REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ MOULOUD MAMMERI DE TIZI-OUZOU
FACULTÉ DE GÉNIE ELECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Mémoire De Fin d'études En Vue De l'Obtention D'un Master Académique

Filière : Informatique

Spécialité : Réseaux, Mobilité et Systèmes Embarqués

Thème

Cryptage RSA par une communauté de Smartphones

Présenté par :

BENMADDI Tarik

BOULAOUCHE Abdessalam

Encadré par :

M^{me} HADAOUI

Promotion 2019/2020

Remerciements

Au terme de ce travail, on tient à remercier

Monsieur, DAOUI

pour avoir accepté de présider le jury de notre

soutenance.

Ainsi l'examinateurs M^{me} ACHEMOUKH

Notre encadreur M^{me} HADAOUI

pour ses qualités humaines et

professionnelles, pour son encadrement, ses directives,

ses remarques constructives, et sa disponibilité.

Dédicaces

Je dédie ce modeste travail à

Mes parents,

Mes sœurs,

Et tous mes amis.

BENMADDI Tarik

Dédicaces

Je dédie ce modeste travail à

Mes parents,

Mes frères,

Et tous mess amis.

BOULAOUCHE Abdessalam

Table des matières

Table des acronymes

AES: Advanced Encryption Standard

CIA: Confidentiality, Integrity and Availability

IOT: Internet of Things

ISO: International Organization for Standardization

PGDC : Plus Grand Diviseur Commun

RSA : Rivest, Shamir and Adleman

Liste des Figures :

Liste des tables :

Liste des algorithmes

Introduction Générale

Contexte et motivation

De nos jours, les équipements mobiles connectés au réseau internet sont en constante évolution grâce à l'apparition de nouvelles technologies de communication et de nouveaux réseaux comme les réseaux sociaux.

Cependant le grand nombre de smartphones connectés nous offre de nouvelles possibilités de partage de ressources. La mise en commun de ces ressources offre une puissance de calcul qu'on peut exploiter pour faire des calculs complexes, notamment dans la cryptographie.

La cryptographie est un outil très utilisé dans le domaine de la sécurité pour assurer la confidentialité et l'authentification des données. Celle la se divise en deux opérations le chiffrement et le déchiffrement. Le chiffrement en question se fait grâce à plusieurs algorithmes dont RSA.

Notamment tous les algorithmes de cryptages exigent des ressources importantes et le plus souvent s'exécutent sur des ordinateurs à usage général. Ces algorithmes ne conviennent pas à un environnement à faibles ressources.

Problématique et Contribution

Dans ce projet, nous nous intéressons à la confidentialité des données dans des systèmes à ressources limitées en utilisant l'algorithme RSA. Le chiffrement et le déchiffrement RSA sont des opérations de type exponentiation modulaire sur de grands nombres (taille de la clé RSA doit être supérieure à 1024 bits). L'application de cette solution nécessite donc des contraintes physiques importantes comme : l'énergie, la capacité mémoire et la puissance de calcul. Ce qui complique son exécution sur des systèmes à faibles ressources. Notre Contribution consiste à adapter l'algorithme RSA sur des systèmes à petites ressources (puissance faible de calcul, petite mémoire et énergie limitée). Pour cela, nous proposons une démarche originale qui consiste à réaliser un calcul réparti de cryptage RSA par une communauté de smartphones.

Organisation du projet

Notre projet comprend deux partie. Chaque partie est composée de deux chapitres.

Partie I

Composée de deux chapitres (1 et 2) cette partie présente l'état de l'art. Le premier chapitre définit la sécurité informatique. Nous évoquons également la cryptographie en exposant des algorithmes de cryptage. Le deuxième chapitre abordera l'algorithme RSA que nous exploitons dans ce project. Ce dernier parlera aussi à propos de la cryptanalyse.

Partie II

Cette partie présente notre contribution qui consiste à réaliser un crypto-système RSA distribué sur des smartphones. Elle est composée de deux chapitres. Le chapitre 3 décrit l'architecture du RSA distribué. La réalisation du projet en question est présentée dans le chapitre 4.

Partie I : Etat de l’art

Chapitre 1 : Cryptographie et sécurité

1.1 Introduction

Dans ce chapitre, nous allons introduire la sécurité informatique qui assure la confidentialité, l'intégrité et la disponibilité les trois objectifs de la sureté informatique. Nous abordons la cryptographie qui a pour but d'atteindre ces objectifs. Nous présentons les méthodes de cryptage populaires. Nous évoquons également la signature numérique, le certificat et les fonctions de hachage.

1.2 La sécurité informatique

La sécurité informatique c'est l'ensemble des moyens mis en œuvre pour réduire la vulnérabilité d'un système contre les menaces accidentelles ou intentionnelles. L'objectif de la sécurité informatique est d'assurer que les ressources matérielles et/ou logicielles d'un parc informatique sont uniquement utilisées dans le cadre prévu et par des personnes autorisées.

Les exigences fondamentales en sécurité informatique selon les utilisateurs de systèmes informatiques se caractérisent en :

- **La confidentialité** – Seules les personnes habilitées doivent avoir accès aux données.
- **L'intégrité** – Il faut garantir à chaque instant que les données qui circules sont bien celles que l'on croit, qu'il n'y a pas eu d'altération (volontaire ou non) au cours de la communication.
- **La disponibilité –** Il faut s'assurer du bon fonctionnement du système, de l'accès à un service et aux ressources à n'importe quel moment.
- **La non-répudiation** - Une transaction ne peut être niée par aucun des correspondants.
- **L'authentification** – Elle limite l'accès aux personnes autorisées.

1.2.1 Les objectifs de la sécurité informatique

Selon la norme [ISO 7498-2], les principaux objectifs sont :

- Empêcher la divulgation non-autorisée des données,
- Empêcher la modification non-autorisée des données,
- Empêcher l'utilisation non-autorisée des ressources matérielles ou logicielles.

Afin d'atteindre ces objectifs, différents schémas ont vu le jour. On site :

1.2.1. a. CIA (1987)

Le triangle de CIA (Confidentiality, Integrity, Availability) présente les grands axes de la sécurité. Ce schéma a pour but de garantir : *La Confidentialité, L'Intégrité, La Disponibilité*. Il est utilisé comme une base a d'autres modèles.

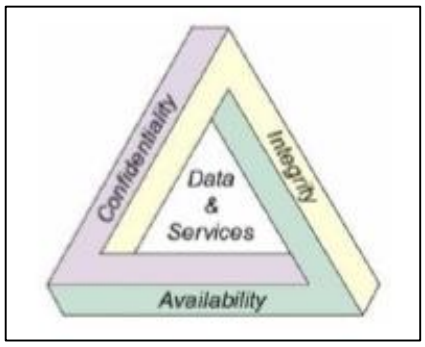


Figure 1 : CIA

1.2.1.b. Pentagone de confiance (2006)

Défini par Piscitello en 2006, il précise la notion d'accès à un système. Indépendamment des notions définies dans le triangle CIA, le modèle de Piscitello précise la confiance que peut/doit avoir l'utilisateur en présence d'un système informatisé.

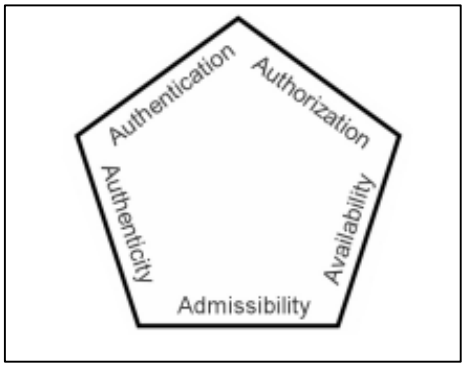


Figure 2 : Le pentagone de confiance

On y retrouve les notions d'authentification, de disponibilité, d'autorisation et d'intégrité (Authenticity). Mais on y découvre un nouveau thème : l'admissibilité : La machine sur laquelle nous travaillons, à laquelle nous nous connectons, est-elle fiable ? En d'autres termes, peut-on faire confiance à la machine cible ?

1.3 La cryptographie

La cryptographie a été inventée, à ses débuts, dans le but de sécuriser des communications, jugées secrètes par l'expéditeur. Étymologiquement, le mot cryptographie vient du grec « kryptos » qui signifie cacher, et « graphein » signifiant écrire. Ainsi la cryptographie est l'étude des différentes méthodes et techniques mathématiques reliées aux aspects de sécurité de l'information, pour assurer le secret et l'authenticité des messages, elle permet de stocker des informations sensibles ou de les transmettre à travers des réseaux non sûrs (comme Internet) de telle sorte qu'elles ne peuvent être lues par personne à l'exception du destinataire convenu. Elle concerne la transformation d'un message (texte, image, chiffres) intelligible vers un message codé, incompréhensible à tous sauf pour les détenteurs d'un code de déchiffrement. Il ne faudra d'ailleurs pas confondre avec la sténographie, qui consiste à dissimuler un message sans transformation « logique ».

Quelques exemples de procédés sténographiques utilisés

L'encre invisible : Au 1er siècle av. J-C, apparu une méthode de dissimulation de message consistant à écrire avec du lait ou du jus de citron. Il suffisait de chauffer le support pour faire réapparaître le message.

Le crâne rasé : A l'antiquité, les Grecs utilisaient parfois la tête des esclaves pour tatouer des messages. Ils étaient ensuite envoyés chez le destinataire et après un long voyage, il fallait raser leur tête dont les cheveux avaient repoussé, pour lire le message.

Le micro point : Pendant la seconde guerre mondiale, les Allemands utilisaient la technique du micro-point, consistant à dissimuler des photos ou des textes dans un point de ponctuation d'une lettre. Il suffisait alors d'agrandir le point pour voir apparaître le message.

La cryptographie utilise des concepts issus de nombreux domaines (Informatique, Mathématiques, Electronique). Toutefois, les techniques évoluent et trouvent aujourd'hui régulièrement racine dans d'autres branches (Biologie, Physique, etc.)

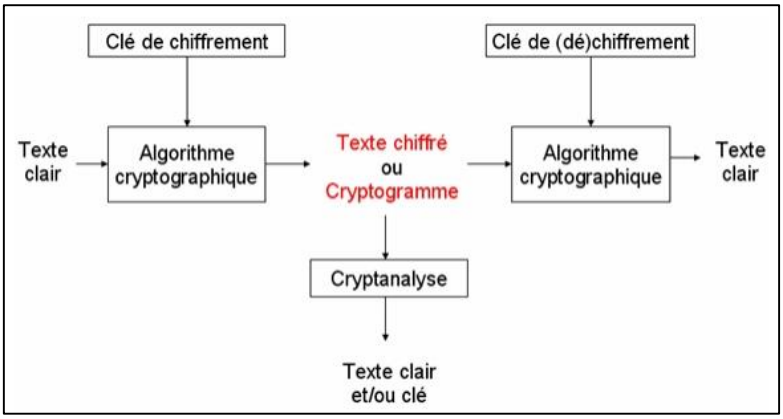


Figure 3 : Protocole de chiffrement

Cryptologie : Il s'agit d'une science mathématique comportant deux branches : la cryptographie et la cryptanalyse.

Cryptographie : La cryptographie est l'étude des méthodes donnant la possibilité d'envoyer des données de manière confidentielle sur un support donné.

Chiffrement : Le chiffrement consiste à transformer une donnée (texte, message, ...) afin de la rendre incompréhensible par une personne autre que celui qui a créé le message et celui qui en est le destinataire. La fonction permettant de retrouver le texte clair à partir du texte chiffré porte le nom de déchiffrement.

Texte chiffré : Appelé également cryptogramme, le texte chiffré est le résultat de l'application d'un chiffrement à un texte clair.

Clef : Il s'agit du paramètre impliqué et autorisant des opérations de chiffrement et/ou déchiffrement. Dans le cas d'un algorithme symétrique, la clef est identique lors des deux opérations. Dans le cas d'algorithmes asymétriques, elle diffère pour les deux opérations.

Cryptanalyse : Opposée à la cryptographie, elle a pour but de retrouver le texte clair à partir de textes chiffrés en déterminant les failles des algorithmes utilisés.

Crypto système : Il est défini comme l'ensemble des clés possibles (espace de clés), des textes clairs et chiffrés possibles associés à un algorithme donné.

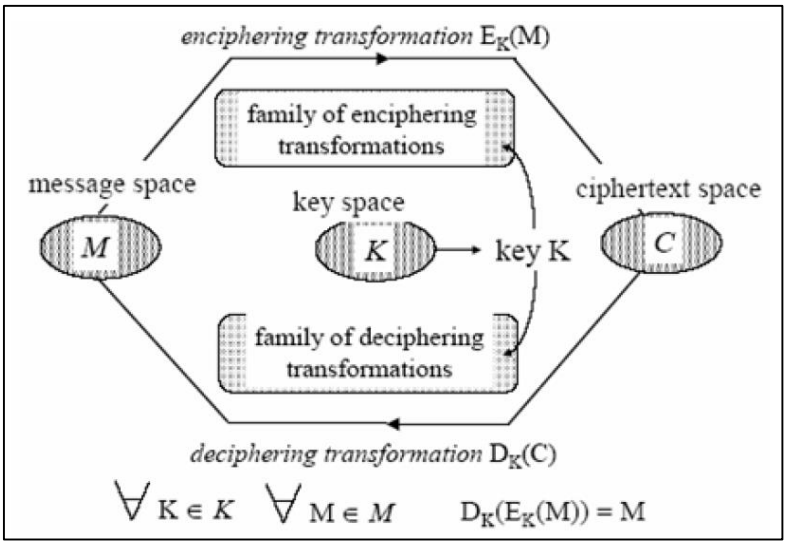


Figure 4 : Schéma d'un Crypto système

L'algorithme est un ensemble de trois algorithmes :

- L'un génère les clés K,
- Un autre pour chiffrer M (message clair),
- Un troisième pour déchiffrer C (message crypté).

