

1ème année prépa en école d’ingénieurs   
Projet Electronique 1ème semestre

Projet StrongBox 3000

Projet Electronique, Livrable 4

Felix Barton ; Gaël saurais ; théo Marcilla ; nicolas kouyoumji

2020

Table des matières

[1 – Contexte 2](#_Toc93337180)

[2 – Contraintes 2](#_Toc93337181)

[3 – Le code Arduino 2](#_Toc93337182)

[4 –Conclusion 7](#_Toc93337183)

# 1 – Contexte

« La célèbre agence d'espionnage MI7 a vécu des heures sombres ces dernières semaines. A plusieurs reprises, le matériel mis à disposition sur le terrain pour leurs agents a été détourné.

Malgré les codes secrets et autres stratagèmes, les cachettes ne sont plus assez sûres et il faut envisager une autre solution !

L'agent R, responsable des innovations technologiques, a proposé de travailler sur un prototype de coffre-fort nouvelle génération avec des mécanismes d'authentification plus efficaces. »

Il est maintenant question d’utiliser les connaissances apportées par le Livrable 3 et les logigrammes pour concevoir le programme du coffre-fort.

# 2 – Contraintes

Nous devons obligatoirement utiliser une carte Arduino.

# 3 – Le code Arduino

Graphical user interface, application

Description automatically generated

Nous avons dans un premier lieu la déclaration des variables nécessaires.

La variable authentication sera le comparatif de notre code.

Cardnbr et i seront utilisés dans pour reconnaitre le modèle de carte entré par l’agent.

Agent et inString seront utilisé dans MA2, MA5 et pour communiquer avec l’agent.

securityLevel est un tableau comportant tout les niveaux de sécurité ainsi que le niveau 0, un niveau d’erreur.

publicKey,privateKey , nbrAgent ainsi que crypted sont utilisés dans MA2 pour le chiffrement asymétrique et CardID est utilisé pour comparer le modèle, la lettre de l’agent et le numéro de carte dans MA5.

A picture containing text

Description automatically generatedA picture containing graphical user interface

Description automatically generatedDans un deuxième temps la mise en place de la fonction Level qui attribue un niveau de sécurité en fonction du modèle de carte attribué avec une lecture de l’array de la variable securityLevel.

Pour aller chercher directement la valeur souhaitée dans notre array, on utilise la fonction Cardread qui change la variable Cardnbr à une variable utilisable dans notre tableau.

A picture containing text

Description automatically generatedNous avons créé la fonction answer ainsi que la fonction read\_input afin d’être en mesure de lire ce que l’agent écrira dans la console et nous renvoyons cette String. Ainsi en faisant un appel de cette fonction, la variable que l’on enverra prendra la valeur que l’utilisateur va écrire.

Text

Description automatically generated

La fonction error nous permet de gérer les erreurs potentielles et d’automatiquement reset le programme après un court instant.

Graphical user interface, text, application, email

Description automatically generatedA picture containing text

Description automatically generatedText

Description automatically generatedLa fonction ci-dessus effectue ce qui est nécessaire pour MA1.

Nous avons ensuite créé la fonction MA2 qui est un algorithme de chiffrement asymétrique utilisé grâce aux clés publiques, tableau de int nommé publicKeys.

Text

Description automatically generatedLa fonction MA2 fais appel à deux sous fonctions, encrypt et modexp. encrpyt renvoyant simplement la variable crypted qui sera calculé dans modexp et modexp étant la formule pour le chiffrement asymétrique.

Text

Description automatically generatedPar la suite nous avons MA3 et MA4 qui sont sensiblement similaires.

Elles sont toujours correctes et renvoient la valeur authentication.

Text

Description automatically generatedA picture containing table

Description automatically generatedLa fonction MA5 est un algorithme de comparaison entre le modèle renseigné par l’agent et sa lettre d’agent, si les deux ne coincident pas, alors l’agent n’est pas correctement identifié.

Le void setup appel toute les fonctions nécessaires au bon fonctionnement du code et envoi la variable authentication.

Cette dernière ne change qu’en cas d’erreur, ainsi si elle est toujours égale à 1, aucune erreur ne s’est produite donc le coffre-fort peut s’ouvrir.

# 4 –Conclusion

Le code est fonctionnel et les erreurs sont traités correctement. Il est suffisamment optimisé pour être lu par la carte Arduino et être exécuté sans problème.