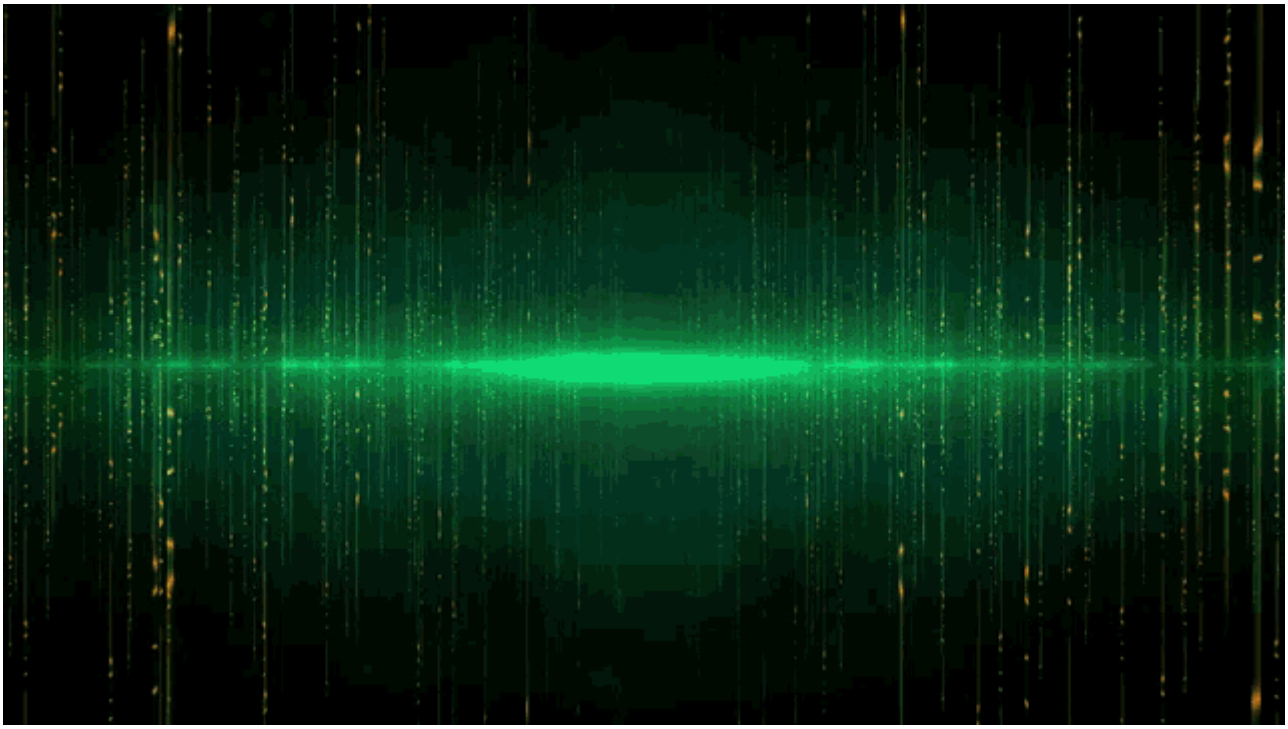
Team de décryptage AKI

Équipe de décryptage initiée par M. Gillet composé de M. FOFANA, M. BARRY et M. ABADA.



Le Projet

- Contexte
    - M. GILLET nous appelle une deuxième fois suite à notre merveilleuse réussite avec la première mission qu'il nous a attribué.
    - Alice et Bob, deux tourtereaux qui s'aiment passionnément, vivent un amour secret depuis des années. Comment arrivent-t-ils à garder ce secret ? L'échang de message chiffrés. Ils utilisent un protocole se nommant PlutotBonneConfidentialité dont il peut être résumer ainsi :
        - A l'aide d'un protocole cryptographique asymétrique, ils s'échangent une clé de session. Cette clé servira à chiffrer les communications futures.

        - Une fois cette clé choisie et échangée, le chiffrement des messages s'opère grâce à un protocole cryptographique symétrique. Ils peuvent ainsi communiquer en toute liberté !
    - Eve, la meilleure amie d'Alice, souhaite réveler au grand jour leur histoire et veut donc déchiffrer leur communication.