1.3.1 Cryptographie à clé symétrique

Les algorithmes de ces types de chiffrement se nomment aussi 'privée' ou conventionnels. Leurs caractéristiques sont comme suit :

- Les clés sont identiques : KE = KD = K,
- La clé doit rester secrète,
- Les algorithmes les plus répandus sont le DES, AES, …
- Ces algorithmes sont basés sur des opérations de transposition et de substitution des bits du texte clair en fonction de la clé,
- La taille des clés est souvent de l'ordre de 128 bits. Le DES utilise 56, par contre l'AES peut aller jusqu'à 256.

- L'avantage principal de ce mode de chiffrement est sa rapidité,
- Son désavantage est dans la distribution des clés avant l'utilisation, ainsi la taille de la clé doit être très grande sinon le risque qu'une partie tierce découvre la clé augmente.

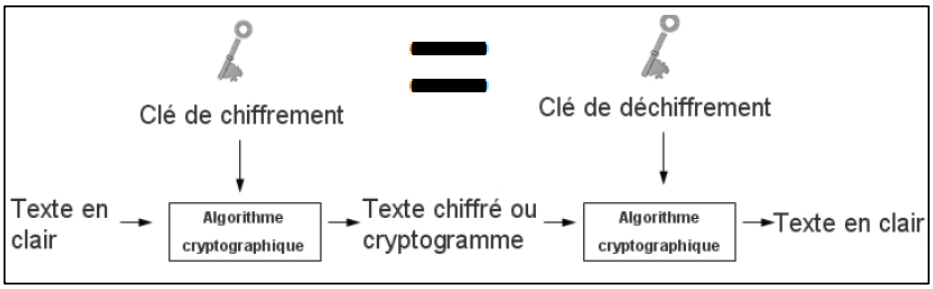


Figure 5 : Chiffrement symétrique

1.3.2 Cryptographie à clé asymétrique

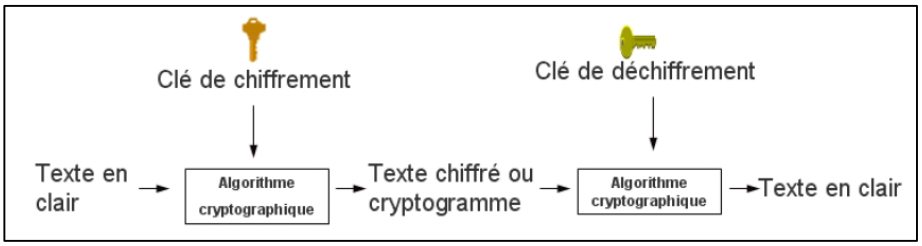


Figure 6 : Chiffrement asymétrique

Caractéristiques :

- Une clé publique P_K (symbolisée par la clé verticale),
- Une clé privée secrète S_K (symbolisée par la clé horizontale),
- Propriété : La connaissance de P_K ne permet pas de déduire S_K,
- DS_K(EP_K (M)) = M,
- Le principe des algorithmes de ce genre de chiffrement s'agit d'une fonction unidirectionnelle à trappe. Facile à calculer dans un sens, mais difficile voire impossible dans le sens inverse,
- La taille des clés s'étend de 512 bits à 2048 bits en standard,
- L'algorithme de cryptographie asymétrique le plus connu est le RSA,
- Au niveau des performances, le chiffrement par voie asymétrique est environ 1000 fois plus lent que le chiffrement symétrique.

1.4 Fonction de hachage cryptographique et signature numérique

1.4.1 Fonction de hachage cryptographique

Est un algorithme qui permet d'assurer l'intégrité des données et des fichiers. Ce mécanisme est utilisé par les signatures numériques qui permettent d'authentifier les expéditeurs. Il est également utilisé pour vérifier des mots de passe et pour identifier des données ou des fichiers, pour générer des mots de passe ou des nouvelles clés à partir des clés ou des mots de passe sécurisés. La fonction de hachage cryptographique notée ***H***() permet d'associer à une donnée notée m de taille arbitraire une image y de taille fixe et beaucoup plus petite que la taille de ***m***. Généralement la taille de y varie entre 128 et 512 bits.

La fonction de hachage cryptographique doit posséder les propriétés suivantes :

- **Fonction Irréversible** : très facile à calculer l'empreinte d'un message ***m***. mais il est très difficile de trouver le message ***m*** à partir de son empreinte,
- **Fonction sensible** : c'est-à-dire impossible de modifier un message m sans modifier son empreinte,
- **Fonction injective** : impossible de trouver deux messages ***m*1** et ***m*2** différents ayant la même empreinte (si ***H***(***m*1**)= H(***m*2**) alors ***m*1**= ***m*2**).

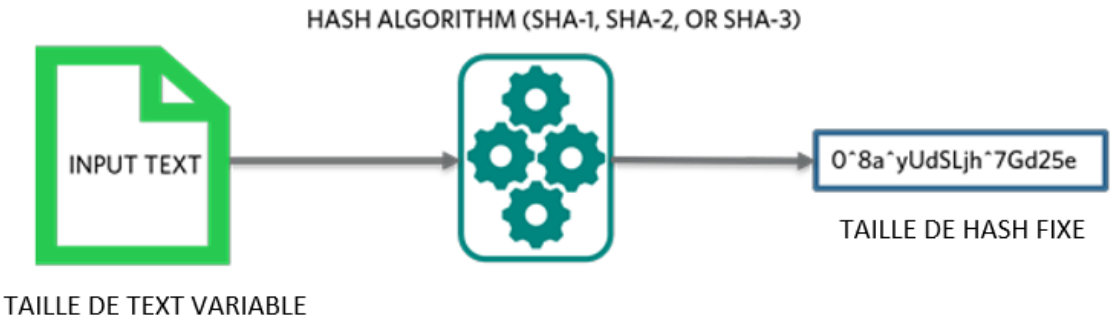


Figure 7 : hachage cryptographique

Les fonctions de hachages les plus utilisées sont: SHA-1 et SHA-2. Mais SHA-1 ne peut plus être utilisé pour générer des empreintes. Il peut, à la rigueur, être utilisé pour des contrôles d'intégrité mais pour des raisons de rétrocompatibilité.**[N1]**

Selon la norme **FIPS 180-2 [N2]**: SHA256 et SHA512 qui, comme leurs noms l'indiquent, génèrent des empreintes de 256 bits et 512 bits.
Donc **SHA2** est le nouveau standard, **SHA-3** est une alternative

1.4.2 Signature numérique

Il est nécessaire de vérifier la provenance des données et leur intégrité, car la confidentialité n'est pas suffisante. Ceci est ce que les signatures numériques nous offre. La signature numérique doit présenter les propriétés suivantes :

- **Authentique :** la signature doit permettre au récepteur du message d'identifier son expéditeur (identifier la personne qui a apposé la signature),
- **Infalsifiable** : la signature ne peut être falsifiée (on ne peut pas utiliser la signature d'une tierce personne),
- **Non réutilisable :** la signature doit dépendre du message à signer. Deux messages différents impliquent deux signatures différentes,
- **Inaltérable** : un message signé ne peut plus être modifié (message n'a pas été altéré entre le moment de sa signature et le moment de sa réception),
- Signature facile à mémoriser,
- **Irrévocable** : le signataire ne peut pas nier sa signature.

La signature numérique permet donc de garantir l'identité de l'émetteur du message, l'intégrité des données reçues et la non répudiation.
De nos jours, la plupart des signatures numériques se basent sur la cryptographie asymétrique. U1 veut envoyer un message m à U2. Celui-ci doit vérifier son authenticité. Le principe de génération et de vérification de signature est comme suit :

Coté U1 :

Entrée : Un message $m \in N$.
Sortie : la signature du message m.

1: Générer une clé privée k_{pr} et une clé publique k_{pb},
2: Envoyer la clé publique k_{pb} à *U2*,

Utiliser les certificats de clés publiques

3 :Calculer $H(m)$,

H(m)le hachage du message m,

4 : Calculer $S_m=C(k_{pr}, H(m))$,

$C(k_{pr}, H(m))$ est le chiffrement du *hachage du message m, en utilisant la clé privée* k_{pr},

5: **return** le couple (m,S_m).

Algo 1: Algorithme de génération de la signature

Coté U2 :

Entrée : Un message *m et sa signature :*(m,S_m)
Sortie : Une vérification de l'authenticité

1: Calculer $s = H(m)$,

La fonction H() idem à celle utilisée par U1

2: Calculer $Ds_m=C(k_{pb}, S_m)$,

$C(k_{pb}, H(m))$ est le chiffrement du *hachage du message m, en utilisant la clé privée* k_{pb}.

3: Si s = Ds_m aller à 4 sinon aller à 5,
4: **return** la signature est authentique,
5: **return** la signature n'est pas authentique.

Algo 2: Algorithme de vérification de la signature

1.4 Conclusion

Nous venons d'aborder la cryptographie qui assure les objectifs de la sécurité informatique, décrit les différents types de chiffrement : symétrique et asymétrique. Nous avons également parlé du hachage et signature numérique.

Dans ce qui suit nous abordons l'algorithme de cryptage asymétrique RSA sur lequel repose notre travail.

Chapitre 2 : RSA

2.1 Introduction

Le cryptage **RSA**, du nom de ses concepteurs, Ron **R**ivest, Adi **S**hamir et Leonard **A**dleman, est le premier algorithme de chiffrement asymétrique. Il a été découvert en 1977 et il a été breveté par le **M**assachusetts **I**nstitute of **T**echnology (MIT) en 1983 aux États-Unis.

Le **RSA** est un chiffrement asymétrique, c'est-à-dire que l'algorithme de chiffrement n'est pas le même que celui de déchiffrement, et les clés utilisées sont différentes. L'intérêt est énorme: il n'y a plus besoin de transmettre la clé à son destinataire, il suffit de publier librement les clés de cryptage. N'importe qui peut alors crypter un message, mais seul son destinataire, qui possède la clé de décodage, pourra le lire. En quelques années, RSA s'est imposé pour le cryptage comme pour l'authentification et a progressivement supplanté son concurrent, le DES.

Le **RSA** est basé sur la théorie des nombres premiers, et sa robustesse tient du fait qu'il n'existe aucun algorithme de décomposition d'un nombre en facteurs premiers. Alors qu'il est facile de multiplier deux nombres premiers, il est très difficile de retrouver ces deux entiers si l'on en connaît le produit.

2.2 Principe de fonctionnement

RSA est un chiffrement asymétrique qui est basé sur l'utilisation d'une paire de clés : publique pour le chiffrement et privée pour le déchiffrement. Il suit le processus suivant: la génération des clés, le chiffrement et le déchiffrement, la signature et la vérification de la signature. [1]

2.2.1 La génération des clés

On choisit donc deux nombres premiers très grand **p** et **q** qui serviront à former les clés publiques et privées. La taille de **p** et **q** est 1024 bits. On calcul **N** (2048 bits de taille), qui est un constituant de la clé publique et de la clé privée en faisant **: N=p*q**. Ensuite on calcul **e**, qui fait partie de la clé publique, avec : **φ(N)= (p-1)*(q-1)** et l'exposant public **e** de telle sorte qu'il est un nombre premier avec **(p-1)*(q-1)**.C.à.d. Le **PGCD(e,φ(N))=1.**

Ainsi le couple **(N, e)** forme la clé publique du chiffrement.

2.2.2 Le chiffrement

Pour crypter un texte, on va se servir de la clé publique de chiffrement (N, e). Tout d'abord, il faut transformer le texte en nombres (en utilisant la valeur ASCII de chaque lettre ou en remplaçant chaque lettre par son rang dans l'alphabet par exemple).

On a un message codé nommé ***M***. Il faut ensuite découper le message en blocs strictement inférieurs à N. On nomme alors chacun des blocs codés M_i et afin de crypter, on fait :

$$\mathrm{C_i} = M_i^e \bmod N$$

Autrement dit, C_i (étant le bloc de texte crypté), équivaut au reste de la division de M_i^e par ***N***.

On rassemble les parties (C_i) pour former la chaine cryptée **C**.

2.2.3 Le déchiffrement

Pour décrypter un texte, On redécoupe le message crypté C en formant des nombres inferieurs à **N**.

Il s'agit désormais de calculer, à partir de **p** et **q**, la clé privée **d** qui satisfait l'équation

$$d.e \bmod \varphi(N) = 1$$

On le décrypte avec la clé privée **(d, N)** :

$$\mathrm{M_i} = C_i^d \bmod N$$

Autrement dit, M_i (étant le bloc de texte décrypté), et qui vaut au reste de la division de C_i^d par **N**.

On retrouve alors nos blocs, il ne reste plus que les rassembler et transférer vers leur équivalant dans l'alphabet.

2.2.4 La signature

La signature RSA est simplement le fait de chiffrer le message haché que nous notons h (H(M)=h) en utilisant la clé privée.

Le signataire envoie donc le message **M** et sa signature **S.**

2.2.5 La vérification de la signature

A la réception, le récepteur calcule le haché du message **M(H(M)=*h*)** en utilisant la même fonction de hachage que le signataire. Puis il calcule $S^E mod\ N$ avec **(N, e)** la clé publique du signataire. A la fin, il compare ce résultat obtenu à ***h***.

