bluetooth

Bluetooth est une norme de télécommunications permettant l'échange bidirectionnel de données à courte distance en utilisant des ondes radio sur la bande de fréquence de 2,4 GHz. Son but est de simplifier les connexions entre les appareils électroniques à proximité en supprimant des liaisons filaires. Elle peut remplacer par exemple les câbles entre ordinateurs, tablettes, haut-parleurs, téléphones mobiles entre eux ou avec des imprimantes, scanneurs, claviers, etc....

Les industriels ont rapidement vu le potentiel de cette technologie, mais des évolutions techniques étant nécessaires pour pouvoir réellement utiliser le Bluetooth en milieu industriel, il aura fallu attendre la version 4.0 pour permettre un déploiement massif. C'est ainsi qu'est apparu le Bluetooth Low Energy (BLE).

2.2.1 Concept de base : le Piconet

Les liaisons physiques se font sur le principe Maître/Esclaves [24], fonctionnant selon le type point à multipoint. Un maître pouvant contrôler jusqu'à sept esclaves dans sa zone. Ceux-ci forment alors un petit réseau appelé Piconet (Figure 2.1).

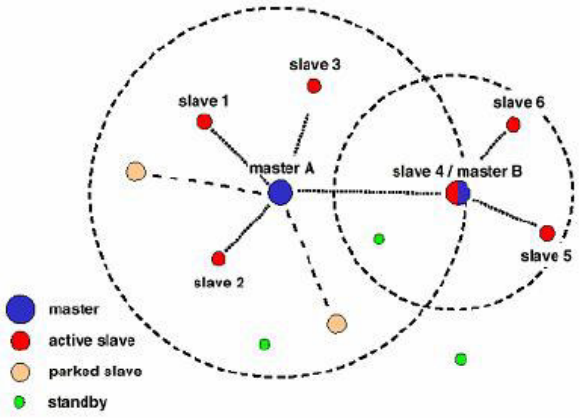

Figure 2.1 : Maître/Esclaves dans un Piconet

Le maître est tout simplement le premier appareil connecté et c'est lui qui fixe l'horloge, la séquence de saut de fréquences et le code d'accès de la liaison. Tous les modules Bluetooth d'un même Piconet utilisent la même séquence de saut de fréquence et sont synchronisés sur l'horloge du maître. Quoiqu'il arrive, le rôle de la machine (maître ou esclave) est invisible pour l'utilisateur.

L'imbrication de plusieurs piconets forme alors ce que l'on appelle un Scatternet. Le schéma d'émission au sein d'un même Piconet est basé sur le principe de duplexage temporel (TDD : Time Division Duplex). C'est d'abord le maître qui envoie un paquet à une fréquence f(k), puis l'esclave auquel ce paquet est destiné a seul le droit d'y répondre pendant l'intervalle de temps suivant l'arrivée du paquet maître. La réponse de l'esclave se fait alors sur le canal de fréquence f(k+1).

2.3 Architecture protocolaire de Bluetooth

L'architecture protocolaire de Bluetooth est essentiellement une amalgamation des protocoles spécifiques Bluetooth et d'autres protocoles tels que WAP, WAE, TCP/UDP/IP, PPA,vCard, vCal et IrMC. Elle soutient également des protocoles de remplacement de câble tels que RFCOMM et des protocoles d'adaptation de téléphonie tels que AT commands. Ces protocoles ont été adoptés pour convenir à des applications Bluetooth mais ils ne sont pas spécifiques à Bluetooth.

Un avantage de cette structure amalgamée est qu'elle permet au contrôle de sécurité spécifique aux applications d'être mis en place de façon transparente au contrôle de sécurité des couches inférieures auxquelles Bluetooth fonctionne (Bluetooth fonctionne à la couche liaison de données). La pile protocolaire de Bluetooth est montrée dans le diagramme montré ci-dessous. Dans la partie propre Bluetooth, on parle des 3 composants : la partie radio (Bluetooth Radio), la bande de base (BaseBand) et le directeur de lien (Link Manager)

2.4 Bluetooth Radio

Bluetooth emploie la gamme radio de 2,45 GHz et une largeur de bande théorique maximale de 1 Mb/s, qui est légèrement ralenti par les effets de correction d'erreurs dus à la méthode FEC qui est une méthode efficace pour améliorer le taux d'erreurs sur les bits sans sacrifier trop de largeur de bande.

Les spécifications de Bluetooth utilisent la méthode thode du saut de fréquence avec la modulation GFSK - ce système de sélection et de saut entre les fréquences pendant les transmissions s'adapte pour n'importe quel environnement bruité entourant l'emplacement du transfert.

2.5 Le protocole Bluetooth

La technologie Bluetooth, comme tout réseau, peut être conceptualisée en termes de couches, bien que son modèle diffère de celui de l'OSI. Elle est plutôt définie par des piles de protocoles (Figure 2.2).

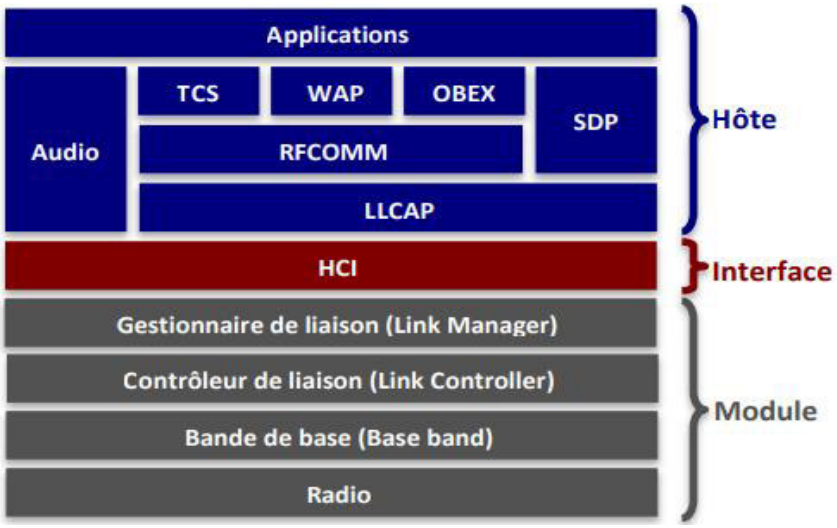

Figure 2.2: Les piles de protocoles Bluetooth

2.5.1 La couche "Radio"

La gestion de la couche "Radio" s'effectue au niveau matériel et englobe l'émission et la réception des ondes radio. Cette couche définit des paramètres tels que la bande de fréquence, la disposition des canaux, les spécifications du transmetteur et du récepteur, ainsi que la modulation, entre autres.

2.5.2 La couche "Bande de base" (Base Band)

C'est la couche qui permet le raccordement physique radio entre les dispositifs Bluetooth. Cette partie de l'architecture Bluetooth emploie une combinaison des technologies de commutation de circuits et de paquets:
- Un canal asynchrone de données et jusqu'à trois canaux synchrones simultanés de voix.
- Un canal qui transfère des données asynchrones et la voix synchrone employant simultanément des combinaisons des technologies de commutation.

La couche "Bande de base" établit les adresses matérielles des périphériques, désignées sous le nom de BD_ADDR (Bluetooth Device ADDRess), codées sur 48 bits. Elle gère également les divers types de connexions entre les appareils, qu'elles

soient synchrones ou asynchrones.

2.5.3 Le "contrôleur de liaisons" (Link Controller)

Le "contrôleur de liaisons" prend en charge la configuration et le contrôle de la liaison physique entre deux appareils.

2.5.4 Le "gestionnaire de liaisons" (Link Manager)

Le "gestionnaire de liaisons" gère les connexions entre les périphériques maîtres et esclaves, facilitant la mise en place des mécanismes de sécurité tels que l'authentification, le pairage, la création et la modification des clés, ainsi que le cryptage.

2.5.5 L'interface de contrôle de l'hôte (Host Controller Interface ou HCI)

HCI (Host Controller Interface) est employé pour fournir une interface de commande aux contrôleurs BaseBand, Link Manager et à d'autres contrôleurs de matériel.

Cette interface fournit une méthode standardisée pour accéder aux couches matérielles, permettant ainsi un développement indépendant du matériel et du logiciel. Elle supporte différents protocoles de transport tels que l'USB (Universal Serial Bus), les cartes PC, RS-232 ou UART.

2.5.6 La couche L2CAP

La couche L2CAP (Logical Link Control & Adaptation Protocol) permet l'utilisation simultanée de divers protocoles de niveaux supérieurs. Elle identifie chaque paquet envoyé pour que l'appareil distant puisse le transmettre au bon protocole une fois reçu. De plus, elle gère la segmentation et le réassemblage des paquets de protocoles de niveaux supérieurs en paquets de liaison de 64 Ko.

2.5.7 Autres protocoles

- ✓ **Le protocole RFCOMM**, basé sur les spécifications RS-232, émule les liaisons séries et peut être utilisé, par exemple, pour transmettre une connexion IP via Bluetooth.

- ✓ **Le protocole SDP** (Service Discovery Protocol) permet à un appareil Bluetooth de rechercher d'autres appareils et d'identifier les services disponibles.

- ✓ **Le protocole OBEX** (OBject EXchange) facilite le transfert de données en

utilisant le Protocole d'échange de fichiers IrDA.

- ✓ **TCS BINARY et AT Commands,** ce sont des protocoles de commande de téléphonie qui permettront à des services tels que les modems et le fax de fonctionner au-dessus de Bluetooth. RFCOMM, TCS BIN et AT commands ensembles et d'autres adoptés tels que TCP/UDP/IP, PPA et WAE/WAP forment les protocoles orientés application qui fonctionnent au dessus des protocoles Bluetooth spécifiques.

2.5.8 Format des paquets

Tous les paquets possèdent un header de 8 bit (Tableau 1.3), suivi par une adresse de 32 bits qui correspond a un identifiant unique pour une connexion donnée, enfin une suite variable de bits avec un PDU (Protocol Data Unit) de 2 a 39 octets4 (en BLE 4.0 et 4.1) contenant la charge utile du message suivi d'un CRC (Cyclic redundancy check) de 24 bits.

TABLEAU 1: Ordres de grandeur significatifs

Champ	Header	Access Address	Protocol Data Unit	CRC
Taille	8 bits	32 bits	2–39 octets	24 bits

2.6 La sécurité dans le Bluetooth

- Pour sécuriser les transmissions au niveau de la couche physique, Bluetooth utilise la méthode du saut de fréquence [25] , autour de la bande radio 1600 fois par seconde. Ceci améliore la clarté et réduit également ce qu'on appelle "écoute clandestine occasionnelle", en permettant seulement aux dispositifs synchronisés de pouvoir communiquer.

- La spécification Bluetooth inclut des mécanismes de sécurité au niveau liaison des données. Elle soutient l'authentification (unidirectionnelle ou mutuelle) et le chiffrage. Ces mécanismes sont basés sur une clé secrète de lien partagée par les deux dispositifs en communication. Pour générer cette clé un procédé appelé "pairing procedure" est employé quand les deux dispositifs se communiquent pour la première fois.

2.6.1 Les paramètres de base pour la sécurité

La sécurité du protocole Bluetooth (niveau liaison de données) est basée sur l'exploitation des trois paramètres suivants:

- ✓ **Un nombre aléatoire RAND :** permettant de simuler le hasard sur 128 bits. Il change fréquemment et il est produit par le dispositif Bluetooth.

- ✓ **Une adresse** dépendante du dispositif physique BD-ADDR (Bluetooth Device ADDResse) : Chaque carte Bluetooth se voit assigner une adressepermanente et unique de 48 bits lors de sa construction. Cette adresse permet aux autres utilisateurs d'avoir de la confiance en la personne à l'autre extrémité de la communication.

- ✓ **Un code personnel d'identification PIN :** C'est un code personnel qui est attribué à l'utilisateur. Ce code PIN peut être stocké sur 1 à 16 octets. Le PIN peut être stocké dans la mémoire non-volatile du dispositif. Ces paramètres permettent de créer des clés pour authentifier et chiffrer les transferts de données afin de les sécuriser.