- Notre employeur nous a donc aider en structurant 4 parties. La première est un peu pour tâter le terrain, comprendre le sujet. La deuxième est le développement d'un protocol expérimental permettant de mettre en évidence la différence de sécurité entre les deux protocoles AES et RSA. La troisième est l'analyse des messages envoyés par Alice et Bob. Enfin, la dernière et quatrième partie, concerne la réponse à plusieurs questions afin d'aller plus loin dans le travail.

Première partie

- **RSA** est un protocole de cryptographie asymétrique qui est considéré comme sûr lorsqu'il est utilisé correctement. Cela signifie que si Alice et Bob utilisent des clés de taille suffisante et suivent les bonnes pratiques (par exemple, ne pas réutiliser les clés, garder les clés privées en sécurité), alors Eve ne devrait pas être en mesure de déchiffrer les messages chiffrés avec RSA. **Cependant**, il est important de noter que la sécurité de RSA repose sur la difficulté de la factorisation des grands nombres premiers. Si Eve avait accès à une puissance de calcul suffisante (par exemple, un ordinateur quantique), elle pourrait théoriquement factoriser les clés RSA et déchiffrer les messages. Mais pour l'instant, avec la technologie actuelle, cela reste largement hors de portée. **En conclusion**, si RSA est utilisé correctement, Eve ne devrait pas être en mesure de déchiffrer les communications entre Alice et Bob. Cependant, la sécurité en informatique est un domaine en constante évolution et il est important de rester informé des dernières avancées et menaces.

- L'algorithme SDES (Simplified Data Encryption Standard) est considéré comme peu sécurisé en raison de la taille relativement courte de sa clé. En effet, SDES utilise une clé de 10 bits, ce qui signifie qu'il y a 2^10, soit 1024 clés possibles. Dans le contexte d'une attaque par force brute, un attaquant essaierait toutes les combinaisons possibles jusqu'à trouver la bonne clé. Avec seulement 1024 clés possibles, un ordinateur moderne pourrait essayer toutes ces combinaisons en un temps très court, rendant l'algorithme vulnérable à ce type d'attaque. De plus, SDES est un algorithme de chiffrement symétrique, ce qui signifie que la même clé est utilisée pour le chiffrement et le déchiffrement. Si la clé est découverte lors d'une attaque par force brute, l'attaquant aurait alors accès à toutes les données chiffrées avec cette clé. Pour ces raisons, SDES n'est pas considéré comme un algorithme de chiffrement sécurisé pour protéger des informations sensibles dans un environnement où les attaques par force brute sont possibles. Des algorithmes de chiffrement plus robustes, tels que l'AES (Advanced Encryption Standard), sont recommandés pour une meilleure sécurité.

- Le Double DES est une technique de chiffrement qui utilise deux instances de DES sur le même texte en clair. Dans les deux instances, il utilise des clés différentes pour chiffrer le texte en clair. Cependant, bien que le Double DES utilise une clé de 112 bits, il n'offre qu'un niveau de sécurité de 2^56 et non de 2^112. C'est à cause de l'attaque "rencontre au milieu" qui peut être utilisée pour percer le Double DES. Dans une attaque de rencontre au milieu, l'attaquant utilise toutes les clés possibles pour crypter le texte en clair et stocker les résultats. Ensuite, il utilise toutes les clés disponibles pour déchiffrer le texte qui a été chiffré. L'attaquant a trouvé les deux clés si l'une des valeurs de déchiffrement correspondait à l'une des valeurs de chiffrement stockées. Le nombre d'essais pour trouver la clé dans une attaque de rencontre au milieu est de 2^56, ce qui est beaucoup moins que les 2^112 essais nécessaires pour une

attaque par force brute sur une clé de 112 bits. C'est pourquoi le Double DES n'offre pas un niveau de sécurité beaucoup plus élevé que le DES simple, malgré l'utilisation d'une clé deux fois plus longue.

Deuxième partie

1. L'utilisation de clés AES de 256 bits est généralement considérée comme une amélioration de la sécurité, cela peut poser des défis potentiels en termes de performances et de compatibilité. Évaluer les compromis entre sécurité, performance et exigences du système est essentiel pour déterminer si ce changement pose un problème significatif pour Eve.