Entrée : taille de la clé //exprimé en bits. **Sortie** : la clé publique (N,e) et la clé privée (N,d)
1. Prendre deux nombres premiers **p** et **q** suffisamment grands (de taille à peu près égale). 2. Calculer N = p.q, 3. Calculer α=(p-1)(q-1), 4. Choisir un nombre E tel que $1 < E < \alpha$ et le PGCD $(e, \alpha)=1$, 3. Prendre un nombre E qui n'a aucun facteur en commun avec α, 4. Calculer D tel que DE mod α =1.
5. **Return** N, e, d

Algo 3: Génération de clés avec RSA

Entrée : (N,E) et M // la clé publique et le texte en clair M avec M∈[0.N-1]. **Sortie :** C // texte chiffré.
1. Calculer $\mathbf{C=M^e\ mod\ N}$
2. **Return** C

Algo 4: Chiffrement avec RSA

Entrée : (D,E) et C // la clé privée et le texte chiffré. **Sortie :** M // texte clair.
1. Calculer $\mathbf{M=C^d\ mod\ N}$
2.**Return** M

Algo 5: Déchiffrement avec RSA

Entrée : (N,D) et M // la clé privée et le texte en clair M avec M∈[0.N-1]. **Sortie :** S,M // la signature et le texte clair.
1.Calculer h=H(M) // le hachage du message M 2.Calculer $S=h^d \bmod N$
3.**Return** S,M

Algo 6 : Signature avec RSA

Entrée : (E, N), M,S **Sortie :** Acceptation ou rejet de la signature
1. Calculer h=H(M) 2. Calculer $h'=S^e \bmod N$
3. Accepter si *h=h'*, rejet sinon

Algo 7: Vérification de la signature avec RSA

2.3 Exemple simple

- On prend un alphabet simple de trois lettres :

A – B – C

- On code cet alphabet de façon que :

A = 1

B = 2

C = 3

- On choisit les deux nombres premiers p et q :

p = 2

q = 5

- On calcule la clé publique n avec :

n = p*q = 2*5 = 10

- On définit la clé publique e de manière que e soit premier avec **(p–1)*(q–1).**

On a alors : **(p–1)*(q–1) = (2–1)*(5–1) = 1*4 = 4**

Le **pgcd (e, 4) = 1** donc **e = 3**

- On calcul d avec **e*d – k((p–1)(q–1)) =1** par exemple :

e*d – k((p–1)(q–1)) =1

3*d – k(4) =1

3*3 – 2*4 =1 car 10 = 3 * 3 + 1

Donc la clé privée **d = 3**

Clé publique : **(N, e) = (10, 3)**

Clé privée : **(N, d) = (10, 3)**

(Dans la pratique, la clé publique n'est jamais égale à la clé privée)

2.3.1 Cryptage du message

- On a : $C_i = M_i^e \bmod N$

Message : BAC

Message Codé : 2 1 3

Crypter A : M_2 = A = 1, donc C_2 =1 mod 10 = 1

Crypter B : M_1 = B = 2, donc C_1 =8 mod 10 = 8

Crypter C : M_3 = C = 3, donc C_3 =27 mod 10 = 7

Message Crypté: 8 1 7

2.3.2 Décryptage du message

- On a: $M_i = C_i^d \bmod N$

Message Crypté : 8 1 7

Décrypter 8 : C_2 = 8, donc M_2 =512 mod 10 = 2

Décrypter 1 : C_1 = 1, donc M_1 =1 mod 10 = 1

Décrypter 7 : C_3 = 7, donc M_3 =343mod 10 = 3

Message Codé retrouvé: 2 1 3

Message décodé: BAC

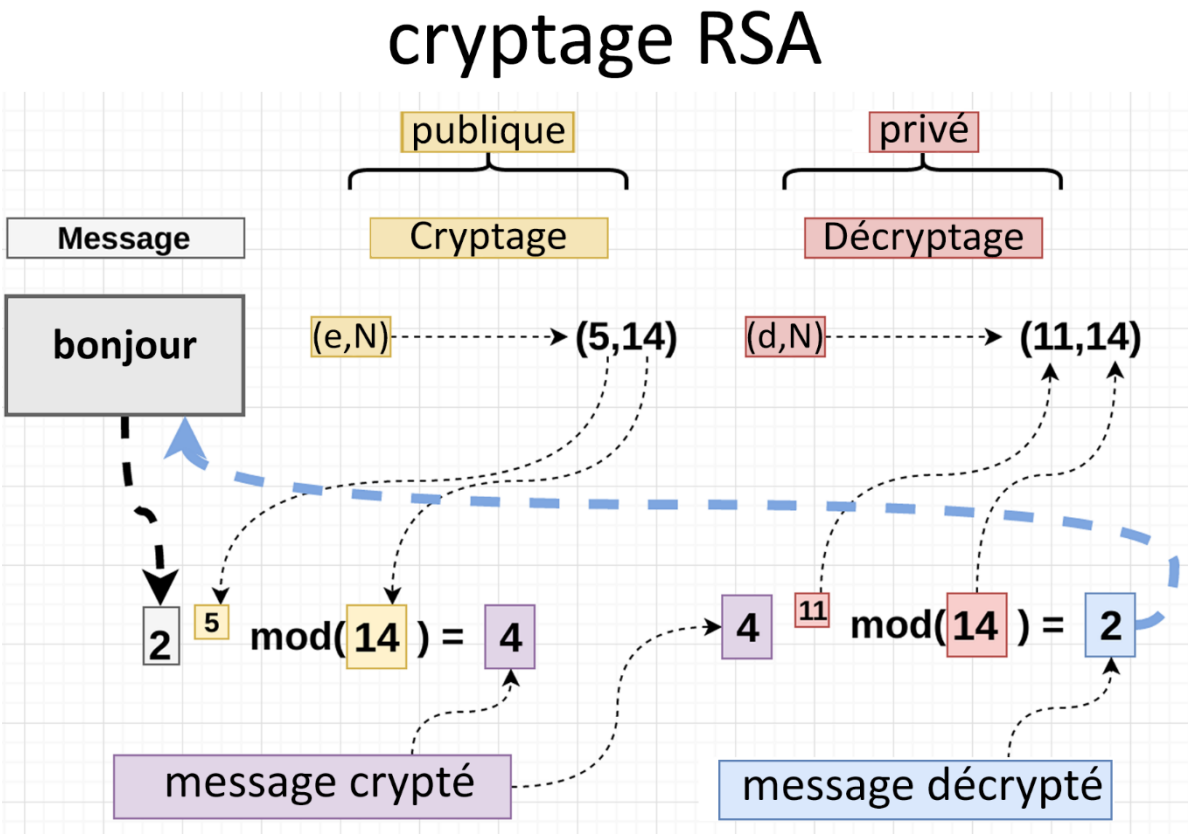


Figure 8 : Cryptage et décryptage RSA

2.4 Les attaques sur RSA

Depuis l'invention du système RSA, la cryptanalyse de ce système n'a pas révélé de failles majeures. Mais si aucune des attaques menées contre RSA n'est réellement pas dévastatrice, elles démontrent toutefois qu'il faut implémenter RSA avec beaucoup de précautions. Dans cette partie nous allons décrire les principales attaques connues, elles ne mettent pas en cause la sécurité de RSA mais illustrent les dangers d'une utilisation incorrecte de RSA

2.4.1 Le problème mathématique

La génération des clés RSA s'appuie sur l'utilisation d'un grand nombre $\boldsymbol{n}$ produit de deux grands nombres premiers $\boldsymbol{p}$et : $\boldsymbol{n}=\boldsymbol{p}*\boldsymbol{q}$ et $\boldsymbol{e}$ premier avec$(\boldsymbol{p}-\boldsymbol{1})*(\boldsymbol{q}-\boldsymbol{1})$, $(\boldsymbol{n}, \boldsymbol{e})$ forment la clé publique, et $\boldsymbol{d}$ l'inverse de $\boldsymbol{e}$ modulo $(\boldsymbol{p}-\boldsymbol{1})*(\boldsymbol{q}-\boldsymbol{1})$**,** la clé privée.

Parmi les attaques mathématiques les plus connues :

a. Factorisation de modulo N:

Cette méthode consiste à décomposer **N** en facteurs premiers afin de retrouver les coefficients **p** et **q**. Bien que cette opération soit compliquée, cependant il existe des algorithmes qui permettent de l'accélérer. Le plus connu est le crible algébrique. L'algorithme du crible du corps de nombres généralisé a été exploité avec succès pour casser **RSA512** bits en une année sur un ordinateur qui a une capacité 10 GIPS [2], et **RSA 768** en décembre 2009 en utilisant un réseau de calcul distribué. De ce fait, les scientifiques ont prédit que **RSA 1024** bits sera cassé en 2034 et le **RSA 2048** bits en 2079. Cette perdition est basée sur la taille de **N** et sur l'évolution des ordinateurs selon la loi empirique de Moore. D'autre part, une équipe de recherche vient de découvrir que la clé 2048 bits peut être factorisée en quelques dizaines de minutes en exploitant l'IA. Cette nouvelle technique de cryptanalyse RSA ne fonctionne que sur des nombres de 2048 bits au maximum mais aucune publication officielle n'a vu le jour pour le moment.

NOMBRE DE MIPS.AN DISPONIBLES	TAILLE MAXIMALE DES CLÉS RSA «FACTORISABLES»
0,048	251
49,6	385
500	438
10000	512
50796	555
512000	620
10^8	784
10^{11}	1035
10^{16}	1551
10^{20}	2057

Figure 9: Nombres de MIPS disponible par taille de clé

Dans [5], les auteurs proposent une nouvelle manière de retrouver les coefficients p et q en exploitant les différentes formes possibles de **p+q** avec **N=pq**.

b. Attaque de Wiener :

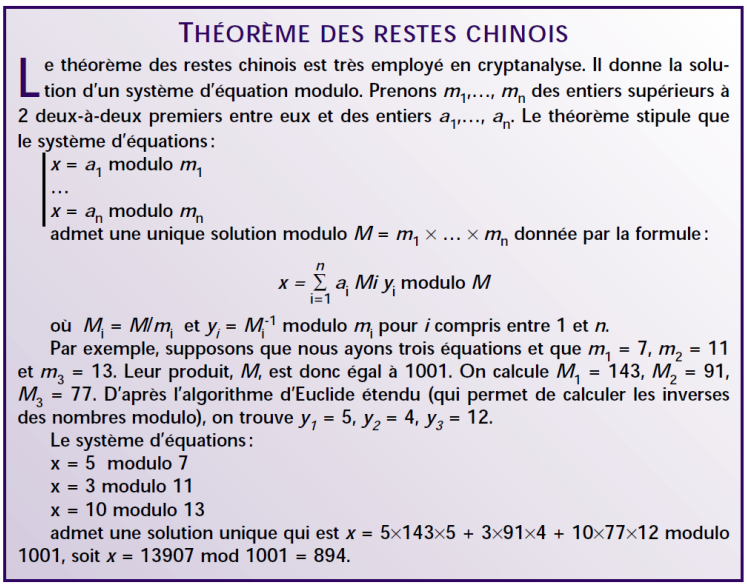
Proposée en 1989 par le cryptologue Michael J. Wiener [4]. Cette attaque permet de retrouver facilement la clé privée à partir de la clé publique (E, N) et pour cela deux conditions sont nécessaires pour l'exploiter : l'exposant D doit être faible et les deux coefficients $\boldsymbol{p}$, $\boldsymbol{q}$ doivent être très proches.

Théorème*: Si les conditions suivantes sont remplies :* $\boldsymbol{d < \frac{1}{3} N^{\frac{1}{4}}}$ *et* $\boldsymbol{q < p < 2q}$ *alors, il est facile de retrouver* ***d****.*

Cette attaque a été améliorée par Dan Boneh et Glen Durfee [4] pour tous les exposants $\boldsymbol{D}$ inférieurs ou égaux à $\boldsymbol{N^{0,292}}$.

c. Attaque d'Hastad :

Proposée par Hastad en 1985 [3], appelée également attaque par broadcast. Cette attaque se base cette fois ci sur un exposant de chiffrement E faible et la connaissance de plusieurs chiffré avec un exposant E identique et des modulos différents. L'attaque *d'Hastad* utilise le Théorème *des restes chinois.*



THÉORÈME DES RESTES CHINOIS

Le théorème des restes chinois est très employé en cryptanalyse. Il donne la solution d'un système d'équation modulo. Prenons $m_1,\ldots, m_n$ des entiers supérieurs à 2 deux-à-deux premiers entre eux et des entiers $a_1,\ldots, a_n$. Le théorème stipule que le système d'équations :

$x = a_1$ modulo m_1
…
$x = a_n$ modulo m_n

admet une unique solution modulo $M = m_1 \times \ldots \times m_n$ donnée par la formule :

$$x = \sum_{i=1}^{n} a_i Mi\, y_i \text{ modulo } M$$

où $M_i = M/m_i$ et $y_i = M_i^{-1}$ modulo m_i pour i compris entre 1 et n.

Par exemple, supposons que nous ayons trois équations et que $m_1 = 7$, $m_2 = 11$ et $m_3 = 13$. Leur produit, M, est donc égal à 1001. On calcule $M_1 = 143$, $M_2 = 91$, $M_3 = 77$. D'après l'algorithme d'Euclide étendu (qui permet de calculer les inverses des nombres modulo), on trouve $y_1 = 5$, $y_2 = 4$, $y_3 = 12$.

Le système d'équations :

x = 5 modulo 7
x = 3 modulo 11
x = 10 modulo 13

admet une solution unique qui est $x = 5\times143\times5 + 3\times91\times4 + 10\times77\times12$ modulo 1001, soit $x = 13907 \bmod 1001 = 894$.

Figure 10 Théorème des restes chinois

Théorème : *Si les conditions suivantes sont remplies : Si on connait les chiffrés de* $\boldsymbol{X}$ *noté* $\boldsymbol{C_0, C_1, \ldots, C_{K-1}}$ *avec* $\boldsymbol{C_i = X^E mod\ N_i}$, $\boldsymbol{i \in [0.k\text{-}1]}$ *et* $\boldsymbol{k \geq E}$ *alors, il est facile de retrouver le message clair X.*

2.4.2 Les attaques de protocole

Même si la fonction RSA est solide, la façon dont on l'utilise pour obtenir un cryptosystème capable d'effectuer du chiffrement ou de la signature n'est pas neutre. De manière générale, il doit non seulement être impossible en pratique de retrouver un message clair à partir de son chiffré (ou de forger une fausse signature), hormis pour l'utilisateur légitime du système bien entendu, mais il doit être également impossible de déceler la moindre information sur le message clair (ou sur la signature). C'est ce que l'on nomme la sécurité

sémantique. Or, la fonction RSA ne vérifie pas cette propriété, car elle possède une propriété de multiplicativité : $f(x \times y) = f(x) \times f(y)$. Cette faiblesse ouvre la voie à certaines attaques, que ce soit en mode de chiffrement ou en mode signature. Illustrons comment cette propriété peut fournir une attaque pour le chiffrement. Supposons que Bernard veuille envoyer le même message M, sous forme chiffrée, à trois interlocuteurs différents, dont les clés publiques RSA sont respectivement (n1, e=3), (n2, e=3) et (n3, e=3). S'il utilise la fonction RSA de façon élémentaire, Bernard envoie donc les valeurs C1 = M3 modulo n1, C2 = M3 modulo n2 et C3 = M3 modulo n3. Si Caroline, une espionne patentée, intercepte les trois chiffrés C1, C2 et C3, elle peut retrouver aisément le message clair M. D'abord, à l'aide du théorème des restes chinois, Caroline calcule la valeur C'telle que $0 \leq C' < n1n2n3$, et $C' \equiv M3$ modulo n1n2n3. Or, par hypothèse, les messages ont une longueur inférieure à la première partie de la clé publique, c'est-à-dire que l'on a $M < n1$, $M < n2$ et $M < n3$. Donc $M3 < n1n2n3$ et par suite $C' = M3$, ou encore $M = (C')1/3$. Caroline a donc retrouvé M en faisant une simple racine cubique! Cela montre qu'il est fort risqué de chiffrer le même message M par plusieurs clés publiques RSA distinctes. Dans une autre direction, D. Coppersmith et ses collègues ont montré en 1996, qu'il est également risqué de chiffrer plusieurs messages « liés » au moyen de la même clé publique RSA. Plus précisément, si Bernard envoie les chiffrés C1 et C2 de deux messages M1 et M2 liés par une relation de dépendance polynômiale M2 = P(M1) modulo n, alors il est facile pour un attaquant de retrouver M1 et M2 à partir de C1 et C2.