2.6.2 Les modes de sécurité

Les appareils Bluetooth sont implémentés selon le GAP (Generic Access Profile) correspondant à un ensemble d'exigences obligatoires que doivent respecter les appareils Bluetooth pour qu'ils puissent communiquer entre eux. Plusieurs niveaux de sécurité sont définis dans le GAP [26] :

- ✓ **Mode 1 :**

1. Ce mode est non sécurisé pour toute opération
2. Aucune procédure d'initialisation sécurisée
3. Les appareils qui utilisent ce mode ne peuvent communiquer qu'avec les appareils de ce même mode.

- ✓ **Mode 2 :**

1. Fournit un niveau de sécurité au niveau application après établissement d'une liaison avec un autre appareil.
2. Il n'y a aucune procédure sécurisée avant l'établissement d'un canal de communication.
3. Niveau de sécurité implémenté au niveau du protocole minimal d'échange de donnée L2CAP.

- ✓ **Mode 3 :**

1. Fournit un niveau de sécurité avant l'établissement du canal de communication.
2. Chiffrement sécurisé au niveau de la liaison avec un autre équipement Bluetooth : LMP protocol.

Si un service ou un service connexe effectue une demande avec un mode de sécurité différent, le mode de sécurité le plus élevé est forcé afin de traiter la demande

2.6.3 Les niveaux de sécurité des dispositifs et des services

Bluetooth définit des niveaux de sécurité pour les dispositifs et les services [27] .

- Pour des dispositifs il y a deux niveaux possibles de sécurité. Un dispositif distant peut être:
1- **Un dispositif fiable :** Il aurait accès à tous les services pour lesquels la relation de confiance a été placée.
2- **Un dispositif non fiable :** Il aurait un accès limité aux services.

- Pour les services, trois niveaux de sécurité ont été définis.
1- **Les services qui exigent l'autorisation et l'authentification :** On accorde seulement l'accès automatique aux dispositifs de confiance. D'autres dispositifs ont besoin d'une autorisation manuelle.
2- **Les services qui exigent l'authentification seulement :** L'autorisation n'est pas nécessaire. C.-à-d. le dispositif devrait être authentifié avant de pouvoir utiliser ces services.
3- **Les services ouverts :** L'authentification n'est pas exigée, aucune approbation d'accès n'est exigée avant qu'on accorde l'accès de service.

L'architecture de sécurité Bluetooth permet à des applications d'imposer leurs propres politiques de sécurité. La couche liaison, sur laquelle les mécanismes de sécurité spécifiques de Bluetooth fonctionnent, est transparente aux mécanismes de sécurité imposés par les couches supérieures. Ainsi il est possible d'imposer le contrôle d'accès granuleux basé sur l'authentification de l'utilisateur dans le cadre de sécurité de Bluetooth.

2.6.4 Le contrôleur de sécurité

Les mécanismes et les politiques de sécurité qui peuvent être soutenues par Bluetooth sont possibles grâce à un composant appelé le contrôleur de sécurité «Security Manager» (Figure 2.3). Le contrôleur de sécurité est l'entité qui décide quelles politiques sont à appliquer quand une demande de connexion est faite. Basé sur le service, le type de dispositif et son niveau de fiabilité, le contrôleur de sécurité peut imposer l'authentification du niveau application, le chiffrage de la session et toutes les autres politiques spécifiques d'accès.

Le contrôleur de sécurité a besoin de l'information concernant des dispositifs comme des services avant qu'elle puisse prendre une décision. Cette information est stockée dans deux bases de données notamment, la base de données de dispositif et la base de données de service.
1- La base de données de dispositif stocke des informations sur le type de dispositif, son niveau de fiabilité et sur la longueur de clé de lien utilisée pour le chiffrage.
2- La base de données de service stocke l'information concernant les conditions d'authentification, d'autorisation et de chiffrage pour les services.

Le processus typique suivi par le contrôleur de sécurité en accordant l'accès à un dispositif pour un service particulier est comme suit :

1- Le dispositif distant demande l'accès

2- La demande de connexion vient à L2CAP

3- L2CAP demande au contrôleur de sécurité d'accorder l'accès

4- Le contrôleur questionne les bases de données de dispositif et de service

5- Si le dispositif est fiable, donc le contrôleur de sécurité peut ou ne peut pas demander l'authentification ou l'autorisation

6- Si le dispositif est non fiable, le contrôleur de sécurité peut terminer la connexion ou imposer l'autorisation. L'authentification au niveau de Bluetooth se produira quand des clés de lien sont échangées. Selon l'accès régissant la politique de sécurité, le contrôleur de sécurité pourrait inviter un protocole d'application pour imposer la sécurité de niveau d'application telle qu'un arrangement d'username/password pour l'authentification. L'appui est également incorporé pour d'autres arrangements d'authentification par l'interface de contrôleur de sécurité.

7- Le contrôleur de sécurité décidera alors si l'accès de service exige le chiffrage de lien. Si oui, des clés sont négociées et échangées au niveau du protocole L2CAP et la connexion continuera à être établie. Alternativement, si le dispositif est en mode de sécurité 3, le contrôleur de sécurité demande au LMP pour authentifier et chiffrer la communication avant que la connexion soit établie. L'architecture générale de sécurité dans Bluetooth est présentée dans la figure suivante :

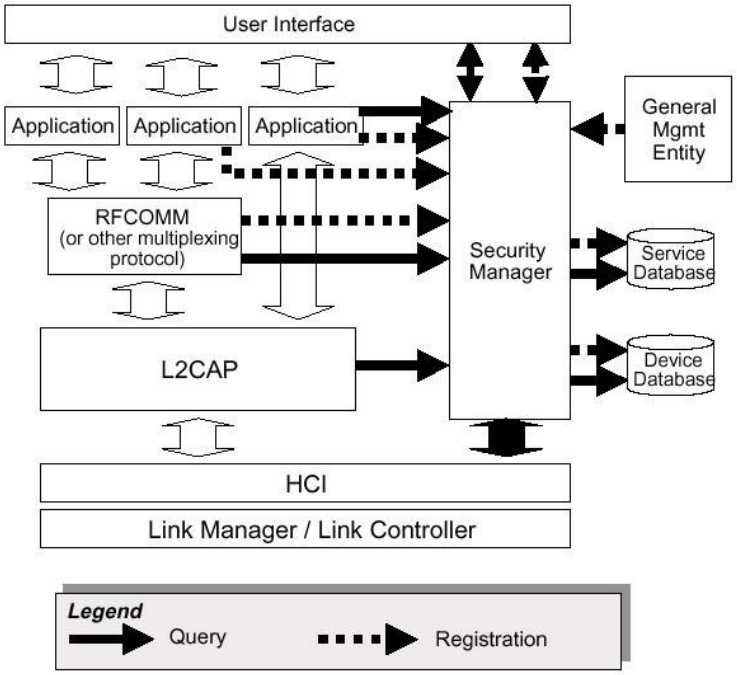

Figure 2.3 : Architecture générale de sécurité dans Bluetooth

Nous voyons ainsi que le contrôleur de sécurité est l'entité centrale qui contrôle et impose la politique de sécurité dans l'architecture de sécurité de Bluetooth.

2.6.5 La gestion des clés dans Bluetooth

Assurer une transmission sécurisée avec le protocole Bluetooth, implique l'utilisation de plusieurs genres de clés et de contrôles [28].

La gestion et l'utilisation des clés (Figure 2.4**)** permet aux protocoles Bluetooth d'établir et de maintenir une connexion entre plusieurs appareils d'un même picoréseau.

Les clés de communication peuvent être des combinaisons de plusieurs types de clés, les Unit keys, les clés de combinaisons, les clés maîtres et les clés d'initialisation.

Plusieurs entités permettent d'assurer une communication sécurisée entre plusieurs équipements Bluetooth : une adresse publique unique de 48 bit (BD-ADDR - Bluetooth Device address) utilisée pour identifier quel appareil est en train d'envoyer les données, aucun autre appareil ne doit avoir cette adresse [29-30]. Cette adresse est définie par l'IEEE (Institute of Electrical and Electronics Engineers).

Une clé secrète d'authentification de 128 bit, l'appareil qui génère aléatoirement cette clé.

Une seconde clé secrète pour le chiffrement des données, la longueur de cette clé dépend du niveau de sécurité de la transaction, cette longueur varie entre 8 et 128 bit [31]. Un nombre aléatoire est généré à chaque transaction de données, l'appareil génère automatiquement ce nombre pour l'utiliser dans les différentes processus de chiffrement et d'authentification.

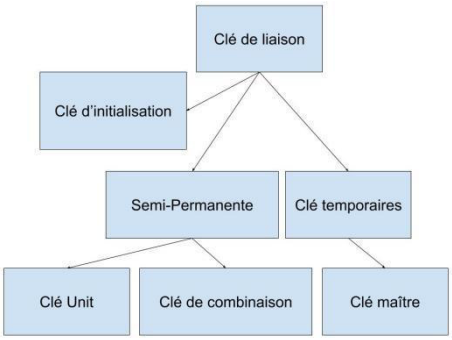

Figure 2.4 : Gestion des clés dans les protocoles bluetooth

2.6.5.1 Les clés de liaison

La clé de liaison est un nombre de 128 bit généré aléatoirement, elle est échangée entre plusieurs appareils Bluetooth pour effectuer toutes les transactions sécurisées. La clé est utilisée pour la routine d'authentification et c'est l'un des paramètres pour créer la clé de chiffrement [29].

C'est une clé semi-permanente, elle est stockée dans la mémoire vive de l'appareil et peut dont être réutilisée lorsque la session est terminée. La clé de liaison

peut s'utiliser dans le processus d'authentification entre plusieurs appareils Bluetooth. La durée de vie de cette clé dépend uniquement de la durée de vie d'une session.

2.6.5.2 Clé d'initialisation

La clé d'initialisation est utilisée pendant l'initialisation d'une communication entre deux appareils Bluetooth quand aucune clé de liaison ou autre combinaison de clé n'existe [31]. Durant l'étape d'initialisation le code PIN des deux appareils doit être saisi. La clé est le résultat de l'algorithme E22 avec le code PIN de l'appareil (Figure 2.5), l'adresse de l'autre appareil Bluetooth et un nombre 128 bit généré aléatoirement en entrée [29].

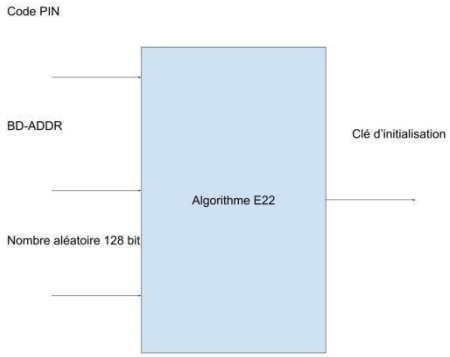

Figure 2.5 : Clé Initialisation

2.6.5.3 Clé Unit

La clé Unit est générée grâce à l'algorithme E21 quand l'appareil opère pour la première fois avec un autre appareil. Après la création, la clé est stockée dans une mémoire non volatile, elle est rarement changée. Un appareil peut utiliser la Unit Key d'un autre appareil comme une clé de liaison entre les deux appareils. Durant la phase d'initialisation, l'application décide quel appareil fournit sa Unit Key en tant que clé de liaison [31]. Si la mémoire d'un des appareils est pleine, il ne pourra pas se souvenir d'une clé supplémentaire sa clé de liaison sera donc utilisée [29].

2.6.5.4 Clé de combinaison

La clé de combinaison est construite à partir d'informations de deux appareils. Elle est créée durant la phase d'initialisation, les deux appareils génèrent leur clé en même temps [31]. Les deux appareils utilisent l'algorithme E21 avec un nombre généré aléatoirement et leur BD_ADDR en entrée. Ils échangent ensuite leurs nombre aléatoire de manière sécurisée afin de calculer leur clé de combinaison [29].