2. Le temps d'exécution de l'algorithme de chiffrement AES dépend de plusieurs facteurs, notamment la taille de la clé, le mode de chiffrement et le matériel utilisé. Dans le cas d'AES-256, le temps d'exécution dépend de la taille de la clé (256 bits), du mode de chiffrement (CBC) et du matériel utilisé (CPU, GPU, FPGA, etc.). Si on compare le chiffrement AES avec le chiffrement SDES, il est plus rapide que SDES que ce soit sur de petites clés ou de grandes clés car sa conception a été optimisée pour une large gamme de tailles de clés, et il est largement implémenté matériellement et logiciellement, ce qui lui confère des performances plus rapides même avec des clés plus grandes. SDES dans son cas, se dégradent considérablement lorsqu'il est utilisé avec des clés plus grandes, tandis que AES maintient généralement de bonnes performances même avec des clés plus grandes en raison de son approche plus moderne et de ses optimisations.

Temps d'exécution de l'algorithme de chiffrement AES obtenu : 4.654099939216394e-05 s

** Configuration de l'ordinateur Macbbok air M2 ** :

- Processeur : Apple M2 (8 cœurs, 16 threads)
- RAM : 8 Go
- GPU : Apple M2
- Système d'exploitation : macOS Sonoma 14.0

Temps d'exécution de l'algorithme de déchiffrement AES obtenu avec la même configuration en moyenne : 2.795899945340352e-05 s

Temps d'exécution de l'algorithme de chiffrement SDES obtenu avec la même configuration en moyenne : 0.0007921660007923492 s

Estimation du temps de cassage d'AES-256 :

- **Configuration de l'ordinateur sur lequel ont été effectuées les tests** :
    - Processeur : Intel Core i7-11700K (8 cœurs, 16 threads)
    - RAM : 16 Go
    - GPU : NVIDIA RTX 3070
    - Système d'exploitation : Windows 10
- **Complexité de l'algorithme de cassage** : Supposons une attaque par force brute exhaustive.
- **Longueur de la clé AES** : Clé de 256 bits

La clé AES-256 a une longueur de 256 bits, ce qui signifie qu'il y a (2^{256}) combinaisons possibles pour la clé.

En supposant un taux de test de clés de l'ordre de plusieurs milliards de clés par seconde (ce qui est assez optimiste pour une attaque par force brute), calculons le temps approximatif nécessaire pour tester toutes les combinaisons possibles de clés :

Nombre de combinaisons possibles pour la clé = 2^{256}

Si nous testons, par exemple, 1 milliard (1 x (10^9)) de clés par seconde :

Nombre de secondes nécessaires = Nombre de combinaisons possibles\Nombre de clés testées par seconde

Temps nécessaire = 2^{256} / 1 * 10^9

Calculons :

Temps nécessaire = 2^{256} / (1 * 10^9) = 1.1579 * 10^{60} secondes

En convertissant cela en années :

Temps nécessaire en années : 1.1579 * 10^60 / (60 * 60 * 24 * 365) = 3.6716 * 10^49

Cela dépasse de loin l'âge de l'univers (environ 13,8 milliards d'années).

Remarques :
C'est une estimation simplifiée et optimiste. En réalité, une attaque par force brute contre AES-256 avec les ressources informatiques actuelles est considérée comme impossible en raison du nombre colossal de combinaisons possibles. Les estimations reposent sur des suppositions sur les capacités de calcul et peuvent varier en fonction de nombreux autres facteurs, y compris les avancées technologiques et les nouvelles méthodes d'attaque.

Attaques par analyse différentielle

L'analyse différentielle est une méthode d'attaque qui exploite les différences de comportement du chiffrement pour obtenir des informations sur la clé secrète. Cette méthode repose sur l'observation des différences dans les opérations effectuées par l'algorithme de chiffrement lorsqu'il traite des données similaires avec des clés légèrement différentes.

Explication succinte :

L'analyse différentielle compare les différences dans les sorties produites par l'algorithme de chiffrement lorsqu'il traite des blocs de données similaires (par exemple, des bits modifiés d'un bloc de texte chiffré) avec des clés légèrement différentes. En observant ces différences, un attaquant peut extraire des informations sur la clé utilisée dans le processus de chiffrement.

Cette méthode d'attaque nécessite une connaissance approfondie de l'algorithme de chiffrement et peut être complexe à mettre en œuvre. Les concepteurs d'algorithmes cryptographiques modernes prennent souvent des mesures pour résister à ce type d'attaque en rendant leurs schémas de chiffrement résistants à l'analyse différentielle.