S'il peut sembler artificiel d'avoir à chiffrer deux messages ainsi liés, un scénario assez proche peut se produire dans des applications réelles. Supposons que pour chiffrer un message M de longueur (k–m) bits, on lui ajoute, à la fin, m bits aléatoires avant de lui appliquer la fonction RSA, dont le modulo fait k bits. Il arrive que Bernard doive envoyer deux fois le chiffré du même message M (par exemple, s'il y a eu un problème de transmission sur la ligne). Mathématiquement, cela signifie que Bernard envoie successivement C1 = f(M1) et C2 = f(M2), avec M1 = 2m.M + r1 et M2 = 2m.M + r2, où r1 et r2 sont deux valeurs aléatoires de m bits. La présence de ces m bits aléatoires offre-t-elle une protection suffisante? Non, car Don Coppersmith a montré en 1997 que l'on pouvait retrouver M à partir de C1 et de C2 à condition que m ne soit pas trop grand : plus p récisément l'attaque fonctionne si m est inférieur à k/e^2 (ainsi pour e = 3, l'attaque est possible si la partie aléatoire ajoutée au message a une longueur inférieure à 1/9 de celle du modulo).

Toutes ces attaques montrent qu'il faut faire très attention à la manière dont on applique la fonction RSA pour chiffrer un message. Dans la pratique, on commence par «formater»le

message M au moyen d'une transformation ϕ qui fait intervenir un nombre aléatoire. Ainsi le message chiffré est obtenu par $C = f(\phi(M,r))$, où r est une valeur aléatoire. Pour déchiffrer, on calcule $\phi(M,r) = f^{-1}(C)$, et ϕ est conçue pour qu'il soit facile de retrouver M à partir de $\phi(M,r)$. La théorie des protocoles de chiffrement RSA est aujourd'hui bien maîtrisée, au point qu'on connaît maintenant des transformations ϕ qui sont prouvées «saines». Cela signifie que, pour ces transformations ϕ, la sécurité sémantique du schéma global de chiffrement $C = f(\phi(M,r))$ est équivalente à la solidité de la fonction RSA de base. En d'autres termes, si la fonction f est mathématiquement solide et qu'on utilise une «bonne» transformation ϕ, alors il n'existe pas d'attaque mathématique plus simple que celle consistant à trouver un moyen d'inverser la fonction f .

2.4.3 Les attaques physiques

Comme nous l'avons vu, si l'on suppose que la fonction RSA est mathématiquement solide, on peut construire des protocoles sûrs, aussi bien pour le chiffrement que pour la signature. Ces protocoles admettent des preuves de sécurité. Néanmoins, dans les applications réelles, malgré ces preuves, certaines attaques restent possibles. Celles-ci ne s'attaquent pas au problème mathématique, mais elles utilisent le fait que le dispositif électronique qui fait les calculs de chiffrement ou de signature, n'est pas une abstraction mathématique : dans le monde physique, il met un certain temps à calculer, il consomme du courant électrique, il peut faire des erreurs de calcul, etc. Avec ces paramètres supplémentaires, l'attaquant parvient parfois à ses fins.

On retrouve principalement trois types d'attaques fondées sur ces données : les attaques sur le temps de calcul, les attaques sur la consommation électrique et les attaques par injection de fautes. Notons que toutes ces attaques « physiques » concernent essentiellement les cartes à puce pour lesquelles, avec un dispositif adéquat, il est facile d'obtenir des données.

a. Attaques sur le temps de calcul:

L'algorithme de calcul de la fonction RSA est toujours le même. En particulier il fait intervenir une boucle pour le calcul de $\boldsymbol{y^d \bmod n}$ (ou $\boldsymbol{d}$ est la clé secrète). $\boldsymbol{d = d_1 d_1 d_3 \ldots d_n}$, chaque bit $\boldsymbol{d_i}$ est égal à **0** ou **1**.Or dans la boucle de calcul, on trouve une instruction du type '*si* $\boldsymbol{d_i = 1}$ *alors faire tel calcul sinon ne pas le faire*'.

En mesurant les temps de calcul pour de nombreuses valeurs initiales de y (message crypté), on peut déduire si le calcul qui est fait lorsque $\boldsymbol{d_i = 1}$ a été réellement effectué et de retrouver

la clé secrète **d** bit par bit. On peut contourner ce type d'attaque en masquant les différences de temps de calcul.

b. Attaque sur la consommation électrique:

Le même type d'attaque peut être effectué avec la consommation électrique pour chacun des calculs, on mesure la courbe de consommation électrique du composant qui fait le calcul. En analysant la statistique de la consommation électrique à chaque étape du calcul, on déduit de proche en proche la valeur de la clé secrète ***d***.

Des méthodes existent pour fausser la consommation électrique et rendre cette information inutilisable.

c. Attaques par injection de fautes:

Cette attaque a été conçue en **1996** et présentée comme un nouveau modèle d'attaque physique sur les cartes à puce : « Cryptanalyse en présence d'erreurs de calcul dans les processeurs », Elle se base sur l'utilisation d'une signature erronée puis de la signature correcte.

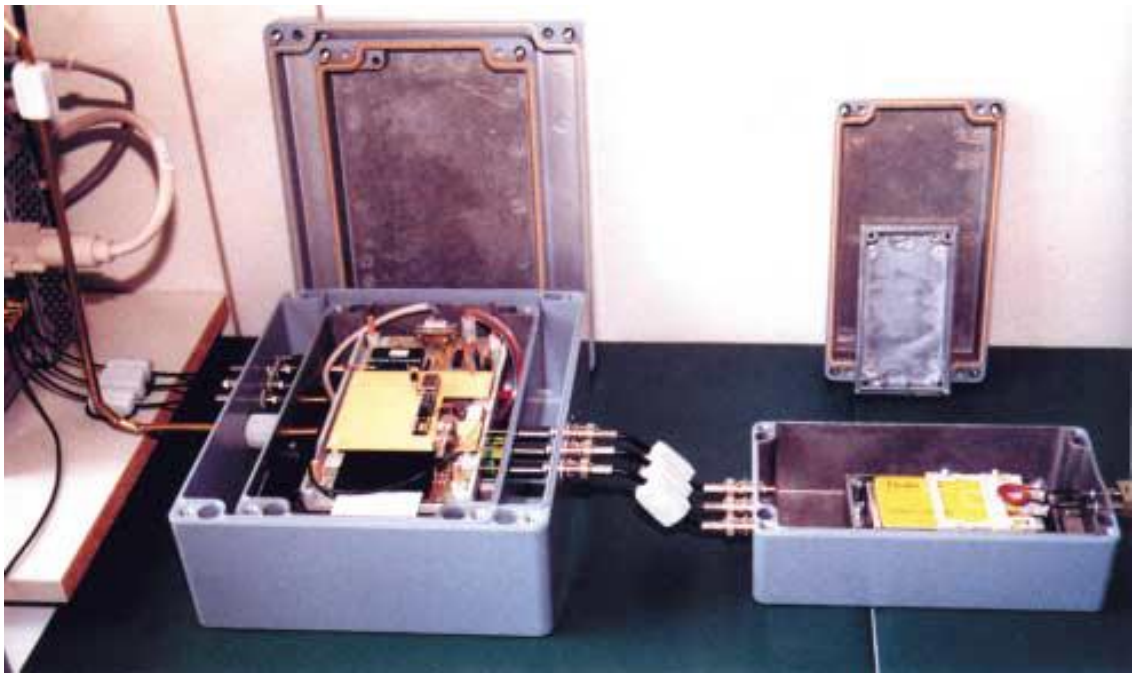


Figure 11: Dispositif modeste d'analyse

2.4 conclusion

Dans ce chapitre, nous nous sommes intéressés à l'algorithme RSA, nous avons expliqué son principe de fonctionnement, ainsi que sa cryptanalyse. Il est très fiable mais RSA dans sa forme brute ne peut être utilisé sur les systèmes embarqué a ressources limitées.

Dans la littérature plusieurs solutions ont été proposé, parmi ces solutions celle décrite dans [6]. Notre contribution consiste à l'adapter pour des systèmes à faibles ressources tel que les smartes phones.

Dans le chapitre qui suit nous allons détailler le RSA distribué que nous allons utiliser dans notre solution.

Partie II
Notre Contribution

Chapitre 3 : RSA Distribué

Notre contribution consiste à adapter le RSA distribué proposé dans la thèse [6] pour des systèmes embarqués à faible ressources dans l'environnement Java Android.

Initialement, l'utilisateur *u* ne peut pas chiffrer seul ses données car il est limité en ressources matérielles. Pour y remédier, selon [6] on fait participer un ensemble d'utilisateurs pour crypter une donnée M. Pour cela, on crée d'abord une communauté d'utilisateurs. Pour créer cette communauté, nous pouvons utiliser, par exemple, un réseau social comme Facebook. Chaque membre met ses ressources à la disposition de la communauté. Cette dernière représente une puissance de calcul importante que nous exploitons pour crypter des données par RSA. Lorsqu'un utilisateur *u* de la communauté veut crypter une donnée X, il envoie des fragments du code aux différents Membres. Pour cela, il remplace l'opération de chiffrement RSA $M^E \bmod N$ par un ensemble d'opérations $F_0, F_1, ..., F_{n-1}$

avec $F_i = (M^{2i})^{ai} \bmod N$ et $E = \sum_{i=0}^{i=n-1} a_i 2^i$

Pour accélérer l'exécution de F_i, nous utilisons des calculs intermédiaires et nous utilisons le nouvel algorithme proposé dans [6] que nous détaillons ultérieurement. Chaque membre exécute le fragment reçu et retourne le résultat R_i. L'utilisateur ***u*** collecte les différents résultats $R_0, R_1, ...R_{n-1}$ et génère le résultat final $R = \prod_{i=0}^{i=n-1} R_i \bmod N$ qui représente la donnée chiffrée C.

3.1 RSA Distribué

Le RSA proposée se base sur les points suivants:

a. Décomposer la formule de chiffrement $M^E \bmod N$ en plusieurs opérations partielles qui utilisent des exposants inférieurs à E. A savoir (E, N) qui représente la clé publique de l'algorithme RSA,

b. Proposer un algorithme efficace pour calculer les opérations générées dans a,

c. Utiliser des résultats intermédiaires pour calculer ces opérations partielles

3.1.1 Décomposition de la formule de chiffrement M^E mod N

Etape1: on convertit l'exposant *E* en notation binaire. *E* peut être écrit :

$$E = \sum_{i=0}^{i=n-1} a_i 2^i \qquad (1)$$

Dans cette formule, la longueur de *E* est de *n* bits. a_i peut prendre la valeur 0 ou 1 pour tout *i* tel que $0 \le i < n-1$. Par définition, $a_{n-1} = 1$. La valeur M^E peut alors être écrite :

$$M^E = M^{\sum_{i=0}^{i=n-1} a_i 2^i} \qquad (2)$$

$$M^E = \prod_{i=0}^{i=n-1} M^{a_i 2^i} = \prod_{i=0}^{i=n-1} (M^{2^i})^{a_i} \qquad (3)$$

Etape2: M^E mod N peut être réécrite sous la forme suivante:

$$C = \prod_{i=0}^{i=n-1} (M^{2^i})^{a_i} \bmod N \qquad (4)$$

$$C = \prod_{i=0}^{i=n-1} F_i \bmod N \qquad (5)$$

Avec :

$$F_i = (M^{2^i})^{a_i} \bmod N \qquad (6)$$

La formule initiale **$C=M^E$ mod N** est décomposée en plusieurs opérations de forme exponentielle modulo (Formule (6)), où F_i est le Chiffré du message M avec un seul bit de la clé publique E. Le but de cette transformation est de rendre la formule F_i plus simple en calcul avec utilisation de moins de ressources, en termes de mémoire et d'énergie, que la formule initiale.

3.1.2 Algorithme de calcul de l'opération de type exponentielle modulaire

Dans la Formule (6), nous pouvons avoir deux valeurs possibles pour a_i :

Si $a_i = 0$ **alors**

$$F_i = (M^{2^i})^0 \bmod N = 1 \qquad (7)$$

Sinon

Si $a_i = 1$ **alors**

$$F_i = M^{2^i} \bmod N \qquad (8)$$

FinSi

FinSi

La Formule (8) est, par conséquent, la seule formule à calculer. Pour cela, nous proposons un nouvel algorithme ***modular_power()*** qui s'inspire de la méthode de Right-To-Left binary [7]. Cet algorithme permet d'accélérer son exécution.

Algo 8 : modular_power (M,i,N)

begin
result ⟵ M
for 0 ≤ j ≤ i
result ⟵ *(result *result) mod N*
end for
return *result*
end

3.1.3Exploitation des résultats intermédiaires

Soit F_x le chiffré de la donnée M avec le bit x de la clé publique (E, N) et F_y le chiffré de la donnée M avec le bit y de la clé publique (E, N). Nous pouvons écrire :

$$F_x = M^{2^x} \bmod N \qquad (9)$$

et

$$F_y = M^{2^y} \bmod N$$

Avec $y=x+a$.