2.6.5.5 Clé Maître

La clé maître est une clé temporaire remplaçant l'actuelle clé de liaison [31]. Elle est utilisée quand l'appareil maître veut transmettre des informations à plus d'un destinataire. L'appareil maître génère une clé maître avec l'algorithme E22 et 2 nombres aléatoires de 128 bit (Figure 2.6) .

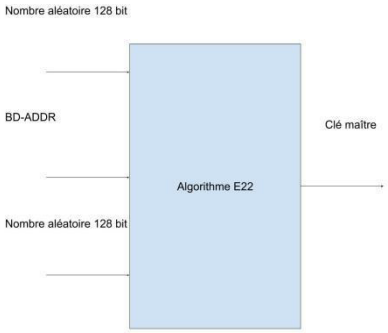

Figure 2.6 : Construction de la clé maître dans le protocole bluetooth.

2.6.6 Appairage

L'appairage (ou *pairing* en anglais) permet l'association entre deux terminaux Bluetooth. Les clefs de liaison, permettant de sécuriser les canaux de communication entre les deux terminaux, sont créés lors de cette phase.

Deux méthodes sont proposées : le « Legacy pairing » et le « Secure simple pairing » :

Le Legacy pairing est utilisé uniquement si un des terminaux n'est pas compatible avec une version supérieure à 2.0. Il se base sur un code PIN à 4 chiffres. La clef de liaison est générée à partir du code PIN. Ce mode n'offre pas de protection contre les attaques de l'intercepteur.

Le Secure simple pairing est utilisé par les terminaux supportant une norme supérieure à 2.0. Il se base sur un secret partagé par l'algorithme Diffie-Hellman. La clef de liaison est générée à partir de ce secret partagé. Ce mode offre une protection contre les attaques de l'intercepteur et contre les écoutes passives.

Les différentes phases du Secure simple pairing, définies par la norme 5.0, sont :

1. Échange des paramétrages d'entrée/sortie
2. Échange d'un secret via l'algorithme Diffie-Hellman
3. Première phase d'authentification via la vérification du code PIN généré par les terminaux. Quatre modèles d'association peuvent être utilisés

1. Comparaison numérique : peut être utilisée si les deux terminaux peuvent afficher un code PIN. Ce dernier est généré et est affiché par les deux terminaux. L'association continue après confirmation manuelle que les codes PIN sont identiques sur les deux terminaux
2. « Just work » : peut être utilisé si un des deux terminaux ne peut pas afficher de code PIN. Celui-ci est généré et l'association continue après confirmation manuelle sans vérifier que les codes PIN sont identiques. Cette méthode ne permet pas de protéger la communication d'attaque de l'intercepteur
3. Passphrase : peut être utilisé si un des deux terminaux peut afficher des chiffres et si l'autre permet d'entrer des chiffres mais n'offre pas d'affichage. L'association continue si le code PIN entré dans le deuxième terminal est identique à celui du premier terminal
4. Out-of-Band : la découverte de terminaux et l'échange des données cryptographiques pour la création des clefs sont faits hors ligne, par exemple via NFC
5. Création des clefs de liaison à partir du secret partagé lors de l'étape 2.

6. Deuxième phase d'authentification via la vérification que les 2 terminaux partagent la même clef de liaison, décrite ci-dessous
7. Création de la clef de chiffrement si l'option est activée

2.6.7 Procédures d'authentification

L'authentification bluetooth utilise une technique de question/réponse, un appareil vérifie si l'autre appareil connait la clé de liaison. Le programme procède ensuite à l'authentification avec la clé de liaison actuelle, le processus nécessite que les deux parties partagent la même clé de liaison.

Pour l'étape d'authentification, le maître envoie un numéro aléatoire au demandeur. Ensuite, les deux parties utilisent la fonction d'authentification avec le numéro généré aléatoirement et la clé de liaison pour produire une réponse signée, enfin le maître vérifie que les deux réponses sont valides [31-32].

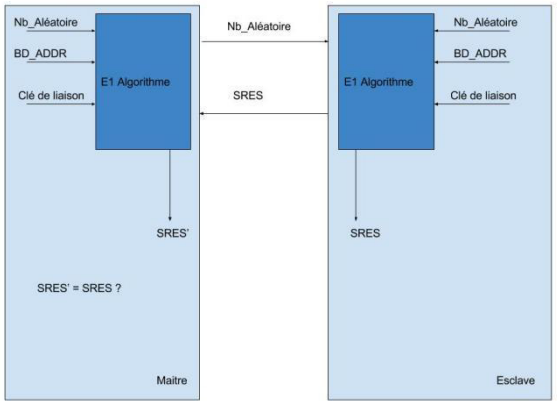

Figure 2.7 : Authentification Bluetooth

2.6.8 Procédures de chiffrement

Les données utilisateurs sont protégées par un chiffrement du paquet du payload. Le code d'accès et l'entête ne sont jamais chiffrés [33]. Le chiffrement du payload est effectué par un algorithme de chiffrement E0 qui se resynchronise à chaque nouveau payload [34].
L'algorithme de chiffrement E0 est réalisé en trois blocs :

- Initialisation de la clé du payload.
- Génération des *keys stream.*
- Réalisation du chiffrement et déchiffrement.

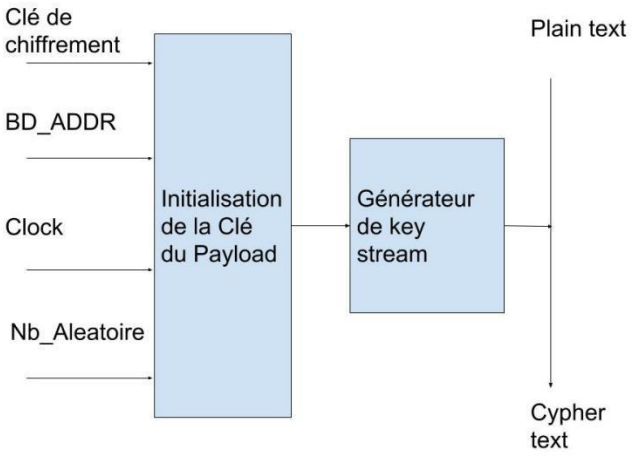

Figure 2.8: Chiffrement des données depuis un système Bluetooth.

Le bloc d'initialisation combine les bits entrants dans un ordre approprié et les décale dans les 4 LFSRs du bloc de génération des *key stream* [33]. Plusieurs modes de chiffrements sont possibles si l'appareil utilise une clé semi-permanente ou la clé maître, dans le cas où l'appareil utilise une clé semi-permanente, la diffusion en broadcast n'est pas chiffrée car la clé est utilisée pour plus d'une session donc potentiellement moins sécurisée. Individuellement, le trafic peut être chiffré ou non. Dans le cas où la clé maître est utilisée, soit rien n'est chiffré, soit la diffusion en broadcast n'est pas chiffrée mais individuellement le trafic est chiffré avec la clé maître, soit tout le trafic est chiffré avec la clé maître.

2.6.9 Les vulnérabilités dans la sécurité de Bluetooth

L'architecture de sécurité de Bluetooth, n'est pas sans faiblesse. Il y a un certain nombre de faiblesses dans cette architecture qui peuvent être exploitées. L'une des failles du protocole Bluetooth est une faille spécifique aux réseaux sans fils :

- ✓ Diverses attaques permettent de vider rapidement la batterie d'un appareil connecté en Bluetooth en provoquant une utilisation intensive et inutile du protocole. Parmi ces attaques on peut retrouver: Ping of Death Flood, BlueSpam Flood, Blueper Flood ou encore BlueSmack Flood. Les attaques Blueper Flood ou BlueSmack Flood peuvent être qualifiées de DDoS, mais lorsque la quantité de requêtes est contrôlée, le service continue de marcher normalement et consomme de la batterie.

- ✓ Un autre problème, cependant pas aussi simple à mettre en œuvre, est l'attaque «**Man In The Middle**» (MITM). L'attaque Man-in-the-Middle est une attaque où un intrus se connecte à deux terminaux et fait croire à ces terminaux qu'ils sont connectés directement l'un à l'autre, il contrôle pourtant le trafic et peut écouter ou modifier les échanges entre les deux terminaux.

- ✓ Cabir est le nom d'un ver informatique infectant les mobiles, développé pour infecter des terminaux tournant sous Symbian OS, il utilise la fonction "*Object*

Push Profile" pour se propager via Bluetooth sur les autres terminaux à portée. Ce virus est inoffensif car il ne fait que se répliquer et affiche le message "Caribe" au démarrage du téléphone, il est pourtant une preuve de concept sur le développement de virus sur mobile pouvant se répandre entre les terminaux et reste le premier ver pouvant infecter les mobiles découverts.

2.7 Conclusion :

Dans ce chapitre, nous avons donné un aperçu sur la communication Bluetooth. Nous avons débuté par présenté le protocole Bluetooth, ainsi que les différents piles composant ce protocole. Nous avons aussi abordé la sécurité dans le Bluetooth, ainsi que les mécanismes de sécurité au niveau liaison des données. Dans le chapitre suivant, nous allons implémenter et étudier différents algorithmes de cryptage dans la communication Bluetooth.

Chapitre 3: Conception et Implémentation

3.1 Introduction

Le chiffrement symétrique repose sur une clé secrète partagée, tandis que le chiffrement asymétrique utilise une paire de deux clés. Le chiffrement asymétrique relève les défis de la distribution des clés et de la communication sécurisée avec des parties inconnues. Les deux méthodes sont utilisées dans les applications modernes et servent des objectifs différents en fonction du niveau de sécurité et d'efficacité requis.

Pour augmenter le niveau de sécurité de la communication Bluetooth, nous proposons par la suite un système amélioré qui comprend le Triple DES, RSA et SH256. Le Triple DES (Variante du DES) renforce la sécurité de la transmission Bluetooth. Raison derrière pour sélectionner triple DES plutôt que Double DES c'est que en double algorithme DES, la clé utilisée pour le cryptage et le déchiffrement est soupçonné d'être une attaque de type Main-In-The-Middle (MITM). L'algorithme RSA est utilisé pour résoudre le problème de distribution des clés; et en plus de cela, SHA256 va permettre de vérifier l'intégrité du message. La combinaison de deux algorithmes cryptographiques fournit une force en matière de sécurité des données transmises par Bluetooth.

Pour mettre en œuvre cette méthodologie, nous avons utilisé deux langages de programmation à savoir le python et le C++. En effet, le python est utilisé pour la mesure de la variable temps avant le processus de cryptage et une fois le processus de cryptage terminé, afin de calculer le temps nécessaire au processus de cryptage d'un élément particulier de fichier texte en clair. Alors que le langage C++ est utilisé par la suite pour l'implémentation et la commande des cartes arduino Uno.

Nous appliquons notre méthodologie à différentes tailles d'échantillons de fichier texte que l'expéditeur souhaite transmettre via la technologie Bluetooth, et la mise en place des résultats expérimentaux, tels que la taille de fichier texte à chiffrer, la taille du fichier texte chiffré (fichier de sortie) et le temps pris par le fichier texte clair lors du cryptage.

3.2 Algorithme Triple DES

Le Triple DES exécute simplement l'algorithme DES 3 fois sur les données avec la clé spécifiée. Il permet d'éviter les problèmes liés à la taille de clé trop courte du DES. La clé fournie est divisée en 3 parties, chaque partie mesurant 8 octets (la taille de clé obligatoire pour DES).

L'algorithme triple DES utilise la méthode DES-EDE3 lorsqu'une clé de 24 octets est fournie. Cela signifie qu'il y a trois opérations DES dans la séquence chiffrer-déchiffrer-chiffrer avec les trois clés différentes. La première clé sera les octets 1 à 8, la deuxième clé les octets 9 à 16 et la troisième clé les octets 17 à 24.

Si une clé de 16 octets est fournie à la place d'une clé de 24 octets, la méthode triple DES utilisée sera DES-EDE2. Cela signifie qu'il y a trois opérations DES dans la séquence chiffrer-décrypter-chiffrer, mais la première et la troisième opérations utilisent la même clé. La première/troisième clé sera les octets 1 à 8 et la deuxième clé les octets 9 à 16.