Analyse des Images identiques

Pour la partie analyse des images, il a fallu récupérer les images qui sont sur CELENE et les analyser bits par bits. La partie de la clé qui nous intéressait était sur 64 bits alors dans la fonction de comparaison, nous avons

pris les 64 premiers pixels de chaque image et nous les avons comparés. Si les 64 premiers bits étaient identiques, alors nous avons considéré que les images étaient identiques. Ceci relevait de notre logique, cependant d'un point de vue algorithmique à la fin ce n'était pas les bons bits qui nous étaient renvoyés. Après réflexion et tests, nous avons remarqué qu'il fallait en fait ne parcourir que la deuxième image (rossignol2.bmp) car c'est elle qui contient les 64 bits qui nous intéressent. Nous avons donc modifié notre fonction de comparaison pour qu'elle ne prenne en compte que la deuxième image et à chaque fois que l'on récupérait un pixel, on ajoutait le bit de poids faible à la clé extraite en le transformant en string. Ainsi, nous avons pu extraire la partie de la clé qui nous intéressait.

Troisième partie

Analyse de la trace réseau/messages échangés entre Alice et Bob

Une fois qu'on ait obtenu une partie de la clé, pour obtenir le reste de la clé, il suffit de répéter 4 fois la partie de la clé que l'on a obtenu. Ensuite, il faut récupérer les messages échangés entre Alice et Bob. Pour cela, nous avons utilisé le logiciel Wireshark. Nous avons donc ouvert la trace réseau et nous avons filtré les paquets en fonction du protocole UDP et du port 9999. Ensuite, nous avons récupéré les données des paquets et nous avons pu voir que les messages étaient chiffrés avec AES-256. Nous avons donc utilisé la bibliothèque cryptography pour déchiffrer les messages. Pour faire tout cela, nous avons utilisé deux fonctions : extractey_key et decrypt_message. La première fonction permet d'extraire la clé de la trace réseau donc de lire un fichier de paquets et d'extraire les messages chiffrés et la deuxième qui permet de déchiffrer un message chiffré avec AES-256 en mode CBC. Une fois ces deux fonctions réalisées il a fallu passer par plusieurs transformations :

- On a d'abord transformé la clé de session en entier
- Par la suite nous avons transformé cette clé en entier en binaire pour qu'elle soit en bytes parce que c'est nécessaire pour la fonction decrypt_message avec AES-256
- Ensuite, nous avons récupéré les messages chiffrés et nous les avons déchiffrés avec la fonction decrypt_message

Quatrième partie

1. Alice et Bob utilisent toujours la même clé. Est-ce une bonne pratique?

Ce n'est clairement pas une bonne pratique. AES est un algorithme de chiffrement symétrique, ce qui signifie que la même clé est utilisée pour le chiffrement et le déchiffrement. Si la clé est compromise, l'attaquant peut déchiffrer toutes les données chiffrées avec cette clé. Afin de respecter la bonne pratique, les deux personnes pratiquant l'AES se doivent de changer régulièrement de clé, pour garder une sécurité élevée. De plus, il est préférable d'utiliser des clés fortes. Pour une sécurité optimale, ces bonnes pratiques sont nécessaire pour réduire l'impact si une clé venait à être compromise.

2. Le protocole PlutotBonneConfidentialité est inspiré d’un vrai protocole réseau. Lequel? Décrivez la partie associé à la certification des clés qui est absente de PlutotBonneConfidentialité

Le réseau auquel le protocole PlutotBonneConfidentialité est inspiré est le protocole TLS (Transport Layer Security). Ce dernier est couramment utilisé dans la sécurisation des communications sur le web.

Ce dernier est un protocole de sécurité qui assure la confidentialité et l'intégrité des données lors de leur transmission sur un réseau, généralement avec TCP. Il est très utilisé pour établir des connexions sécurisées

sur Internet, par exemple avec les transactions bancaires, l'accès sécurisé aux courriers électroniques, et d'autres échanges possédant des données sensibles. TLS utilise un processus de certification de clés pour authentifier les deux communiquants et établit des clés de session secrètes pour chiffrer les données pendant la transmission.