Nous pouvons réécrire F_y de cette façon :

$$F_y = M^{2^{(x+a)}} \bmod N = M^{2x} * M^{2^a} \bmod N$$

F_y se réécrit alors :

$$F_y = F_x * M^{2^a} \bmod N \qquad (10)$$

Finalement, pour calculer la formule F_y , il suffit de calculer $M^{2^a} \bmod N$ avec ($a=y-x$) en utilisant l'algorithme (Algo 8). Et enfin, le multiplier par le résultat de la formule F_x. Au final, nous avons remplacé le chiffrement RSA standard que nous nommons cryptage non distribué par un algorithme parallèle qui est composé de plusieurs tâches séquentielles F_i de la formule (8) et qui peuvent s'exécuter simultanément. Chaque formule F_i sera exécutée par un membre de la communauté en utilisant l'Algorithme (Algo 8). Pour accélérer l'exécution des opérations $F_0, F_1, ..., F_{n-1}$,chaque résultat de calcul F_i est exploité pour générer le résultat de F_{i+1} (voir Formule (**10**)).

3.2 Schéma de la solution proposée

Soit $U = \{u_0, u_1, .., u_{l-1}\}$ une communauté d'utilisateurs de taille ***l*** utilisant des équipements à ressources limitées (nœuds capteurs, Smartphones, IoT, ...etc.). Cette communauté est créée via un réseau social tel que Facebook... Lorsqu'un utilisateur u_i veut crypter une donnée M en utilisant son équipement, il fait appel aux ressources de sa

communauté U. Pour cela, quatre phases sont nécessaires. La première phase est la phase de décomposition qui consiste à décomposer la clé de chiffrement E en plusieurs clés partielles $e_0, e_1, e_2, .., e_{n-1}$. La deuxième phase est la phase de transmission : l'utilisateur u_i envoie à chaque Membre de la communauté une partie de sa clé de chiffrement e_i et la donnée à chiffrer M. La troisième phase est la phase de calcul distribué utilisant les différentes parties générées à partir de la première phase. Chaque participant est capable de calculer et de diffuser efficacement le résultat de la formule F_i (voir Formule (8)). La quatrième phase est la phase de récupération des résultats. L'utilisateur u_i peut générer alors la donnée cryptée C à partir des résultats récupérés, la Formule (5) est utilisée pour calculer le message chiffrée C :

$$C = \prod_{i=0}^{i=n-1} F_i \bmod N$$

La figure suivante (Figure(12)) présente le schéma de l'architecture que nous proposons.

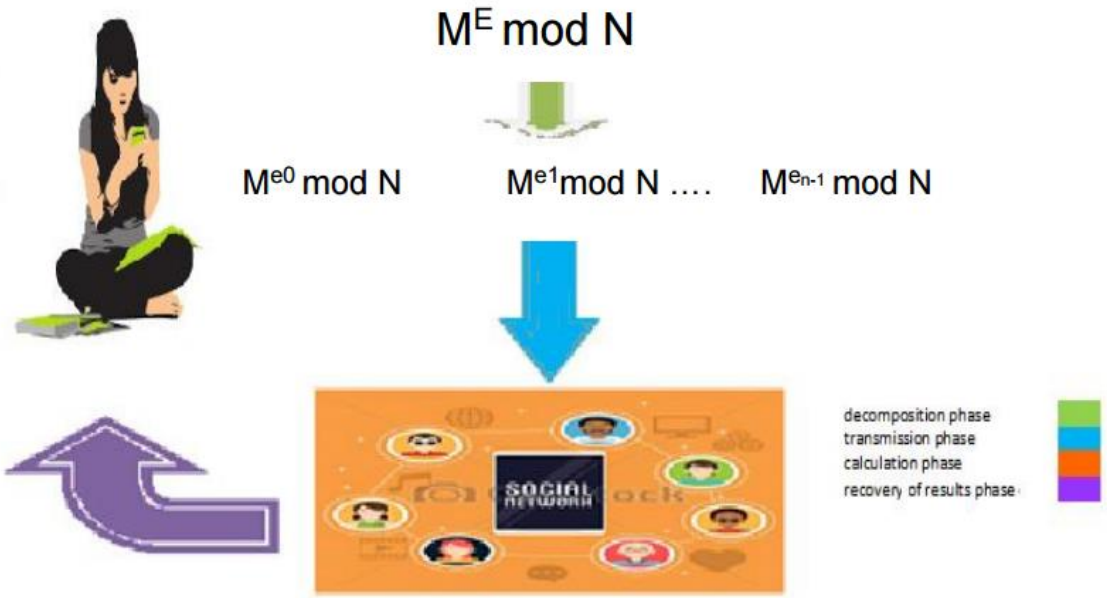


Figure 12 : Cryptage réparti sur une communauté de réseau social

3.3 Les protocoles utilisés

La solution proposée se base sur trois protocoles de bases: login, consultation et chiffrement distribué.

3.3.1 Protocole login

Dans les réseaux sociaux tel que Facebook, chaque utilisateur est identifié d'une manière unique en utilisant par exemple son adresse E-mail. Ce protocole permet à l'utilisateur d'accéder à son espace et d'annoncer aux autres utilisateurs qu'il est en ligne et qu'il met ses ressources à la disposition de sa communauté.

3.3.2 Protocole de consultation

Lorsqu'un utilisateur connecté ui veut chiffrer une donnée M, il consulte la liste de ses amis (les membres de son groupe) connectés afin d'exploiter leurs ressources.

3.3.3 Protocole de chiffrement distribué

L'utilisateur envoie aux membres de sa communauté, la donnée à chiffrer M et les clés partielles générées à partir de la clé de chiffrement E en utilisant le principe décrit dans la section **3.1**

Exemple : Soit l'utilisateur u_0 qui veut crypter la donnée M avec la clé de chiffrement de RSA (E, N) avec E = 289, et N = 40301. Comme E= (100100001) en base 2, trois membres sont nécessaires car E comprend trois 1. Les informations envoyées aux Membres et les traitements effectués par ces derniers sont illustrés dans le table (1)

Les Utilisateurs	Les données envoyées aux Utilisateurs	Les traitements effectués par les Utilisateurs (en utilisant l'Algorithme (1))
u_1	M ; 0	$R_0 = M^{2^0} \bmod N$
u_2	M ; 5	$R1 = M^{2^5} \bmod N$
u_3	M ; 3	$R_2 = (M^{2^3} \bmod N) * R1$

Table1 : le cryptage RSA distribué

Chaque Utilisateur de la communauté est capable d'effectuer les traitements suivants:

u_1 : calcule la formule $M^{2^0} \bmod N$ en utilisant l'Algorithme (Algo 8), u_1 génère le résultat R_0. Ce résultat est envoyé à l'utilisateur u_0.

u_2 : calcule la formule $M^{2^5} \bmod N$ en utilisant l'Algorithme (Algo 8) et génère le résultat

R_1. Ce résultat est envoyé à l'utilisateur u0 et à l'utilisateur u3 pour l'utiliser comme calcul intermédiaire.

u_3 : calcule la formule $M^{2^3} \bmod N$ en utilisant l'Algorithme (Algo 8) et génère le résultat R_3. A la réception du résultat R_1, u_3 génère le résultat $R_2=R_3*R_1$. Puis il envoie le résultat R_2 à l'Utilisateur u_0.

A la réception de tous les résultats partiels R_i pour $i= \{0, 1, 2\}$, l'utilisateur u_0 génère le résultat final qui correspond à la donnée cryptée ***C***. Pour calculer ***C***, nous utilisons la Formule (5). Dans l'exemple ci-dessus, $C = R_0 * R_1 * R_2 \; Mod \; N$. Voir Figure (13).

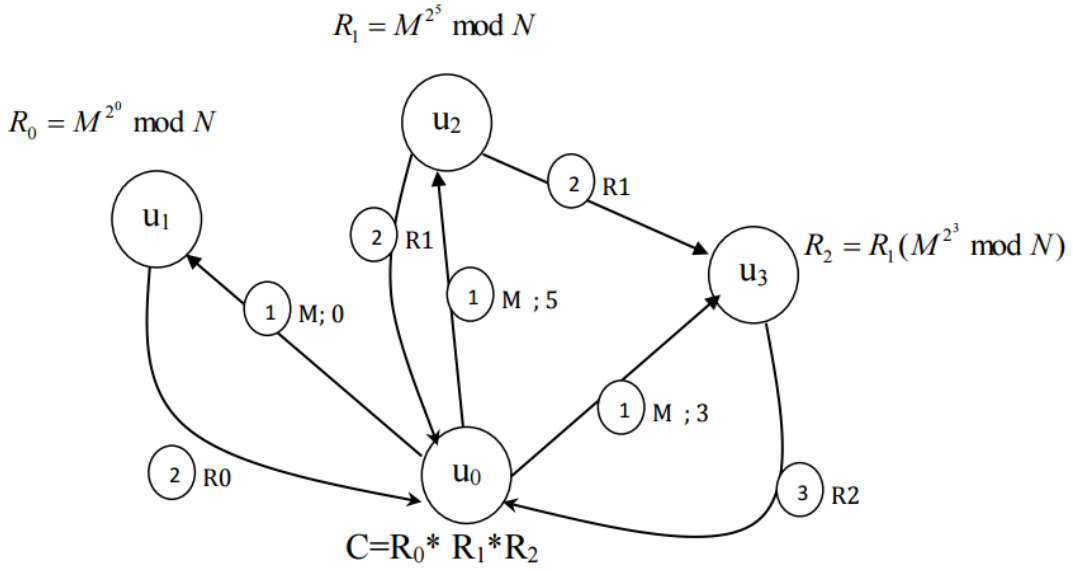


Figure 13 : Cryptage distribué

3.4 Conclusion

Dans ce chapitre, nous avons décrit une nouvelle manière de crypter la donnée M par RSA en faisant participer une communauté d'utilisateurs à ressources limitées. Cette nouvelle approche nommée cryptage distribué consiste à remplacer l'opération de chiffrement M^E mod N par un ensemble d'opérations indépendantes les unes des autres qui peuvent s'exécuter en parallèles et utilisent moins de ressource. Pour valider l'idée, nous utilisons une communauté de smart phone connecté à un réseau sans fil.

Dans ce qui suit, nous nous intéressons à la réalisation de notre projet.

Chapitre 4 : Réalisation

Pour adapter le RSA distribué décrit dans le chapitre précèdent dans la plateforme Java-Android, nous créons une communauté liée en wifi. Ces derniers ont des unités de calcul à faibles ressources qui doivent réaliser un cryptage complexe.

Nous supposons que le réseau formé est un réseau de confiance (pas de partie malveillante). Soit **A** une personne qui veut envoyer un message M d'une importance capitale. Afin d'assurer la confidentialité du message, **A** va appliquer l'algorithme RSA.

Pour parvenir à régler le problème de manque de ressource de son smartphone, **A** sollicitera de l'aide chez ses amis (**B_1, B_2 , …, B_n**). Grace à cela la donnée M va être crypté en RSA avec des clés dépassant les 1024bits.