L'algorithme triple DES (ou TDES) à 2 clés est un algorithme symétrique de chiffrement. Il fonctionne sur des blocs de texte en clair de 64 bits (ou 128 bits), et utilise des clés de 112 bits (2*56), ce qui le rend pratiquement insensible aux attaques par force brute et attaques de l'homme du milieu possibles en DES et double DES.

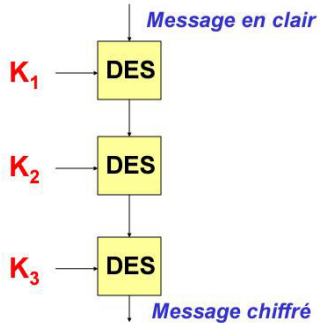

Figure 3.1 : Triple-DES avec 3 clés K1, K2 et K3.

3.2.1 Fonctionnement

Le Triple DES est généralement utilisé avec seulement deux clés différentes. Le mode d'usage standard est de l'utiliser en mode EDE (*encryption, decryption, encryption*, c'est-à-dire « chiffrement, déchiffrement, chiffrement »)[4], ce qui le rend compatible avec DES quand on utilise trois fois la même clé. Dans le cas d'une implémentation matérielle, cela permet d'utiliser le même composant pour respecter les standards DES et Triple DES.

Dans ce mode proposé, TDES s'écrit plus formellement de cette manière :

$$C = E^{k3}{}_{DES}\left(D^{k2}{}_{DES}\left(E^{k1}{}_{DES}(M)\right)\right) \qquad \textbf{3.1}$$

Une variante de Triple DES est celle de Carl Ellison, mais elle ne fait pas partie du standard défini pour 3DES :

$$\mathbf{C = E^{k3}{}_{DES}\left(T\left(E^{k2}{}_{DES}\left(T\left(E^{k1}{}_{DES}(M)\right)\right)\right)\right)} \qquad \textbf{3.2}$$

où T est une fonction de transposition destinée à augmenter la diffusion.

3.3 Algorithme RSA

Le RSA est l'un des algorithmes asymétriques les plus répandus. Dans ce système, une clé publique est utilisée pour crypter le texte, et seule une clé privée, connue uniquement du destinataire, est utilisée pour le décrypter. Ce qui le rend crucial est l'impossibilité de retrouver la clé privée à partir de la clé publique.

Il a été développé en 1978 par Ronald L. Rivest, Adi Shamir et Leonard M. Adleman, des chercheurs du MIT (Massachusetts Institute of Technology). Actuellement, il est largement utilisé dans divers domaines, notamment dans les navigateurs Internet pour sécuriser les sites web et pour le chiffrement des e-mails. De plus, il est adopté comme standard de chiffrement dans le secteur bancaire de nombreux pays.

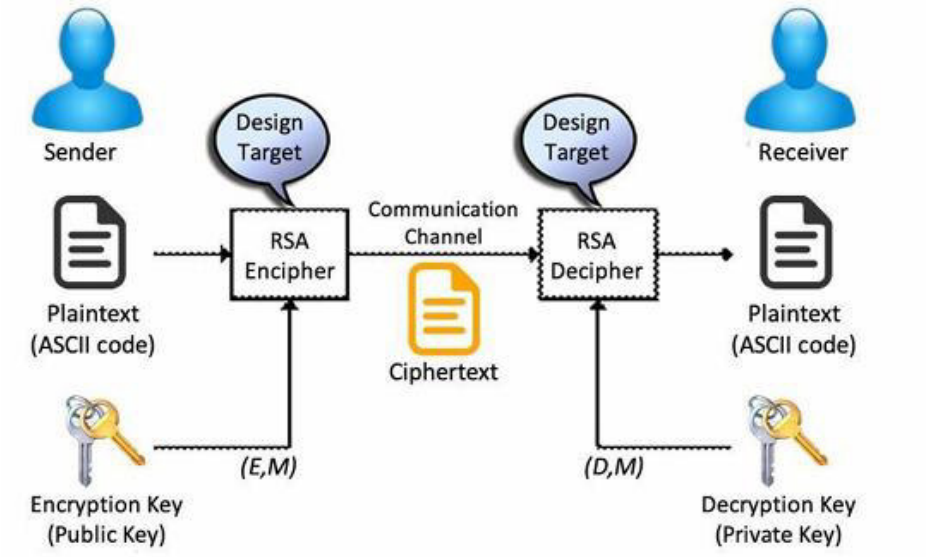

Figure 3.2: Structure de l'algorithme RSA

3.3.1 Description du protocole:

L'objectif principal de l'algorithme est de permettre la transmission d'un message codé, qui ne peut être déchiffré que par le destinataire désigné, assurant ainsi que seules les parties légitimes puissent accéder au contenu du message. Dans ce contexte, nous identifierons le destinataire du message sous le nom de B, tandis que l'émetteur sera désigné par A.

3.3.2 Déroulement de l'algorithme :

Le destinataire du message B peut maintenant partager sa clé publique, représentée par la paire d'entiers (N, e), tandis que sa clé privée est d. La sécurité du protocole RSA repose sur le fait que B est le seul à connaître la factorisation de N. Cela lui permet de calculer φ(N) et, par conséquent, d'inverser e pour obtenir sa clé privée d. Désormais, toute personne disposant de la clé publique de B peut lui envoyer des messages sécurisés. Il est important de noter qu'Alice n'a joué aucun rôle dans la création de la clé publique de B, mais elle peut maintenant lui envoyer des messages sécurisés conformément au protocole RSA.

3.3.3 Limites du RSA:

Lorsque nous programmons un logiciel de cryptage / décryptage RSA, un nouveau problème se pose : le dépassement de capacité des variables. En effet, le type d'entier le plus grand disponible dans le langage C est le type unsigned long int, capable de gérer des nombres allant jusqu'à 32 bits au maximum. Cependant, cette capacité est largement insuffisante pour un cryptage efficace. Ainsi, notre programme se limitera à fonctionner avec des "petits nombres", c'est-à-dire des nombres p et q compris entre 0 et 255. En conséquence, la clé n sera comprise entre 0 et 65535. Malheureusement, en raison de ces limitations, il ne sera pas possible de crypter par blocs de plusieurs caractères, car cela nécessiterait que n soit supérieur à 65536 pour permettre le codage par blocs de 2 caractères.

3.3.4 Le chiffrement RSA

Voici les étapes principales du processus de chiffrement RSA :

1. Sélection de deux grands nombres premiers, p et q, qui serviront à générer les clés.
2. Calcul de N, un composant crucial des clés, en effectuant le produit de p et q.
3. Détermination de l'exposant public, e, de manière à ce qu'il soit premier avec (p-1)*(q-1), c'est-

à-dire que le PGCD(e,φ(N))=1. Le couple (N, e) forme la clé publique de chiffrement.

4. Le message à chiffrer doit être préalablement converti en nombres, par exemple en utilisant les valeurs ASCII de chaque lettre ou en les remplaçant par leur rang dans l'alphabet.
5. Le message codé, noté M, est divisé en blocs de taille inférieure à N, et chaque bloc est crypté individuellement pour former le texte chiffré C. En pratique, les blocs et les clés sont constitués de plusieurs chiffres.
6. Le cryptage s'effectue en calculant le reste de la division de chaque bloc codé par N.
7. Le processus de déchiffrement RSA comprend les étapes suivantes :

- ❖ Calcul de la clé privée d à partir de p et q, en satisfaisant l'équation d. e ≡ 1 mod φ(N).
- ❖ Utilisation de la clé d pour décrypter chaque bloc du texte chiffré, récupérant ainsi le message initial M.
- ❖ Le déchiffrement est réalisé en calculant le reste de la division de chaque bloc crypté par N.

Exemple 1 :

- p = 5 et q = 17
- n = 85
- ϕ(n) = 64

Remarque : Le calcul de ϕ(n) nécessite la connaissance de la décomposition de n en facteurs premiers p × q, ce qui reste secret même si n est public.

Exemple 2 :

- p = 101 et q = 103
- n = 10 403
- ϕ(n) = 10 200

Alice procède ensuite comme suit :

- Elle choisit un exposant e tel que le PGCD(e,ϕ(n)) = 1.
- En utilisant l'algorithme d'Euclide étendu, elle calcule l'inverse d de e modulo ϕ(n) : d × e ≡ 1 (mod ϕ(n)).

Exemple 1 :

- Alice choisit e = 5, où PGCD(5, 64) = 1.
- En appliquant l'algorithme d'Euclide étendu, elle trouve que l'inverse de e modulo ϕ(n) est d = 13.

Exemple 2 :

- Alice choisit e = 7, avec PGCD(7, 10 200) = 1.
- En utilisant l'algorithme d'Euclide étendu, elle obtient que l'inverse de e modulo ϕ(n) est d = 8743.

La clé publique d'Alice est constituée des deux nombres n et e, qu'elle partage avec le monde entier.

Exemple 1 :

- n = 85 et e = 5

Exemple 2 :

- n = 10 403 et e = 7

Quant à sa clé privée, Alice la garde secrète, ne divulguant que le nombre d.

Exemple 1 :

- d = 13

Exemple 2 :

- d = 8743

En secret, Alice détruit les valeurs de p, q et ϕ(n) qu'elle n'a plus besoin de conserver, préservant ainsi sa clé privée.

3.4 Algorithme SHA256

Le SHA - secure hash algorithm - est un algorithme de hachage utilisé par les autorités de certification pour signer certificats et CRL (certificate revocation list). Introduit en 1993 par la NSA avec le SHA0, il est utilisé pour générer des condensats uniques (donc pour "hacher") de fichiers.

Exemple : Un fichier haché en SHA1 pourrait donner :
752c14ea195c369bac3c3b7896975ee9fd15eeb7

Comme toute solution cryptographique, le SHA se doit d'évoluer en même temps que les capacités de calcul de nos ordinateurs et éviter de devenir vulnérable.

Il existe donc plusieurs versions de SHA : SHA0 (obsolète puisque totalement vulnérable), SHA1 (actuellement le plus utilisé), SHA2 (qui nous intéresse) et enfin le tout dernier SHA3 né en 2012.

Le SHA2 comprend 4 types de hash : SHA224, SHA256, SHA384 et SHA512. Il fonctionne sur le même principe que SHA1 mais est plus résistant aux attaques et donne un condensat plus long.

Il existe plusieurs propriétés uniques qui sont communes aux fonctions de hachage cryptographiques :

- **L'unicité** : une fonction de hachage doit produire une empreinte de hachage unique pour chaque entrée donnée. Cela signifie qu'il ne doit pas y avoir deux entrées différentes qui produisent la même empreinte de hachage. Par exemple, si nous utilisons une fonction de hachage pour hacher la chaîne "Bonjour", elle devrait produire une empreinte de hachage différente de celle produite par la chaîne "Salut".
- **La rapidité** : une fonction de hachage doit être rapide à calculer pour une entrée donnée. Si une fonction de hash prend trop de temps à calculer, elle peut être inefficace pour des applications qui nécessitent un traitement rapide de grandes quantités de données.
- **La sécurité** : une fonction de hachage doit être sécurisée, c'est-à-dire qu'il doit être difficile de trouver deux entrées différentes qui produisent la même empreinte de hachage (une collision).

L'algorithme de hachage SHA-256 a pour intérêt d'être rapide à calculer et sécurisé, car il est très difficile de trouver deux entrées qui produisent la même empreinte de hachage. Autrement, cet algorithme propose un juste équilibre entre sécurité et coût.

3.5 Solution proposée

La solution proposée doit couvrir les besoins de confidentialité et d'authentification. La confidentialité sera couverte par l'utilisation de deux algorithmes TDES/RSA en exploitant les avantages et inconvénients de ces deux types d'algorithmes de cryptographie. En effet, l'algorithme TDES est pratiquement insensible aux attaques par force brute et attaques de l'homme du milieu, tandis que l'algorithme RSA est éventuellement vulnérable à une attaque MITM assez poussée, dans laquelle un pirate impose la clé publique. L'authentification sera renforcée par l'utilisation d'une fonction de hachage basée sur la méthode SHA256.