Dans TLS, le processus de certification des clés se déroule comme suit :

D'abord, le serveur envoie un certificat numérique signé à son client. Ensuite, le client vérifie la validité du certificat en utilisant la clé publique de l'autorité de certification associée. Si la vérification réussit, le client et le serveur génèrent des clés de session partagées. Enfin, ces clés de session sont utilisées pour chiffrer la communication entre le client et le serveur.

3. Il n'y a pas que pour l'échange de mots doux q'un tel protocole peut se révéler utile. . . Donnez au moins deux autres exemples de contexte où cela peut se révéler utile.

- Communications par Courrier Électronique (SMTP/IMAP/POP):

TLS est souvent utilisé pour sécuriser les communications par courrier électronique. Lorsque les serveurs de messagerie et les clients de messagerie prennent en charge TLS, il est possible d'établir des connexions sécurisées pour protéger le contenu des courriels contre une interception non autorisée, par exemple d'un message.

- Accès à des Applications Web (HTTPS):

TLS est le plus souvent utilisé dans le contexte du protocole HTTPS. Lorsque vous accédez à des sites Web via HTTPS, le protocole TLS est utilisé pour crypter les données entre votre navigateur et le serveur Web, ce qui garantit la confidentialité des informations sensibles telles que les noms d'utilisateur, les mots de passe, et les données bancaires.

4. Connaissez-vous des applications de messagerie utilisant des mécanismes de chiffrement similaires? (on parle parfois de chiffrement de bout en bout)? Citez-en au moins deux et décrivez brièvement les mécanismes cryptographiques sous-jacent

Plusieurs réseaux sociaux utilisent le chiffrement de bout en bout, notamment *WhatsApp* et *Telegram*. *WhatsApp* utilise le protocole Signal. Ce dernier utilise des clés Diffie-Hellman pour échanger secrètement des clés de session, qui sont utilisées pour chiffrer les communications audio, vidéo et par message. Les clés de session sont générées de manière dynamique pour chaque session de communication, ce qui renforce la sécurité. Le protocole utilise des clés publiques, une master_secret clé pour maintenir une session sécurisée, et des clés de session pour chaque message. Ces clés sont générées à partir d'une combinaison de la courbe elliptique Diffe-Hellman et du protocole HMAC-SHA256. Les clés de session sont constamment mises à jour pour renforcer la sécurité. L'échange de clés Diffie-Hellman permet à deux parties de créer une clé commune secrète, même lorsqu'elles communiquent sur un canal non sécurisé. Voici un résumé simple :

Choix des paramètres : Les parties conviennent de paramètres publics, un nombre premier p et un générateur g.

Clés privées : Chaque partie choisit une clé privée secrète (a pour la première partie, b pour la deuxième).

Clés publiques : Chaque partie calcule une clé publique en utilisant les paramètres et sa clé privée.

Échange des clés publiques : Les parties s'échangent leurs clés publiques.

Clé partagée : Chaque partie calcule la clé partagée en utilisant sa clé privée et la clé publique de l'autre partie.

Résultat : Les deux parties ont désormais une clé partagée commune sans jamais avoir échangé leurs clés privées directement. Cette clé partagée peut être utilisée pour sécuriser la communication entre les parties.

5. Récemment, différents projets de loi et règlements (CSAR, EARN IT Act) visent à inciter voir obliger les fournisseurs de services numériques à pouvoir déchiffrer (et donc analyser) les communications de leur.e.s utilisateur.rices. Discutez des arguments en faveur ou contre ces législations, notamment en matière de vie privée

Alors comme nous pouvons nous en douter, il y a plus d'arguments contre ces projets de loi que l'inverse. En effet, les personnes s'affolent directement lorsqu'ils apprennent que leurs messages peuvent être lus par leur fournisseur. Cependant, ce n'est pas pour autant que ces mesures soient totalement négatif. Voici les différents arguments en faveur :

- Une sécurité sur le territoire : On y pense pas directement, mais la possibilité de lire les messages chiffrés permet d'attraper toutes les activités terroristes, criminalités organisées et toutes autres menaces à la sécurité nationale.
- Protection des victimes : Toutes les activités illicites telles que la pédopornographie ou bien l'harcèlement, seraient un trésor pour condamner les auteurs de ces actes ignobles, et ainsi aider les victimes qui subissent ces derniers.
- Responsabilité des plateformes : En donnant une plus grande responsabilité aux plateformes dans la modération, cela aidera à lutter contre les abus en ligne, par exemple les partisans de l'EARN IT Act soutiennent cette idée.