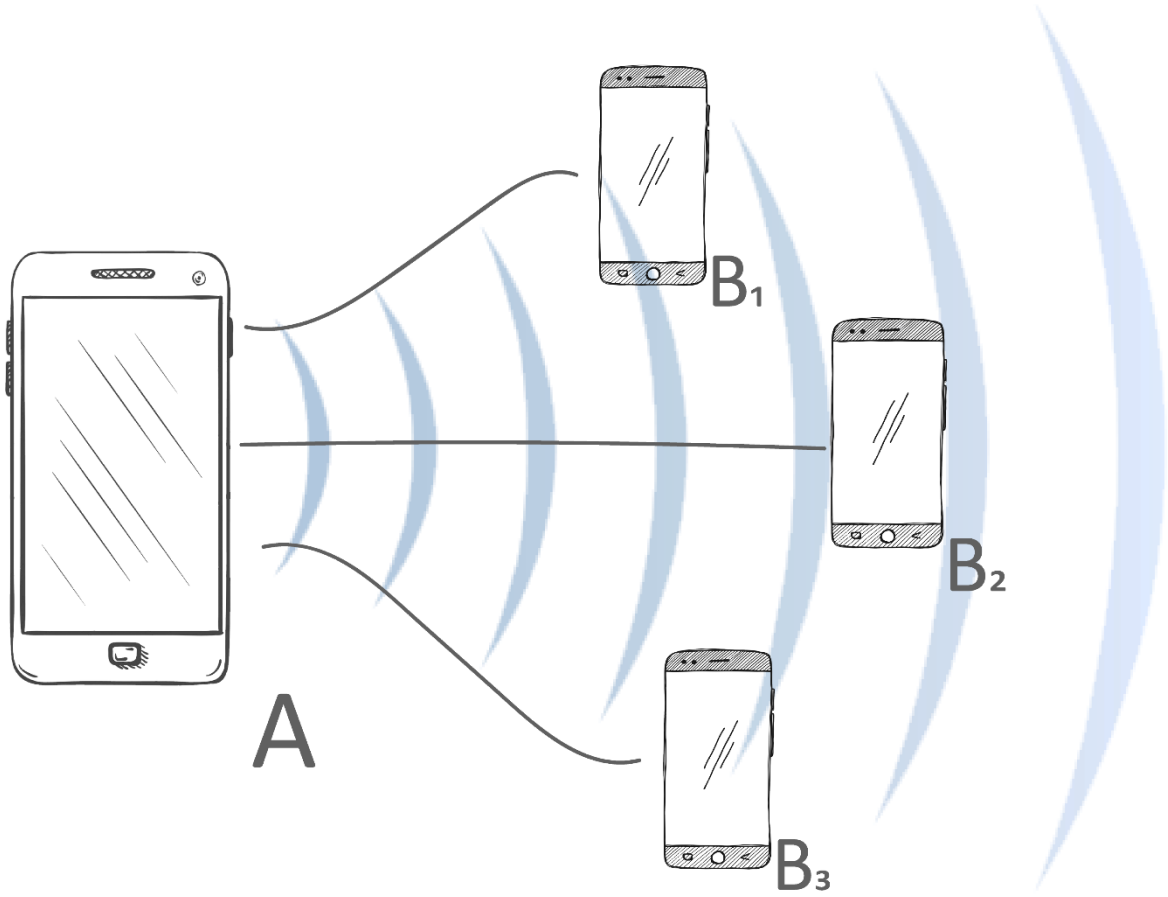


Figure 14 : A,B1,B2 et B3 sont connecté au meme réseau wifi

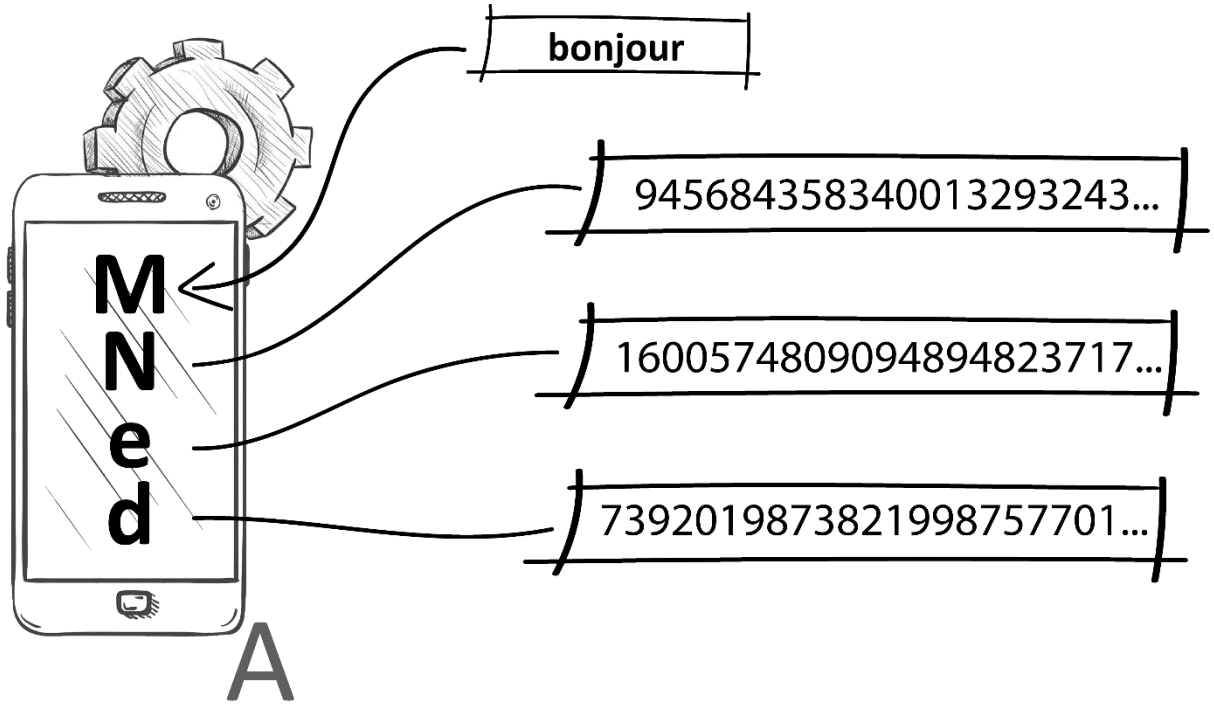


Figure 16 : Phase récupération du message et génération des clés

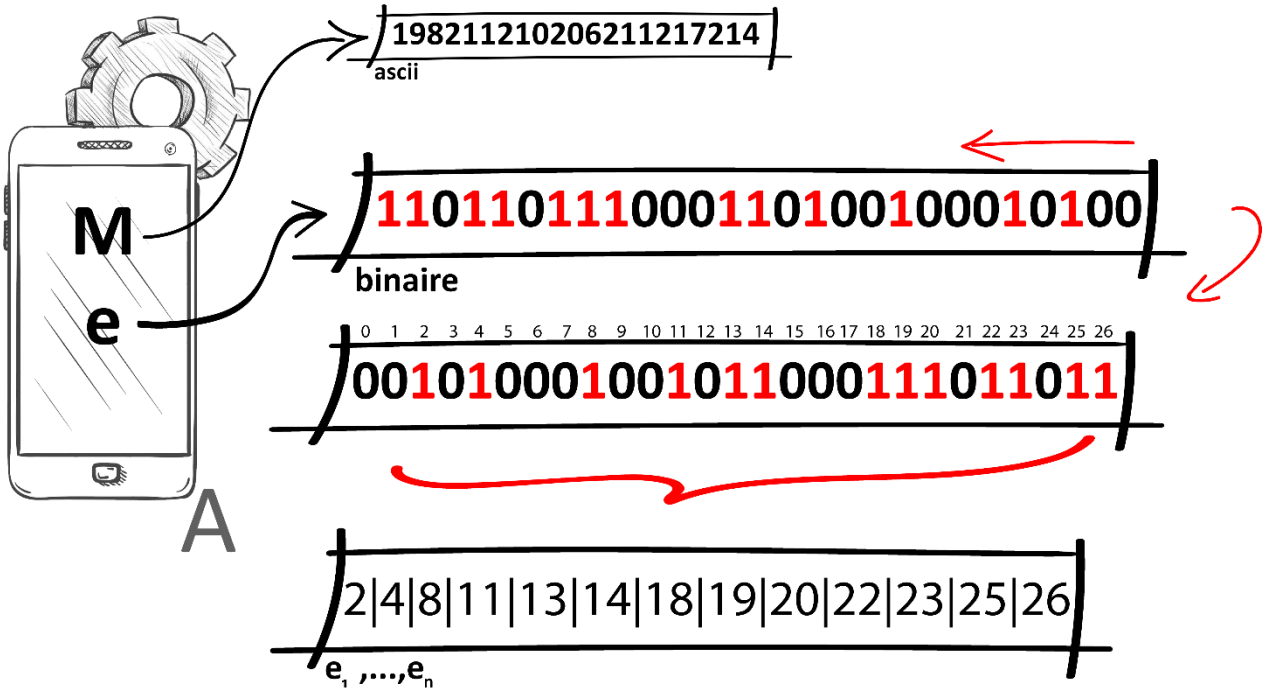


Figure 15 : Phase transformation du message et de la clé publique (e).

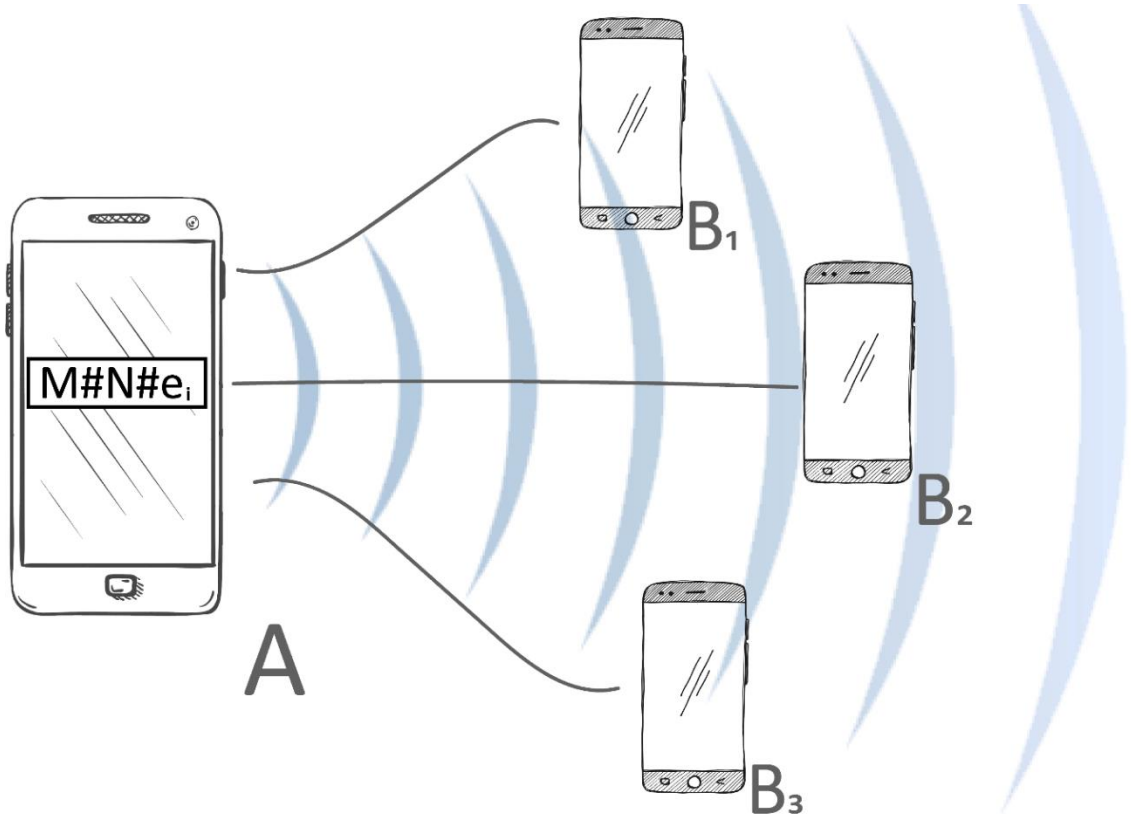


Figure 17 : Phase génération des messages à envoyer.

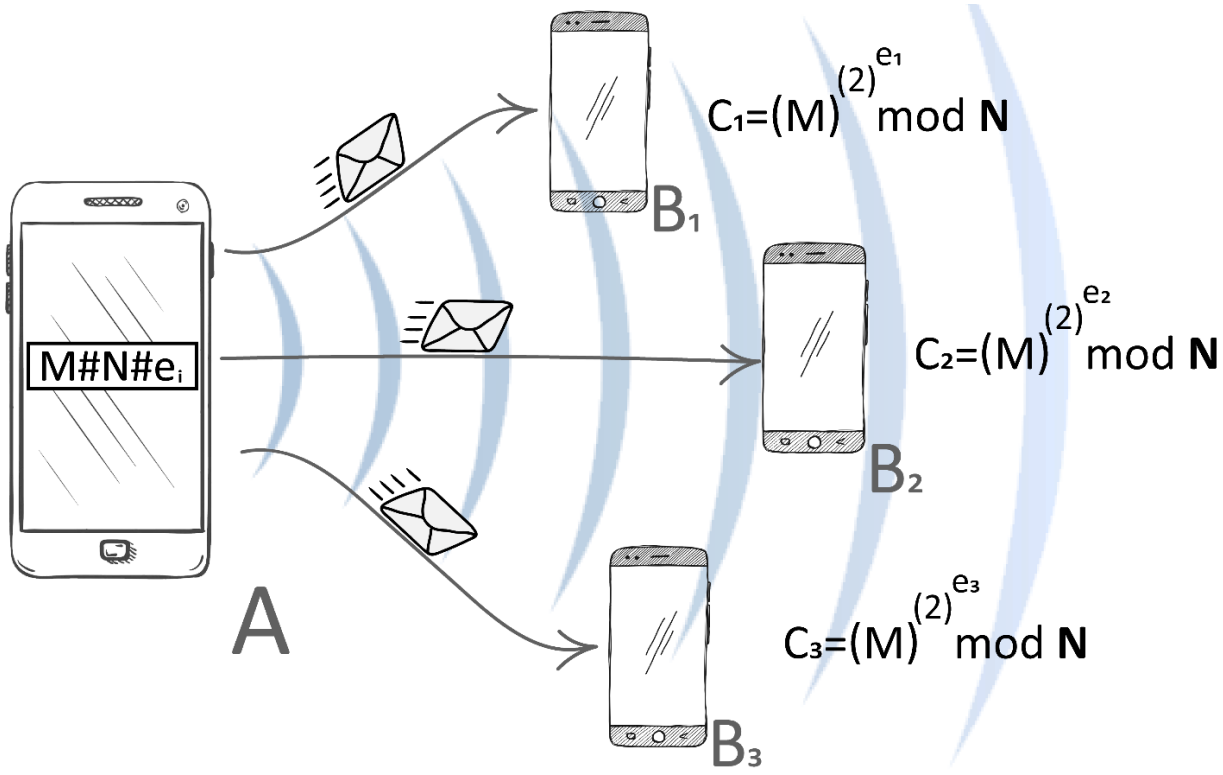


Figure 18 : Phase envoi des messages à B1,B2,B3 pour faire le chiffrement.