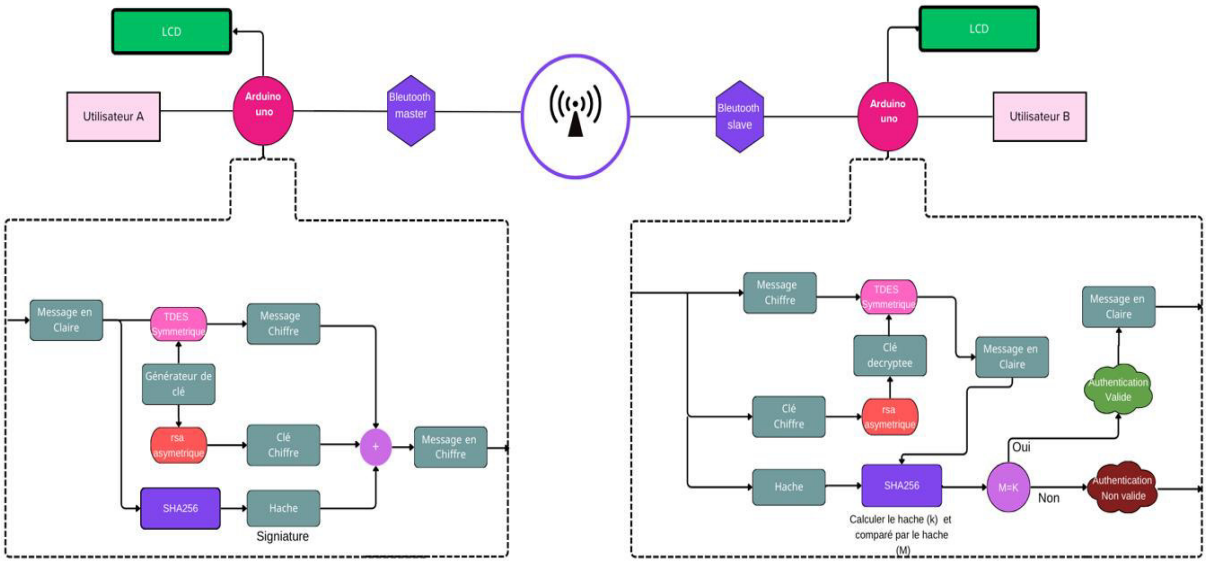

Figure 3.3: Solution proposée

3.6 Méthodologie de Conception

Cette méthodologie de recherche comprend les processus suivants.

3.6.1 Processus de cryptage

1. Le génération de nombre aléatoire génère une clé (KA).
2. L'algorithme TDES chiffre le message en claire (M) avec la clé symétrique (KA), et produit un texte chiffré (CT).
3. L'algorithme RSA chiffre la clé (KA) et produit le texte chiffré de la clé (CKA);
4. Combinez un texte chiffré (CT), un texte chiffré de clé (CKA) produit un message complexe (CM).
5. Le Message complexe (CM) est envoyé au récepteur B.

3.6.2 Processus de décryptage

1. Le récepteur B a reçu le Message complexe (CM) en trois parties, une est le texte chiffré de la clé CKA issu de l'algorithme de chiffrement RSA, l'autre est le texte chiffré CT de l'algorithme triple DES, plus la signature (Msign).
2. Le récepteur B déchiffre lui-même le texte chiffré de la clé (CKA) en utilisant la clé privée (KB), et récupère la clé symétrique (KA), puis décrypter le texte chiffré (CT) en utilisant l'algorithme T-DES.
3. On compare le message en claire (M) avec la signature (Msign) afin de vérifier l'authenticité du message d'origine.

3.7 Génération de nombres aléatoire

Une clé secrète est générée à l'aide d'un générateur de nombres aléatoires ou d'un algorithme sécurisé. La longueur de la clé dépend du niveau de sécurité requis par l'expéditeur et le destinataire. En général, les clés plus longues sont considérées comme plus sûres, car elles offrent plus de

combinaisons possibles à un attaquant. Nous allons utiliser le modèle hexadécimal pour générer nos clés aléatoires. Voici un exemple de clé privé obtenu lors de l'exécution du code RSA.

Figure 3.4: Exemple de clé privée RSA

3.8 Mesure du temps d'exécution

Nous allons réaliser plusieurs séries de tests afin de constater quelles parties du processus étaient les plus chronophages. Lors de nos différents tests, nous avons fait varier la taille des messages envoyés, ainsi que la taille de la clé.

3.8.1 Temps d'exécution d'un algorithme RSA

Le Code développé en python utilise la bibliothèque Python **rsa** pour générer des paires de clés publiques et privées, puis chiffre et déchiffre un message à l'aide de ces clés. Le programme boucle 50 fois, chaque fois générant de nouvelles paires de clés et mesurant le temps nécessaire pour cette opération, ainsi que le temps nécessaire pour chiffrer et déchiffrer un message donné avec ces clés.

À chaque itération de la boucle, le programme effectue les étapes suivantes :

1. Génération de nouvelles clés publiques et privées de taille 4048 bits à l'aide de la fonction **rsa.newkeys()**.

2. Mesure du temps nécessaire pour cette opération de génération de clés.

3. Chiffrement d'un message (**mess**) en utilisant la clé publique générée à l'étape précédente avec **rsa.encrypt()**.

4. Déchiffrement du message chiffré en utilisant la clé privée correspondante avec **rsa.decrypt()**.

5. Mesure du temps nécessaire pour le chiffrement et le déchiffrement du message.

À la fin des 50 itérations, le programme affiche la moyenne des temps de génération de clés (**generation time**) et la moyenne des temps de chiffrement et de déchiffrement du message (**mid generation time**). Ces mesures donnent une idée du temps moyen requis pour ces opérations avec les paramètres donnés.

Tableau 1. Temps de chiffrement et déchiffrement de l'algorithme RSA d'un message de taille 14bits

Taille de clés	Temps pour chiffrer et déchiffrer en seconds
520 bits	0.07282625675201417 e-05
1024 bits	0.43430975914001463 e-05
2024 bits	3.9352388191223144 e-05
4048 bits	49.617960929870605 e-05

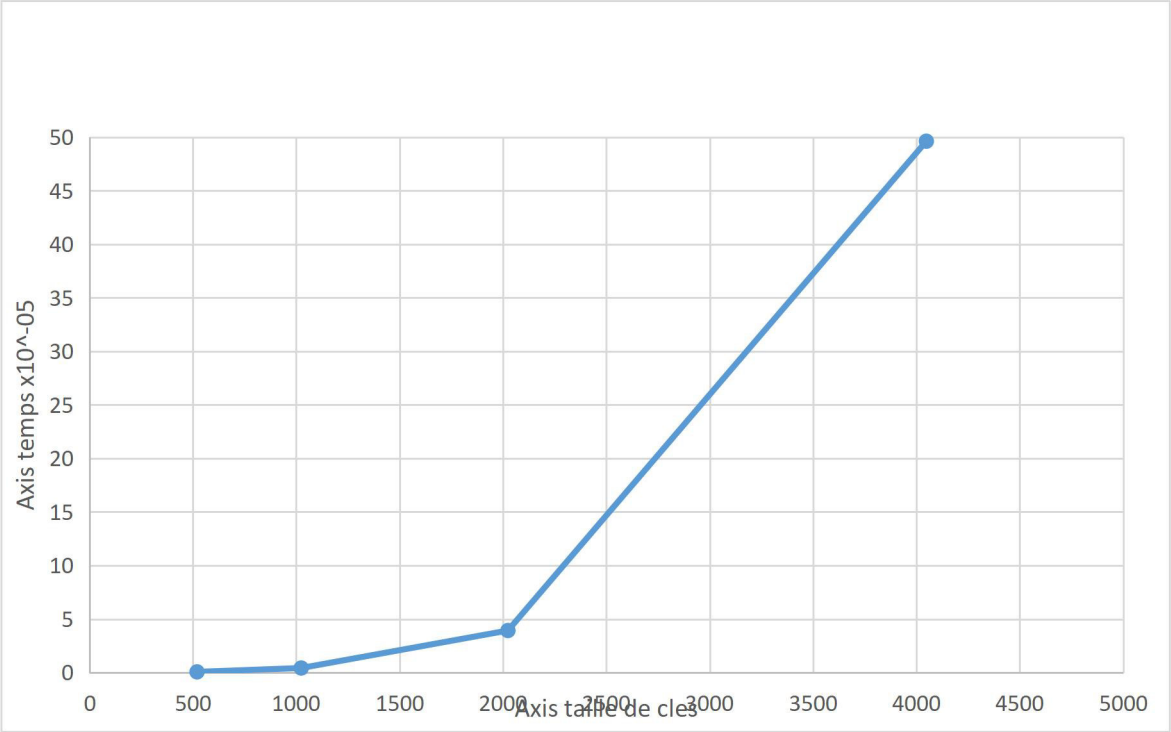

Figure 3.5: Variation du temps en fonction de la taille de la clé RSA

3.8.2 Temps d'exécution d'un algorithme TDES:

Ce programme mesure le temps moyen nécessaire pour générer une clé 3DES, chiffrer un message variable (défini en byte) en mode CFB, et affiche ce temps en secondes après avoir effectué 250 itérations. L'utilisation de `time.time()` permet de mesurer le temps d'exécution en secondes pour chaque itération. Le programme gère également les exceptions potentielles lors de la génération de la clé pour s'assurer qu'une clé valide est toujours utilisée.

Tableau 2. Temps de chiffrement et déchiffrement de l'algorithme T-DES d'un message

Taille de clés TDES	Taille de message	Temps en secondes
128 bits	1000 bits	0.004862079620361328
128 bits	2000 bits	0.00898435115814209
128 bits	4000 bits	0.011673517227172851
128 bits	8000 bits	0.018417558670043944
128 bits	16000 bits	0.07077588558197022
128 bits	32000 bits	0.13473851203918458
128 bits	64000 bits	0.27265560150146484

3.9 Embarquement du système sur une carte Arduino UNO:

L'Arduino est une plate-forme de prototypage d'objets interactifs à usage créatif constituée d'une carte électronique et d'un environnement de programmation. Cet environnement matériel et logiciel permet à l'utilisateur de formuler ses projets

Au cours de notre projet, nous avons utilisé la carte arduino UNO (Figure 3.6). Elle dispose de 14 Entrées / Sorties et mesure 68,6 mm x 53,3 mm. L'Arduino UNO est une multiplateforme qui peut fonctionner sous windows, macintosh et linux, avec un environnement de programmation clair et simple. De plus, c'est une carte qui n'est pas chère, en open source qui contient de nombreuses bibliothèques disponibles avec diverses fonctions implémentées.