Voici les arguments contre ces projets de lois :

- Droit à la vie privée : En effet, un accès forcé aux messages des civils peut être perçu comme une violation du droit fondamental à la vie privée. Les individus possèdent un droit d'échanger de manière confidentielle sans craindre une surveillance excessive de l'État.
- Risque de surveillance de masse : Si on accepte ces projets de lois, cela voudrait dire que les plateformes auront la capacité de surveiller même des personnes qui ne sont pas soupçonnées de comportements criminels, c'est-à-dire une surveillance abusive.
- Faiblesse de la sécurité des échanges : La création de "backdoors" - portes dérobées - pour permettre l'accès aux communications pourrait rendre les systèmes plus vulnérables aux pirates et aux hackers qui pourraient exploiter ces faiblesses pour accéder à des informations sensibles, présents dans les messages chiffrés.
- Inefficacité des projets de lois : En réflechissant un peu plus, on remarque une chose : en ordonnant au plateformes de fournir un moyen de déchiffrer les communications, les criminels auront juste à choisir d'autres moyens de communications hors des plateformes réglementées. Plusieurs experts affirment cet argument.

Répartition des tâches

Ibrahima :

J'ai tout d'abord commencé à faire la partie 1 du projet. Après avoir analysé les questions qui étaient proposées, j'ai tenté d'apporter des premiers éléments de réponses puis mes deux autres camarades ont à leur tour apporté leurs réponses. Après ça, j'ai réalisé les fonctions qui nous étaient demandées, j'ai d'abord codé le cassage brutal puis Khalil a retravaillé dessus. Après ça je me suis penché sur le cassage astucieux qui dans un premier temps puis Abdoulahi a complété cette fonction et a réussi à réduire le temps d'exécution. Au final nous avons tous contribué à cette partie. Par la suite j'ai poursuivi avec la partie 2 que j'ai entièrement réalisé.

Khalil : Premièrement, j'ai commencé par prendre conscience des différentes attentes de ce projet attribué par M. GILLET, notre employeur. Au départ, cela m'a paru presque impossible, mais finalement je retiens une chose : ne jamais se fier aux apparences. J'ai commencé, accompagné de mes collègues, qui vous diront de même, à répondre aux différentes questions de la première partie. En effet, un projet ne commence jamais par le code mais par l'étude du projet même. Ensuite, nous avons continué sur l'implémentation des fonctions demandé, Ibrahima a commencé puis j'ai vérifié que tous aller bien pour la suite, deux cerveaux valent mieux qu'un. Ensuite, moi et Ibrahima avons persisté assez longtemps sur le cassage brutal pour avoir trouvé une petite erreur bête, mais bon c'est le comble d'un développeur ces petites erreurs bête (décidément deux cerveaux ne valent peut être pas mieux qu'un). Enfin, pour accélérer le pas, je suis passé directement à la partie 4, qui ne se lier pas trop aux autres (à part deux questions). En effet, je ne suis pas passé de la partie 1 à la partie 4 sans étudier les parties intérmédiaires, j'ai vérifié ce que mes collègues ont fait pour ainsi comprendre ces parties.

Abdoulahi :

J'ai commencé par travailler sur la partie 1 du projet en collaboration avec mon collègue Ibrahima, et avec son aide, nous avons pu créer le rendu tel qu'il est aujourd'hui. Nous avons travaillé en collaboration durant tout le projet, ce qui signifie que je n'ai pas seulement codé individuellement, mais j'ai également aidé sur leurs tâches en donnant mon avis, comme par exemple sur les questions des différentes parties, en suggérant des améliorations ou des suppressions. Ils ont fait de même pour moi. Ensuite, nous avons répondu aux questions de manière à ce que tout le monde soit d'accord avec la réponse. Enfin, j'ai mis en place une petite application à exécuter dans le terminal, qui permet de tester nos fonctions implémentées durant le projet.

Credits

Nous tenons à remercier tous le personnel de l'IUT d'Orléans pour leur soutien et leur aide dans ce projet. Nous tenons aussi à remercier M. GILLET pour nous avoir confié ce projet, ce qui nous a permis de nous améliorer dans le domaine de la cryptographie. De plus, cela signifie que notre employeur M. GILLET a confiance en nous et nous le remercions pour cela.

License

(back to top)