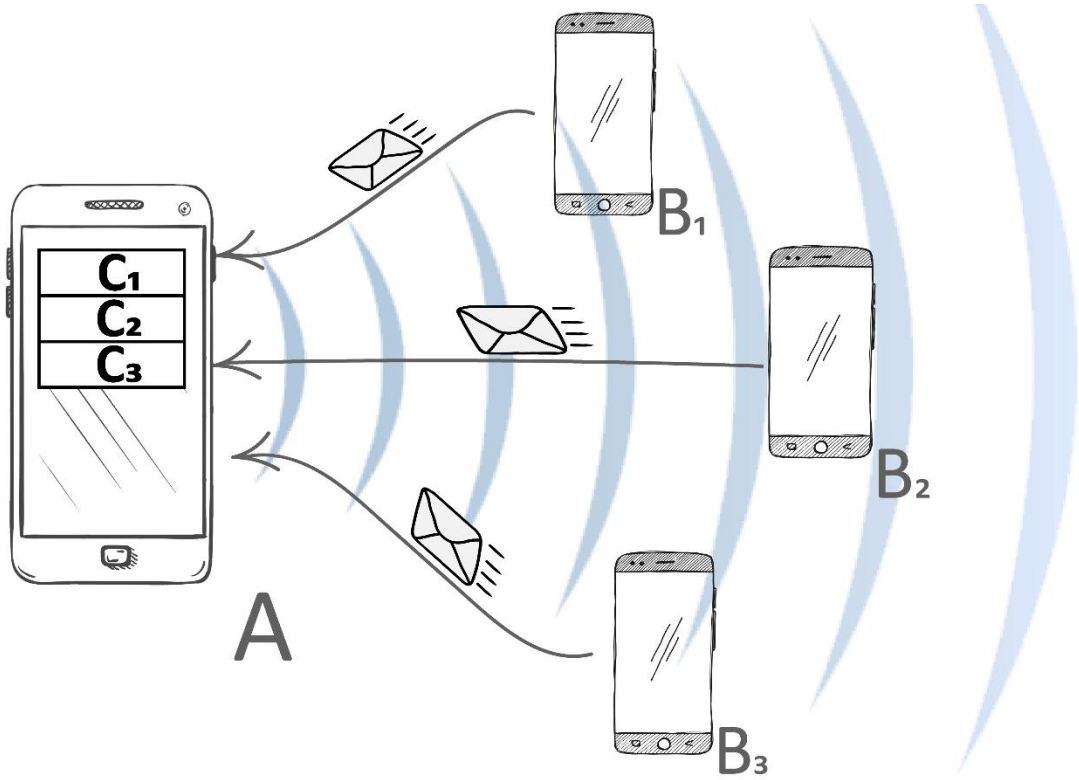


Figure 19 : Phase récupération des résultats

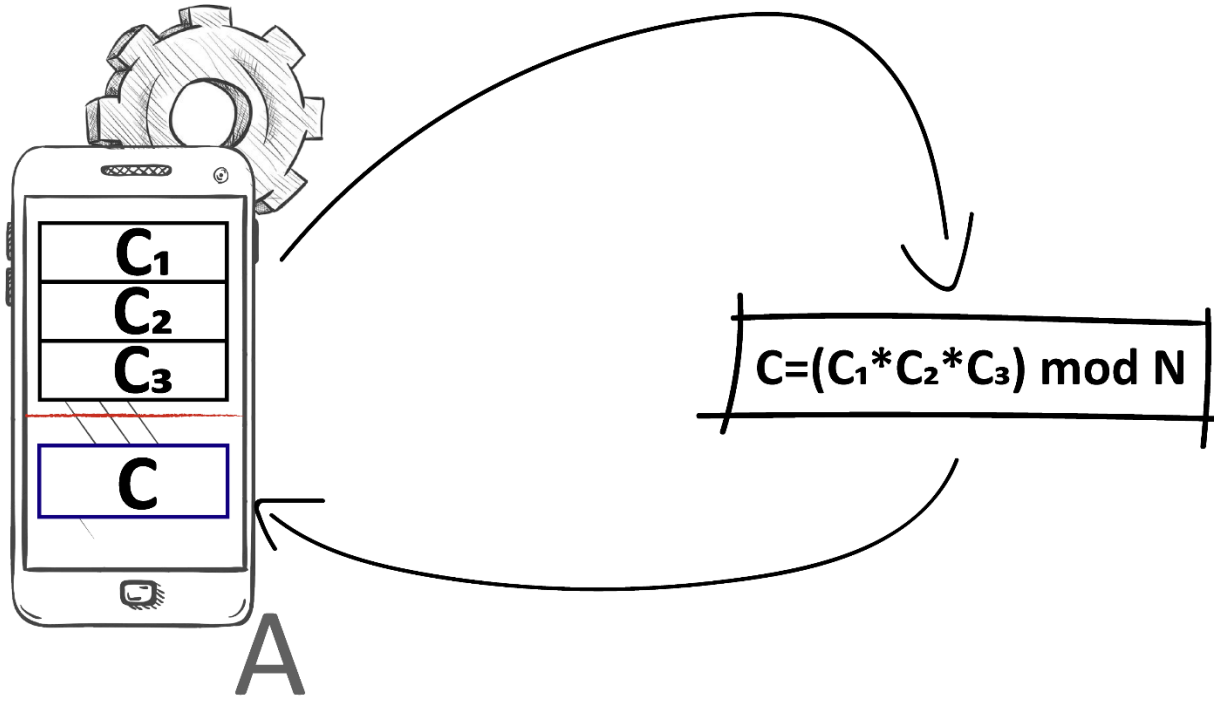


Figure 20 : Phase calcul final en se basant sur les calculs précèdent

4.1 Technologies utilisées

4.1.1 IDE Eclipse

Eclipse est un environnement de développement Java gratuit, open source et extensible. Il est capable d'intégrer des modules (Plugins) de base permettant de gérer l'ensemble de ressources et faciliter le travail du programmeur. **[N3]**

Figure 21 : Logo Eclipse

a. Le Plugin ADT (Android Development Tools)

Pour développer une application android, nous avons à installer le plugin *Android* qui rajoutera à **Eclipse** les fonctionnalités spécialisées dans le développement sous Android.

b. Software Development Kit (SDK)

C'est un kit de développement basé sur le langage Java. Il s'agit des outils que Google a fournis pour interagir avec **Android** pour la réalisation des applications.

4.1.2 le langage de programmation JAVA

Java est un langage de programmation orienté objet ,a été épuré des concepts les plus subtils du C++ et à la fois les plus déroutants, tels que les pointeurs et références, ou l'héritage multiple contourné par l'implémentation des interfaces. Une particularité de Java est que les logiciels écrits dans ce langage sont compilés vers une représentation binaire intermédiaire qui peut être exécutée dans une machine virtuelle Java (JVM) en faisant abstraction du système d'exploitation**.[N4]**

Figure 22 : Logo JAVA

4.1.3 Android OS

Android est le système d'exploitation mobile crée par Google. Il équipe la majorité des téléphones portables du moment (smartphones). Son principal concurrent est Apple avec l'iPhone. Android est un système vous permettant de personnaliser votre téléphone, télécharger des applications (navigateur Internet, GPS, Facebook...). Ce système est complet, car il compte tous les services nécessaires pour le développement d'application complexes. **[N5]**

Figure 23 : Logo Android

Ci-dessous le pourcentage qu'occupe les systèmes d'exploitation les plus connu : **[N6]**

Figure 24 : Les Systèmes d'exploitation dans le monde, septembre 2020

4.2 Conclusion

Dans ce chapitre réalisation nous avons présenté le principe de fonctionnement de notre application mobile.

Nous avons par la suite présenté les technologies utilisées pour implémenter la solution sur des smartphones sous Android.

Conclusion générale

Dans ce Projet, nous avons détaillé le RSA distribué qui est une nouvelle variante de l'algorithme RSA pour des systèmes à ressources limitées (capteurs, Smartphones…). Son principale fonctionnement consiste à remplacer l'opération de chiffrement par un ensemble d'opérations moins coûteuses et indépendantes les unes des autres. Le RSA distribué fait participer une communauté d'utilisateurs de confiance dotée d'équipements à faibles ressources. Dans notre solution nous exploitons cette nouvelle méthode du chiffrement et nous l'adaptons pour des systèmes embarqués à faible ressources (les smartphones) dans l'environnement Java Android.

Annexe : Les Différentes captures de notre application

1. Accueil

L'application débute par le lancement de la fenêtre Splash qui montre le logo de notre application pour une seconde et disparait pour laisser sa place à la fenêtre principale qui est Accueil qui consiste à choisir entre chiffrer les données et aider à chiffrer les données.

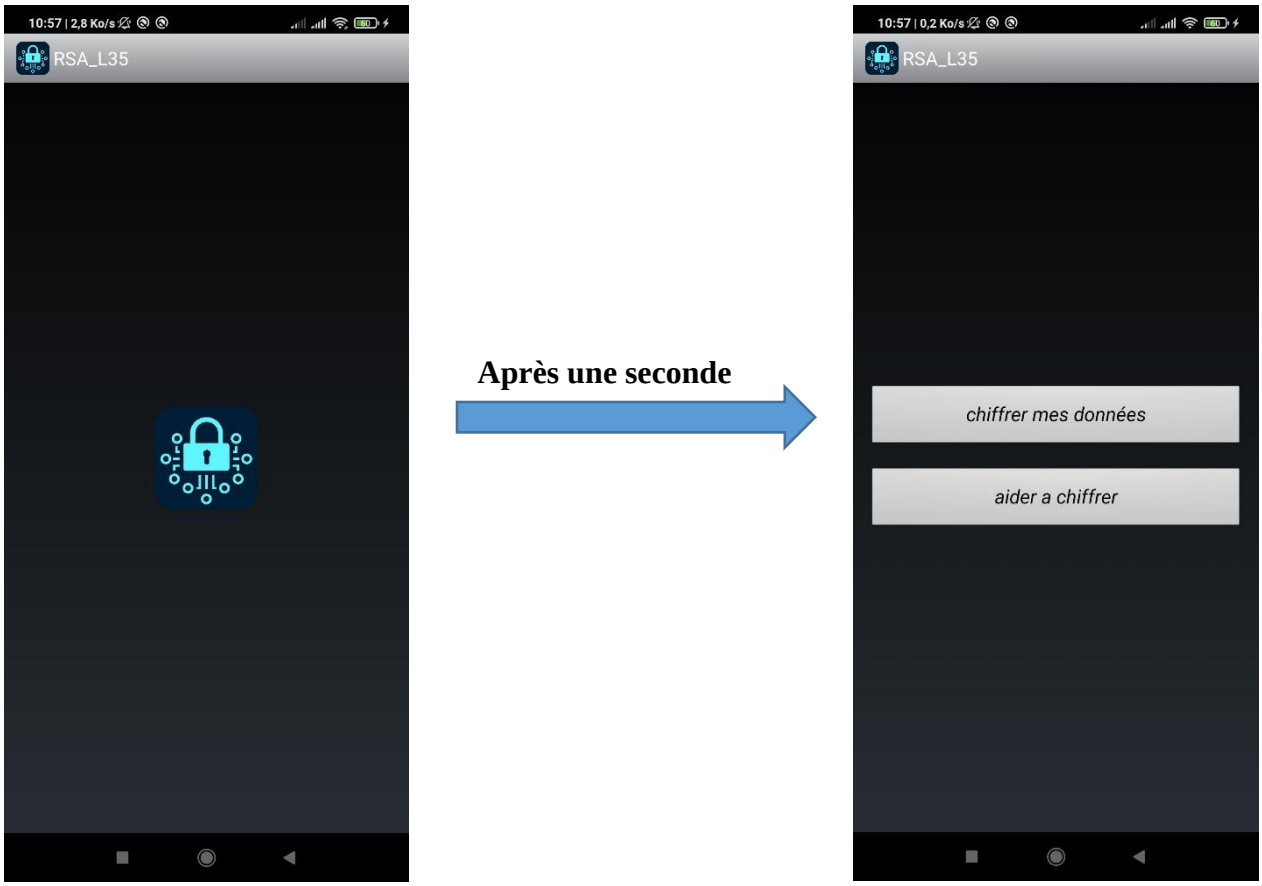


*Figure 25 : Capture_accueil_*

2. Chiffrer mes données

L'utilisateur qui veux chiffrer ses données arrive sur l'onglet « chiffrer mes données ».
Il pourra ensuite choisir la clé de chiffrement qui souhaite utiliser pour son chiffrement à travers une liste déroulante qui contient les clés entre 256 et 4096 bits.
L'utilisateur confirme son choix avec un bouton « confirmer » puis un texte lui confirmera la création des clés et le chemin du texte dans lequel les clés sont enregistrées sur son appareil (voir figure (27)).
Désormais l'utilisateur peut poursuivre son opérations grâce au bouton « Suivant » en bas de la fenêtre.