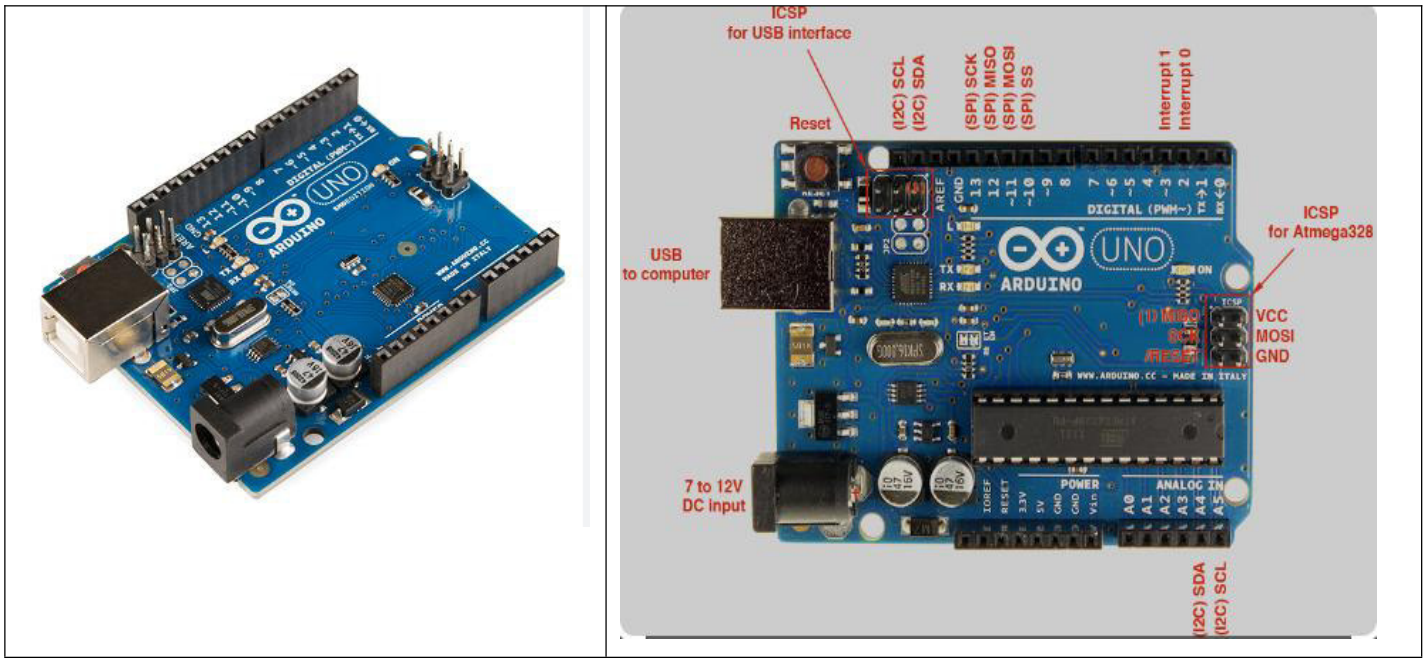

Figure 3.6: Carte Arduino Uno

3.9.1 Les modules Bluetooth HC-05 et HC-06

Il existe deux sortes de module Bluetooth, tous deux compatibles Aruino et utilisables sur un breadboard (plaque d'essai en français).

On les distingue par le nombre de pattes d'entrées / sorties :

- HC-05 : 6 sorties. Ce module peut être « maître » (il peut proposer à un autre élément bluetooth de s'appairer avec lui) ou « esclave » (il ne peut que recevoir des demandes d'appairage). Ce module fait l'objet d'un autre article car il y a quelques différences pour le régler (**Figure 3.7**).

- HC-06 : 4 sorties. Ce module ne peut être qu'esclave. C'est ce module que nous utilisons dans ce projet (**Figure 3.68**).

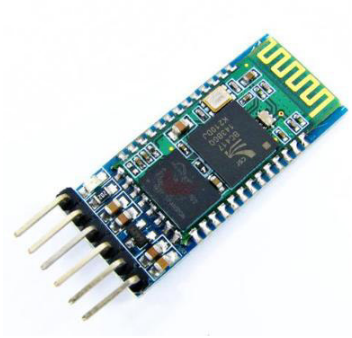

Figure 3.7: Bluetooth HC-05

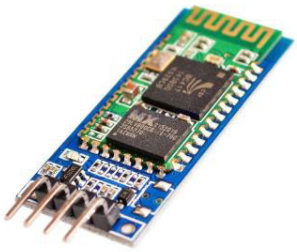

Figure 3.8: Bluetooth HC-06

3.9.2 Partie logicielle

3.9.2.1 Programmation de l'Arduino

En parallèle au matériel, un IDE (Environnement de développement) à été développé afin de permettre la programmation des modules de l'Arduino. Cet IDE est une application Java libre servant d'éditeur de codes (programmes) et de compilateur. Les opérations de compilation et de chargement dans la mémoire du microcontrôleur sont alors ramenées à de simples clics. La communication en Arduino et le PC se fait via le port USB. L'interface de l'IDE Arduino est plutôt simple (**Figure 3.9**), il offre une interface minimale et épurée pour développer les codes. Il est doté d'un éditeur de codes avec coloration syntaxique et d'une barre d'outils rapide. On retrouve aussi une barre de menus plus classique qui est utilisée pour accéder à différente fonctionnalités de l'IDE. Enfin une console affichant les résultats de la compilation du code source, des opérations sur la carte, … etc.

3.9.2.2 Structure d'un projet Arduino

L'outil de développement d'Arduino impose au programmeur une structure de programme.
1. La fonction main () imposée et non modifiable.
2. La fonction setup () qui doit contenir les différentes initialisations du code.
3. La fonction loop () : Elle doit contenir les boucles, les fonctions répétée indéfiniment, elle est exécutée après la fonction setup.

3.9.2.3 Communication de l'Arduino

La carte Arduino-Uno dispose de toute une série de facilités pour communiquer avec un ordinateur, une autre carte Arduino ou une avec d'autres microcontrôleurs. L'ATMega 328 dispose d'une UART un émetteur -récepteur asynchrone universel pour une communication série disponible au niveau des broches 0 (RX) et 1 (TX). Un circuit intégré ATMega 8U2 sur la carte assure la connexion entre cette communication série vers le port USB de l'ordinateur et apparait alors comme un COM virtuel pour les logiciels de l'ordinateur.

L'IDE Arduino inclut une fenêtre terminal série (moniteur série) sur l'ordinateur qui permet d'envoyer des textes simple depuis et vers la carte Arduino (**Figure 3.11**).

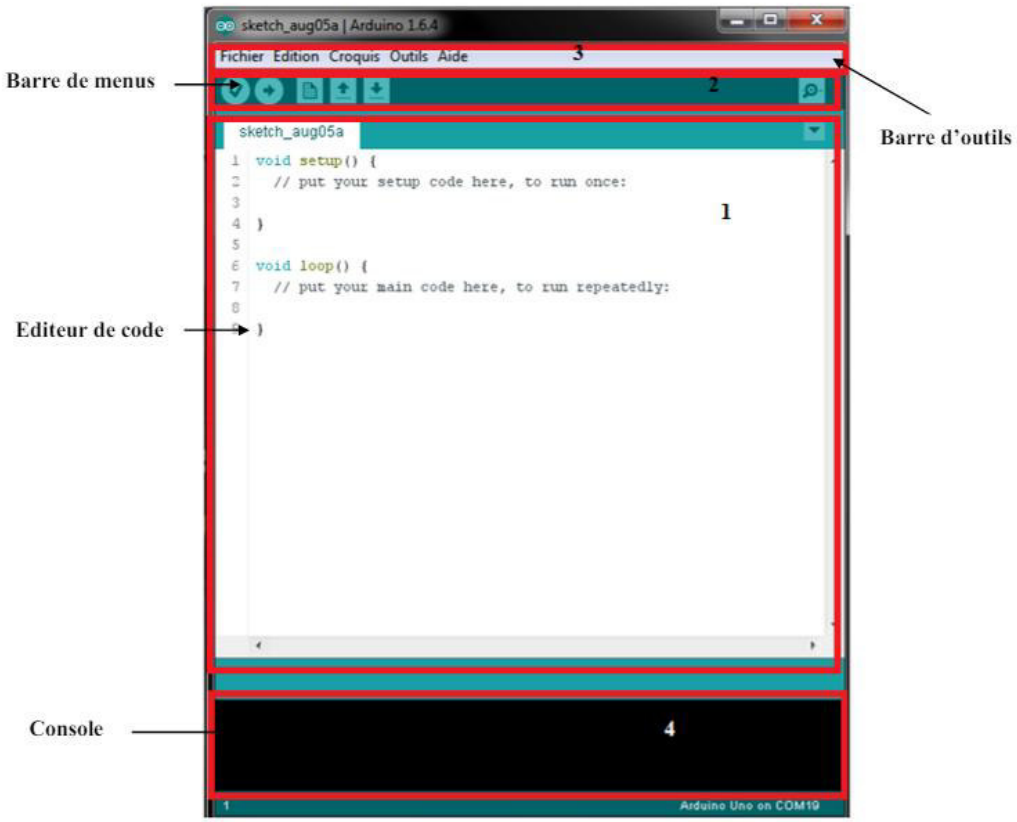

Figure 3.9: L'interface de l'IDE Arduino

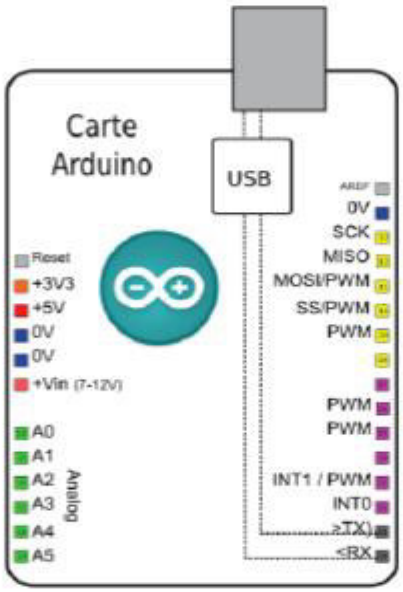

Figure 3.10: Brochages de la carte Arduino

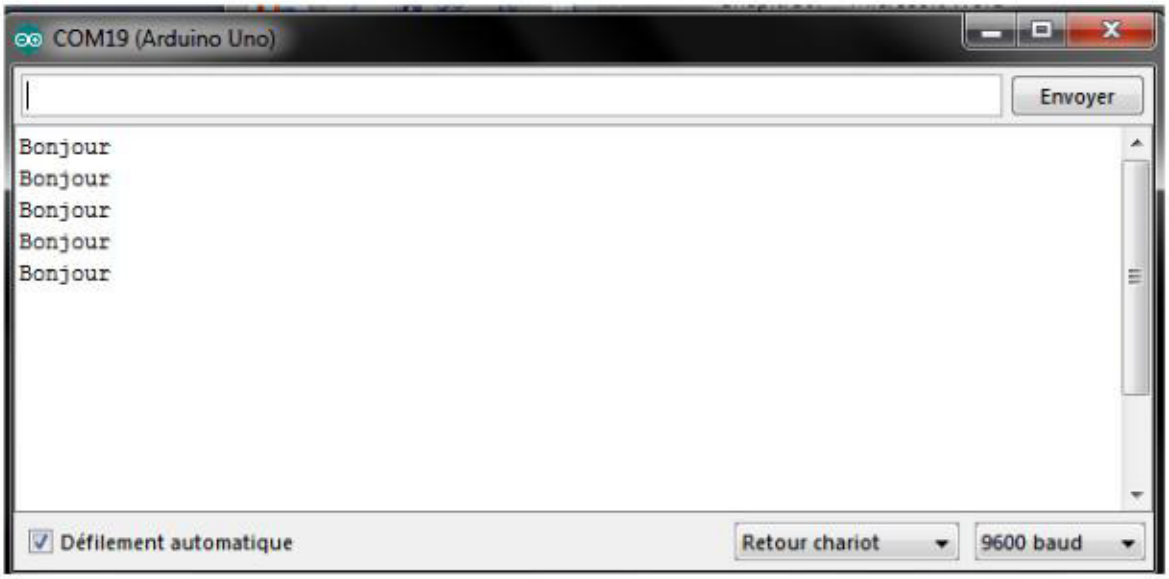

Figure 3.11: Interface du moniteur série d'Arduino.

3.9.2.4 Configuration d'un HC avec une carte Arduino :

Connectez le module HC-05 à votre Arduino (**Figure 3.12**). :

- HC-05 VCC à Arduino 5V.
- HC-05 GND à Arduino GND.
- HC-05 TX à Arduino RX (pin 0).
- HC-05 RX à Arduino TX (pin 1).

Utilisez le code suivant pour configurer le module HC-05 avec Arduino (**Figure 3.13**).