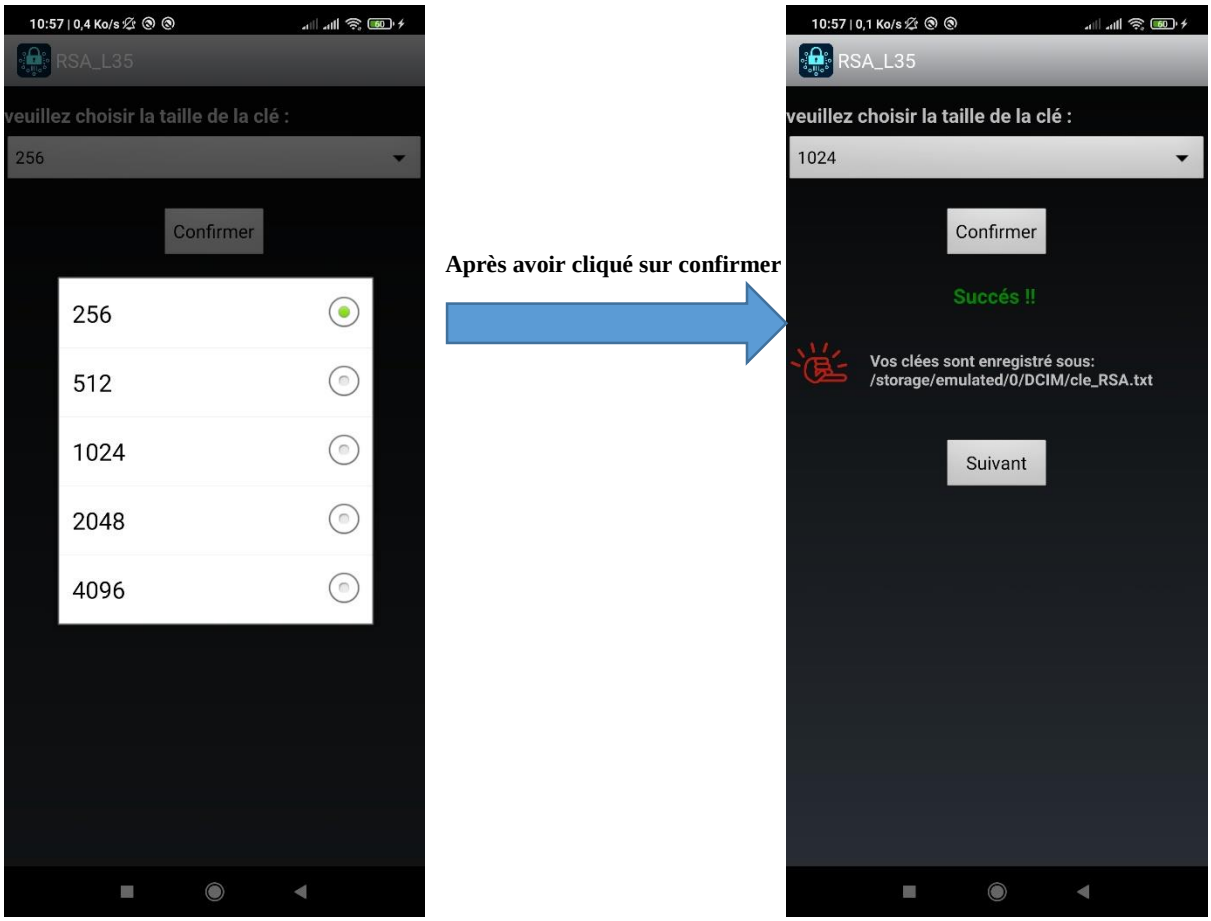


*Figure 26 : Capture _chiffrer mes données_*

```
**************** IMPORTANT ****************

Veulliez garder votre Cle Prive ( si vous la perdez vous ne
pourriez plus dechifrer votre message

**************** IMPORTANT ****************


Votre cle public :

237595232128163247538081666595393813657508973269731229064197l0
512138372859799001663505913586771716002761192628566767050036O8
163476847243321954111858446202918139505086072075008030261996Z6
227282786531107094007533593588937604371250828104110977020411J9
596125852784460512797480161153584351840114768373008790512721


Votre cle prive :

130577645463280524739343647133380592908625698872391861451698S2
809726277864086725974701787091505659316078930808100629920255g7
134002299042769277489667502542180077726666952720850806320725S1
210421024992138309936738245789140085518504723780012815276799T9
238230658297186665267752935052355956065237944398094257070885l9
943816422693828344908022237834049583299164442311325230086794O9
108338907547905434525155457459247381188654707064324934088277S8
118709128256770381950985907061782080459657004628120208762071O2
090325335941241084601244243754256120119814802935107052655234G5
398839823777099005827901748828381433129131554577644070042S7


Le N est egale a :

220956112363276578930403401484158317373152651284890086120074Z5
627258342562802369004092093295544740171702075054803338140459q2
304168296414162960347637019960336233409442769711240799438605q2
272930665788460605944909576781887420298059934011370516493813T1
120810854186329562211443092144058261956071726063624463995390q6
352521146585261894347993226182479221871442619851347740489021T7
581708722025013392441343949297573000438267654031227634568949J9
254741398717720725670565819590054624026552905716205562025342g0
352314503504024837398052425196805155368486909185598685467485O8
906612757891556919249942106539589213665076574372359454338631
```

*Figure 27 : Capture _les clés générées_*

2.1. Saisie du texte

L'onglet « texte à chiffrer » permet à l'utilisateur de saisir le texte qui veux chiffrer.

Le bouton « Chiffrer » permet de poursuivre l'opération et commencer le chiffrement du message.

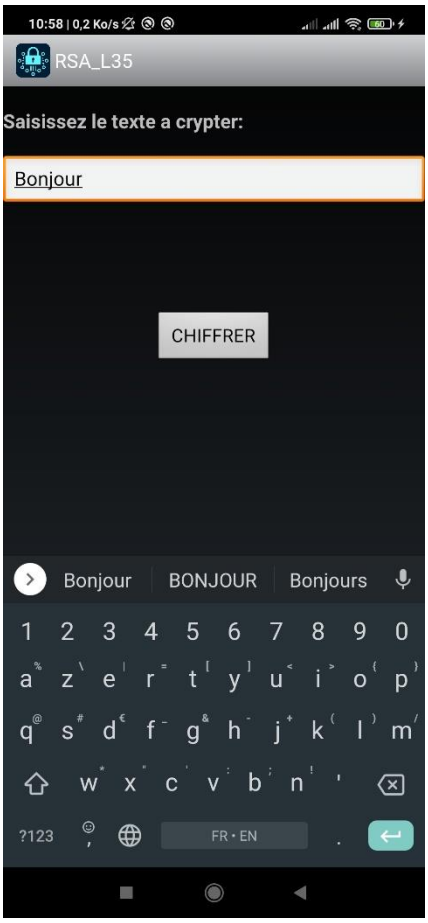


*Figure 28 : Capture _saisie du texte_*

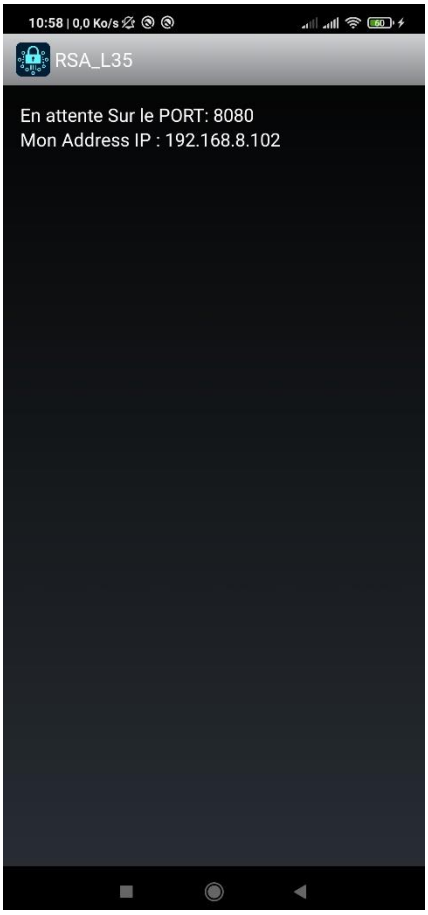


*Figure 29 : Capture _établir la connexion_*

2.2Etablir la connexion

Dans cette fenêtre le server se lance automatiquement et attend que d'autre appareilles se connecte à lui à l'aide du Port et de l'adresse IP qui sont afficher en haut de l'écran.

- Une fois que les appareilles se connectent, l'envoie des données peut enfin commencer et le serveur leur transmis les calculs à faire.
- Les connexions sont affichées sur liste déroulante comme la figure l'indique.

*Figure 30 : Capture _réception de messages_*

*Figure 31 : Capture _opération réussit_*

- Le serveur récupère tous les résultats transmis par les clients et à la fin, un bouton « Afficher le message chiffré » apparait.

- En cliquant dessus un texte seras affiché.
- En bas de l'écran un lien vers le chemin où le texte est sauvegardé sur l'appareil (voir la figure (33)).

3. Aider à chiffrer

Dans cet onglet l'utilisateur saisie l'adresse du serveur ainsi que le port de connexion. Lorsqu'il clique sur le bouton « Se Connecter » une connexion avec le serveur seras établie et la transaction de données peut commencer. Lorsque tous les calculs seront finies, un message seras affiché « opération finie, merci pour votre aide ».

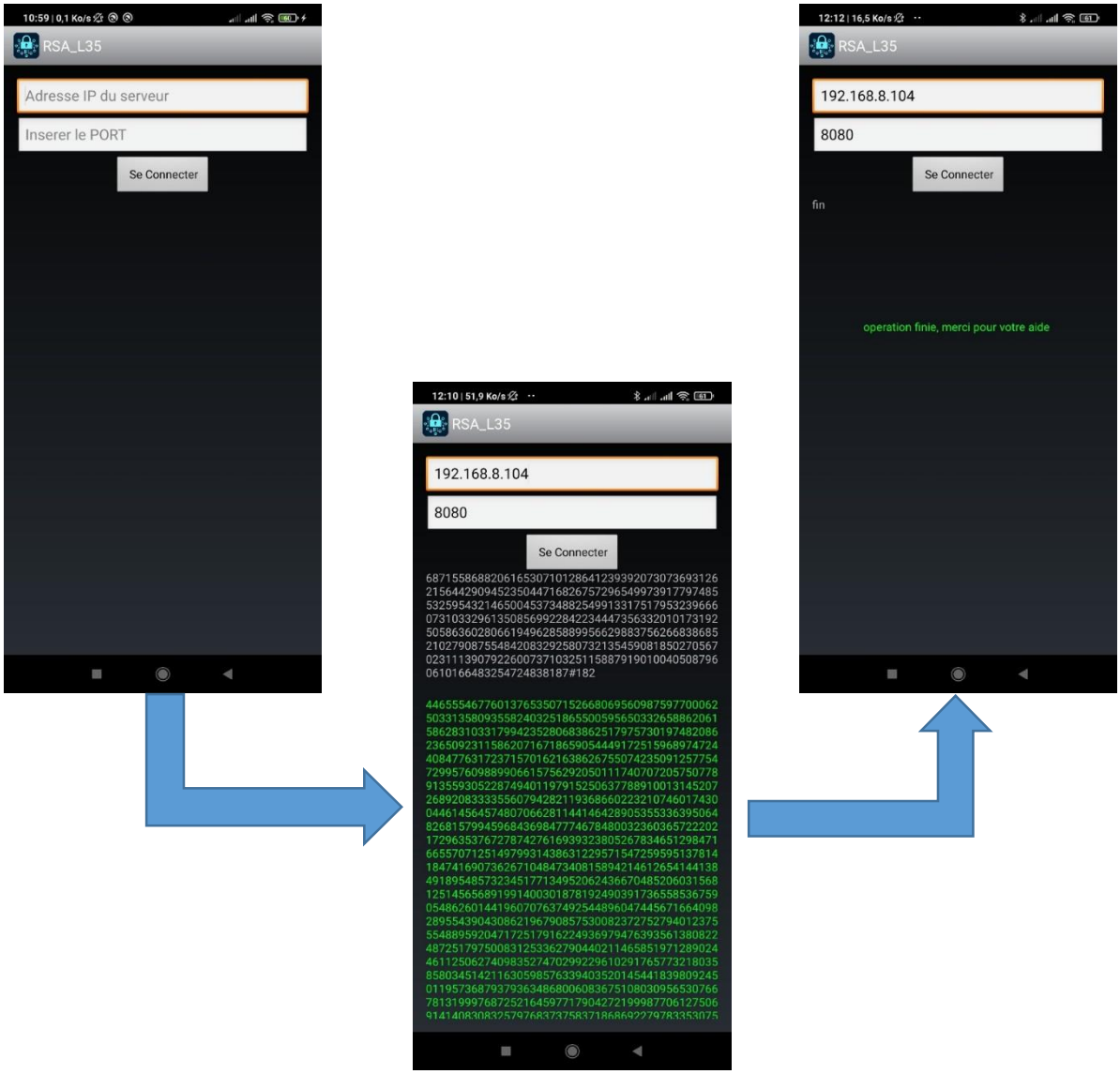


*Figure 32 : Capture _aider à chiffrer_*

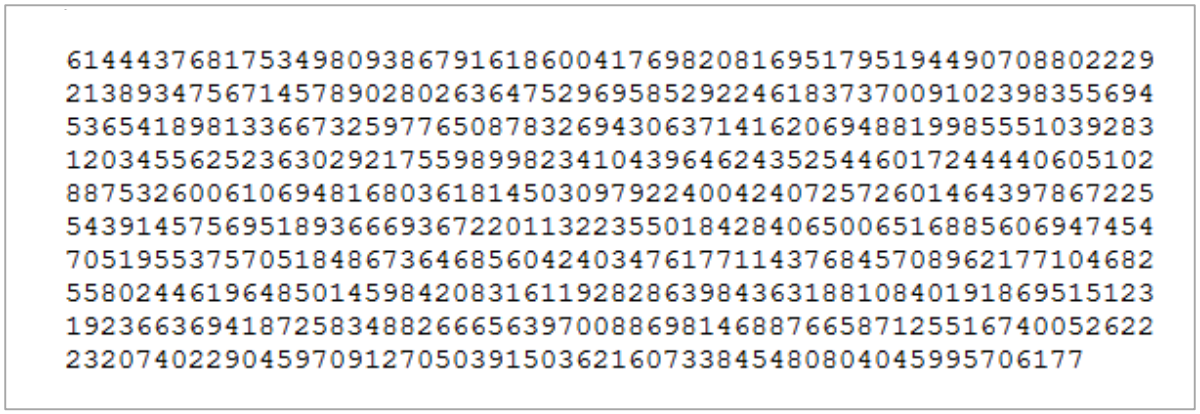


*Figure 33 : Capture _message chiffré_*

BIBLIOGRAPHIE

Ouvrages

[1] : Mollin, R.A.RSA and Public-Key Cryptography, CRC Press, Inc., Boca Raton, FL, USA.2002

[2]: CAVALLAR, Stefania, DODSON, Bruce, LENSTRA, Arjen K., *et al.* Factorization of a 512-bit RSA modulus. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg, 2000. p. 1-18.

[3]: KOÇ, Çetin K .High-Speed RSA Implementation. Technical report, RSA Laboratories TR 201 (November 1994)

[4]: WIENER, Michael J. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information theory*, 1990, vol. 36, no 3, p. 553-558.

[5]: MOHAMMED, Ahmed et ALKHELAIFI, Abdulrahman. RSA: A number of formulas to improve the search for p+ q p+ q. *Journal of Mathematical Cryptology*, 2017, vol. 11, no 4, p. 195-203.

[6] : M^{me} HADAOUI , Thèse de Doctorat de l'Université Mouloud MAMMERI TIZI OUZOU : *Cryptage RSA par une communauté de réseau social à ressources limitées, 2020.*

[7] : JOYE, Marc. Highly regular right-to-left algorithms for scalar multiplication. In : *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, Berlin, Heidelberg, 2007. p. 135-147.

Site internet

[N1] : https://www.arsouyes.org/blog/2019/52_Hash_Function_Cryptography/ ,date de consultation 25/09/2020

[N2] : https://csrc.nist.gov/publications/detail/fips/180/2/archive/2004-02-25/ , date de consultation 25/10/2020

[N3] : https://help.eclipse.org/2020-09/index.jsp , date de consultation 25/10/2020

[N4] : https://www.futura-sciences.com/tech/definitions/internet-java-485/ ,date de

consultation 26/10/2020

[N5] : https://cours-informatique-gratuit.fr/dictionnaire/android/ ,date de consultation 26/10/2020

[N6] : https://gs.statcounter.com/os-market-share/desktop-mobile/worldwide/#monthly-201909-202009 ,date de consultation 26/10/2020