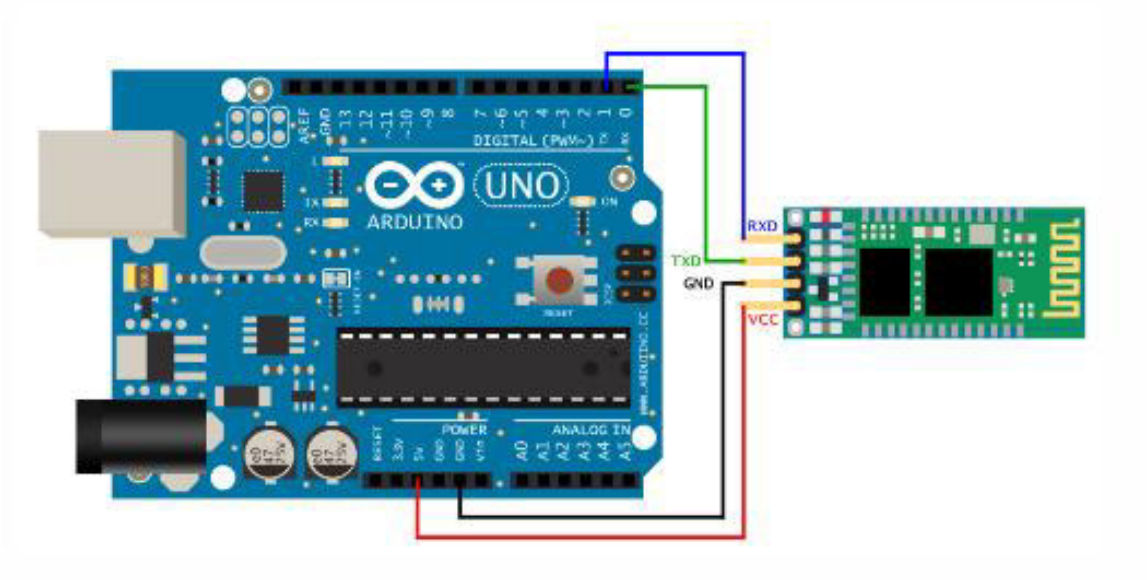

Figure 3.12: Connexion du Bluetooth avec l'Arduino.

Figure 3.13: Code suivant pour configurer le module HC-05 avec Arduino.

- Configuration du module HC-05:

1. Assurez-vous que le module HC-05 est déconnecté de l'Arduino.
2. Branchez l'Arduino à votre ordinateur via USB.
3. Ouvrez l'IDE Arduino et téléchargez le code ci-dessus sur votre Arduino.
4. Débranchez l'Arduino.
5. Connectez le module HC-05 à l'Arduino.
6. Ouvrez le moniteur série de l'IDE Arduino et réglez la vitesse à 9600 bauds.
7. Branchez l'Arduino à nouveau.

Vous devriez voir les informations provenant du module HC-05 s'afficher dans le moniteur série.

- Configuration AT Commande:

- Entrez en mode AT en envoyant "AT" à partir du moniteur série. Vous devriez recevoir une réponse "OK"
- AT : Vérifier la connexion avec le module
  AT+NAME : Voir le nom du module
  AT+ADDR : Voir l'adresse du module
  AT+VERSION : Connaître la version
  AT+UART : Connaître la vitesse de connexion
  AT+ROLE: Voir le rôle du module (1=master/ 0=slave/ 2=esclave boucle)
  AT+RESET : Redémarrage du module et sortir du mode AT
  AT+ORGL : Restaurer le module d'usine
  AT+PSWD: Consulter le mot de passe
  AT+BIND=adresse,du,slave (on a remplacé les : par des ,) , permet de connecter un module master à un slave

3.9.2.5 Configuration d'un écran LCD I2C avec une carte arduino

Un écran LCD (Liquid Crystal Display) est un outil couramment utilisé pour afficher des informations dans divers projets électroniques. L'écran LCD I2C (Inter-Integrated Circuit) simplifie le câblage en réduisant le nombre de connexions nécessaires entre l'Arduino et l'écran (**Figure 3.14**).

- Matériel Nécessaire :

- Arduino Uno (ou autre modèle compatible)
- Écran LCD I2C (16x2)
- Câbles de connexion (Dupont)

- Schéma de Connexion :

- **GND** de l'écran LCD I2C connecté à **GND** de l'Arduino.
- **VCC** de l'écran LCD I2C connecté à **5V** de l'Arduino.
- **SDA** (Serial Data) de l'écran LCD I2C connecté à **A4** de l'Arduino
- **SCL** (Serial Clock) de l'écran LCD I2C connecté à **A5** de l'Arduino.

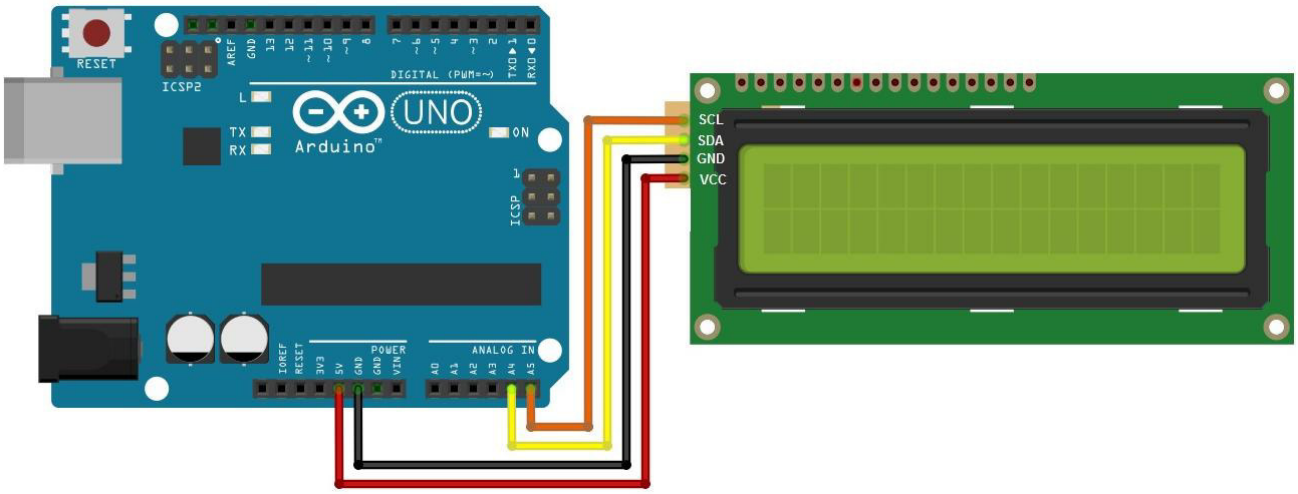

Figure 3.14: Connexion d'un écran LCD I2C avec Arduino.

3.10 Communication entre deux Arduino Uno configurés en Maître/esclave

Afin de valider nos résultats de simulation, selon la solution proposée en figure 3.3, nous avons réalisé un système de communication via une liaison série en utilisant deux cartes Arduino, deux Bluetooth HC05 configurés en Maître/esclave (**Figure 3.15**). Cette liaison s'établit sur deux broches RX et TX définies dans notre programme en tant que broches 9 et 8. Pour l'affichage des résultats, nous avons ajoutés deux écrans de type LCD 1602A. Dans la partie soft, nous avons réalisé aussi, sous le logiciel python, une interface graphique (**Figure 3.16**) qui communique avec le programme développé en C++ pour le pilotage des matériels électroniques de notre système radiofréquence Bluetooth.

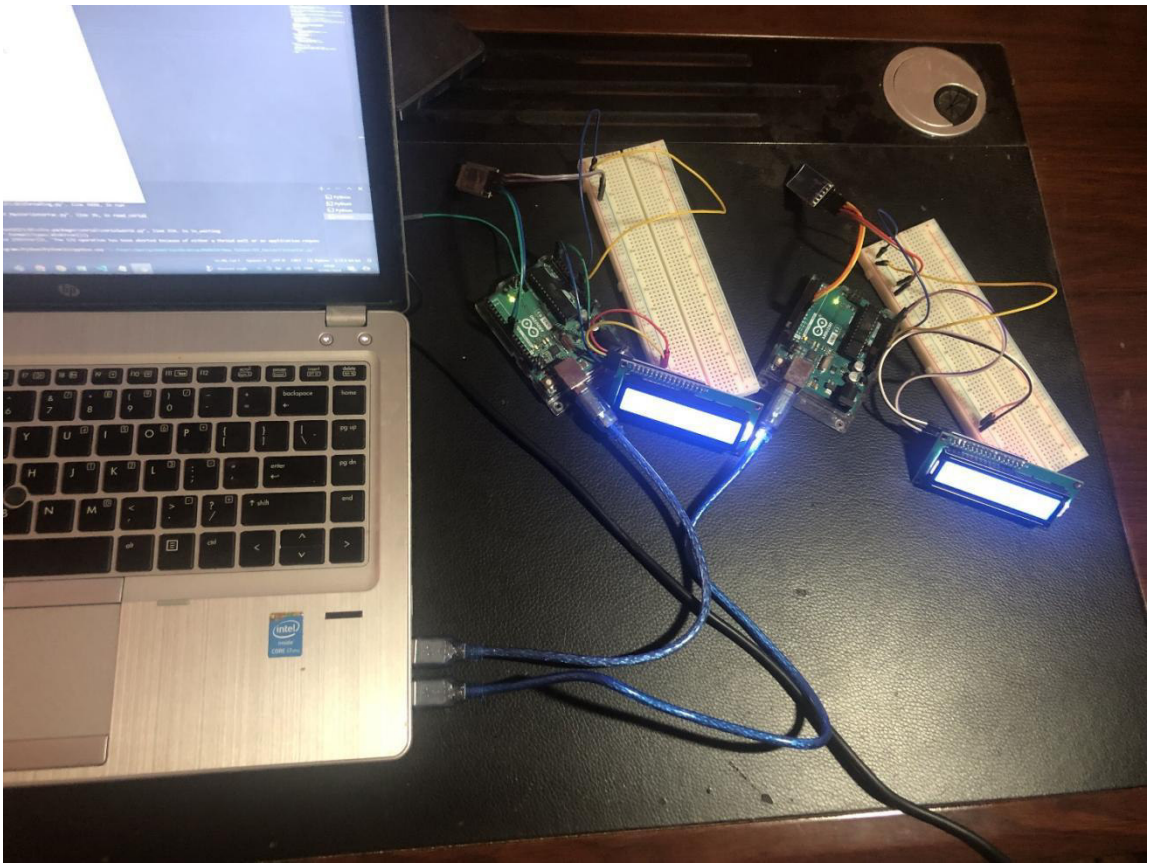

Figure 3.15: Communication entre deux Arduino-Uno via Bluetooth avec PC.

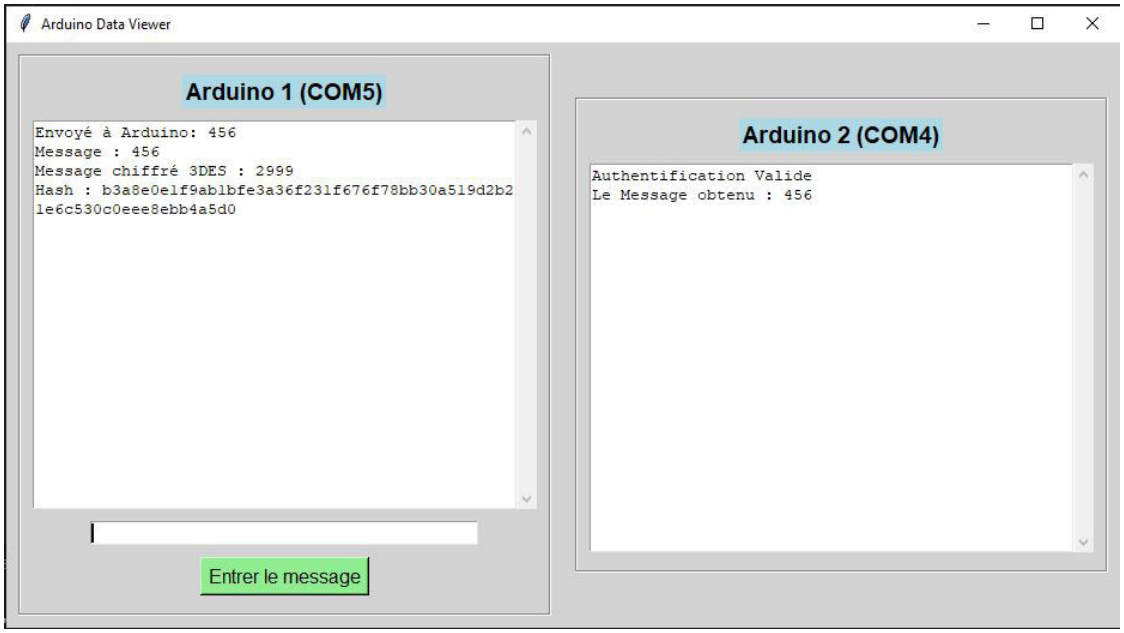

Figure 3.16: Interface d'entrée/sortie entre l'Arduino Master et l'Arduino Slave

Conclusion générale

Conclusion générale

Notre projet a pour objet de l'étude et l'implémentation d'une communication Blutooth sécurisée sur deux cartes Arduino Uno, et en utilisant deux bluetooth HC05 en mode maitre-escalave.

Comme solution possible, nous proposons l'hybridation de deux algorithmes de cryptographie à savoir l'algorithme symétrique TDES et l'algorithme asymétrique RSA. La solution proposée doit couvrir les besoins de confidentialité et d'authentification. La confidentialité sera couverte par l'utilisation des deux algorithmes TDES/RSA en exploitant les avantages et inconvénients de ces deux types d'algorithmes de cryptographie. En effet, l'algorithme TDES est pratiquement insensible aux attaques par force brute et attaques de l'homme du milieu, tandis que l'algorithme RSA est éventuellement vulnérable à une attaque MITM assez poussée, dans laquelle un pirate impose la clé publique. L'authentification sera renforcée par l'utilisation d'une fonction de hachage basée sur la méthode SHA256.

Tenant compte du temps qui a été imparti à notre projet, nous avons implémenté et testé notre solution sur une communication Blutooth sécurisée sur deux cartes Arduino Uno, et en utilisant deux bluetooth HC05 en mode maitre-escalave.

Comme perspective à notre travail, nous proposons d'entreprendre une étude comparative avec d'autres types d'algorithmes de cryptographies. Aussi de pousser cette étude au cas de générateur de nombre aléatoire. Et pour finir, ce système hybride semble bien adapter pour d'autres applications informatiques.

[1] K. Sakamura. The Tron Project. *IEEE Micro*, 7(2) :8–14, April 1987.

[2] H. Takada and K. Sakamura. Compact, low-cost, but real-time distributed computing for computer augmented environments. In *Proceedings of the Fifth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, pages 56–63, Aug 1995.

[3] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3) :66–75, September 1991.

[4]Auguste Kerckhoffs. La cryptographie militaire. IX :5–83, Jan 1883.
http: //www.petitcolas.net/fabien/kerckhoffs/.

[5]Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana and Chicago, 1949.

[6]Paul Van De Zande. The Day DES Died, July 2001.
https://www.sans.org/reading-room/whitepapers/vpns/day-des-died-722.

[7]David McNett. US Governement's Encryption Standard Broken in Less than a Day. Technical report, 1999.
http://www.distributed.net/images/d/d7/19990119_-_PR_-_release-des3.pdf.

[8]Anne Canteaut. *A5/1*, pages 1–2. Springer US, Boston, MA, 2011.
https: //doi.org/10.1007/978-1-4419-5906-5_332.

[9]Morris J. Dworkin. SP 800-38D. Recommendation for Block Cipher Modes of Operation : Galois/Counter Mode (GCM) and GMAC. Technical report, Gaithersburg, MD, United States, 2007.

[10]wolfSSL. The Benefit of Stream Ciphers, 2010.
https://www.wolfssl.com/the-benefit-of-stream-ciphers/.

[11]Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Message Authentication using Hash Functions-The HMAC Construction. *CryptoBytes*, 2, 1996.

[12]ISO/IEC 9797-1 :1999 Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1 : Mechanisms using a block cipher. Standard, International Organization for Standardization, 1999.

[13]Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22 :644–654, 1976. https://ee.stanford.edu/~hellman/publications/24.pdf.

[14]R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Commun. ACM*, 21(2) :120–126, February 1978. http://doi.acm.org/10.1145/359340.359342.

[15]Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177) :203–209, 1987.
http://www.jstor.org/stable/2007884.

[16]Victor S. Miller. Use of Elliptic Curves in Cryptography. In *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[17]ECRYPT. Algorithms, Key Size and Protocols Report (2018), 2018.
http:// www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf.

[18]F. OMARY, "Applications des algorithmes évolutionnistes à la cryptographie ", Thèse de Doctorat d'état, Université MOHAMMED V – AGDAL RABAT (MAROC), 26. Juillet. 2006.

[19]A. K. BENHAOUA, "Approche cryptographie base sur les algorithmes génétique pour la sécurité des réseaux Ad hoc ", Mémoire de Magistère, Université D'ORAN ES-SENIA, (Algérie), 2005.

[20]O. AZZOUZI, "Système embarqué flexible pour un chiffrement hybride symétrique/asymétrique ", Mémoire de Magister, Ecole nationale supérieure d'informatique, (Algérie) ,2014.

[21]M. VIDEAU, "Critère de sécurité des algorithmes de chiffrement à clé secrète ", Thèse de Doctorat, Université de Paris 6, (France), 10. Novembre. 2005.

[22]http://www.urec.cnrs.fr/IMG/GER/-060124-les algorithmes de cryptographie.pdf.
[23]Biaoshuai Tao and Hongjun Wu. Improving the Biclique Cryptanalysis of AES. In *Information Security and Privacy*, pages 39–56, Cham, 2015. Springer International Publishing.
[24]Bluetooth, The Bluetooth Specification, v.1.0B
http://www.bluetooth.com/developer/specification/specification.asp
[25]An overview of Bluetooth security, Nikhil Anand,
http://www.giac.org/practical/gsec/Nikhil_Anand_GSEC.pdf
[26]Redjem Bouhenguel, Imad Mahgoub et Mohammad Ilyas, « Bluetooth Security in Wearable Computing Applications », *2008 International Symposium on High Capacity Optical Networks and Enabling Technologies*, novembre 2008, p. 182-186
(ISBN 978-1-4244-2960-8, DOI 10.1109/HONET.2008.4810232)
[27]Bluetooth security, Juha T. Vainio.
http://www.niksula.cs.hut.fi/~jiitv/bluesec.html
[28]Évaluation des vulnérabilités de Bluetooth (ITSPSR-17), rapport sur les produits et systèmes de sécurité TI.
http://www.csecst.gc.ca/fr/services/publications/itspsr/itspsr17.html
[29]Paraskevas Kitsos, Nicolas Sklavos Sklavos, Kyriakos Papadomanolakis et Odysseas Koufopavlou, « Hardware implementation of Bluetooth security », *IEEE Pervasive Computing*, vol. 2, nº 1, janvier 2003, p. 1096-1102 (ISBN 978-1-4673-2843-2, ISSN 1536-1268, DOI 10.1109/MPRV.2003.1186722)
[30]Markus Jakobsson, Susanne Wetzel et David Naccache, « Security Weaknesses in Bluetooth », *Lecture Notes in Computer Science*, vol. 2020, avril 2001, p. 176-191 (ISBN 978-3-540-41898-6, ISSN 0302-9743, DOI 10.1007/3-540-45353-9_14)
[31]Paraskevas Kitsos, Nicolas Sklavos Sklavos, Kyriakos Papadomanolakis et Odysseas Koufopavlou, « Hardware implementation of Bluetooth security », *IEEE Pervasive Computing*, vol. 2, nº 1, janvier 2003, p. 1096-1102 (ISBN 978-1-4673-2843-2, ISSN 1536-1268, DOI 10.1109/MPRV.2003.1186722)
[32]« An analysis of Bluetooth security vulnerabilities », *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, 16-20 march 2003, p. 1825-1831 vol.3 (ISBN 0-7803-7700-1, ISSN 1525-3511, DOI 10.1109/WCNC.2003.1200664)
[33]Paraskevas Kitsos, Nicolas Sklavos Sklavos, Kyriakos Papadomanolakis et Odysseas Koufopavlou, « Hardware implementation of Bluetooth security », *IEEE Pervasive Computing*, vol. 2, nº 1, janvier 2003, p. 1096-1102 (ISBN 978-1-4673-2843-2, ISSN 1536-1268, DOI 10.1109/MPRV.2003.1186722)

RESUME :

Le projet porte sur l'étude et l'implémentation d'une communication Bluetooth sécurisée entre deux cartes Arduino Uno, en utilisant des modules Bluetooth HC-05 configurés en mode maître-esclave. Le contexte de ce projet s'inscrit dans l'importance croissante de la communication sans fil et la nécessité de sécuriser ces échanges contre diverses menaces. Nous proposons une solution hybride combinant l'algorithme symétrique TDES et l'algorithme asymétrique RSA pour assurer la confidentialité des communications Bluetooth. L'algorithme TDES est choisi pour sa résistance aux attaques par force brute et man-in-the-middle (MITM), tandis que l'algorithme RSA assure la robustesse contre les attaques de substitution de clé publique. Une fonction de hachage SHA256 est utilisée pour renforcer l'authentification. Enfin, nous avons implémenté et testé une communication Bluetooth sécurisée entre deux Arduino Uno en utilisant d'un coté les modules HC-05 en mode maître-esclave, et d'un autre coté les trois algorithmes TDES, RSA et SHA256 pour la partie logicielle.

Mots clés : IoT, Bluetooth, TDES, RSA, SHA256, Arduino Uno, HC05, Confidentialité, Authentification..

The project involves the study and implementation of secure Bluetooth communication between two Arduino Uno boards, using HC-05 Bluetooth modules configured in master-slave mode. The context of this project is part of the growing importance of wireless communication and the need to secure these exchanges against various threats. We propose a hybrid solution combining the symmetric TDES algorithm and the asymmetric RSA algorithm to ensure the confidentiality of Bluetooth communications. The TDES algorithm is chosen for its resistance to brute force and man-in-the-middle (MITM) attacks, while the RSA algorithm provides robustness against public key substitution attacks. A SHA256 hash function is used to strengthen authentication. Finally, we implemented and tested secure Bluetooth communication between two Arduino Unos using on one side the HC-05 modules in master-slave mode, and on the other hand the three algorithms TDES, RSA and SHA256 for the software part. .

Keywords: IoT, Bluetooth, TDES, RSA, SHA256, Arduino Uno, HC05, Confidentiality, Authentication.

ملخص:

يتضمن المشروع دراسة وتنفيذ اتصال Bluetooth آمن بين لوحتي Arduino Uno، باستخدام وحدات Bluetooth HC-05 التي تم تكوينها في الوضع الرئيسي والتابع. ويأتي سياق هذا المشروع ضمن الأهمية المتزايدة للاتصالات اللاسلكية والحاجة إلى تأمين هذه التبادلات ضد التهديدات المختلفة. نقترح حلاً هجينًا يجمع بين خوارزمية TDES المتماثلة وخوارزمية RSA غير المتماثلة لضمان سرية اتصالات Bluetooth. تم اختيار خوارزمية TDES لمقاومتها لهجمات القوة الغاشمة وهجمات الرجل في الوسط (MITM)، بينما توفر خوارزمية RSA المتانة ضد هجمات استبدال المفتاح العام. يتم استخدام وظيفة التجزئة SHA256 لتعزيز المصادقة. أخيرًا، قمنا بتنفيذ واختبار اتصال Bluetooth الآمن بين اثنين من Arduino Unos باستخدام وحدات HC-05 من جهة في الوضع الرئيسي والتابع، ومن ناحية أخرى الخوارزميات الثلاث TDES وRSA وSHA256 لجزء البرنامج. .

الكلمات الرئيسية: إنترنت الأشياء، بلوتوث، TDES، RSA، SHA256، Arduino Uno، HC05، السرية، المصادقة.