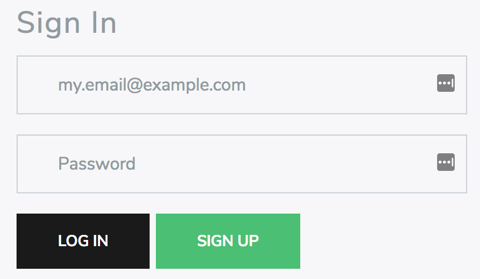
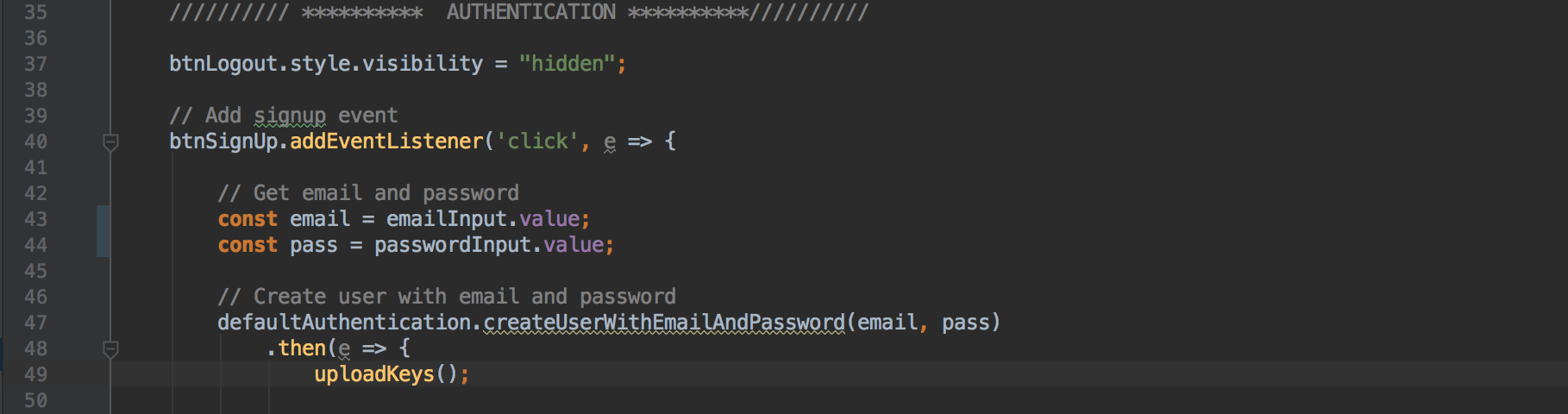
Code revealed

ENCRYPTIE

1. Public en private keys genereren (RSA), private key encrypteren met symmetric-key (AES) die de user-password (SHA-256) gebruikt.

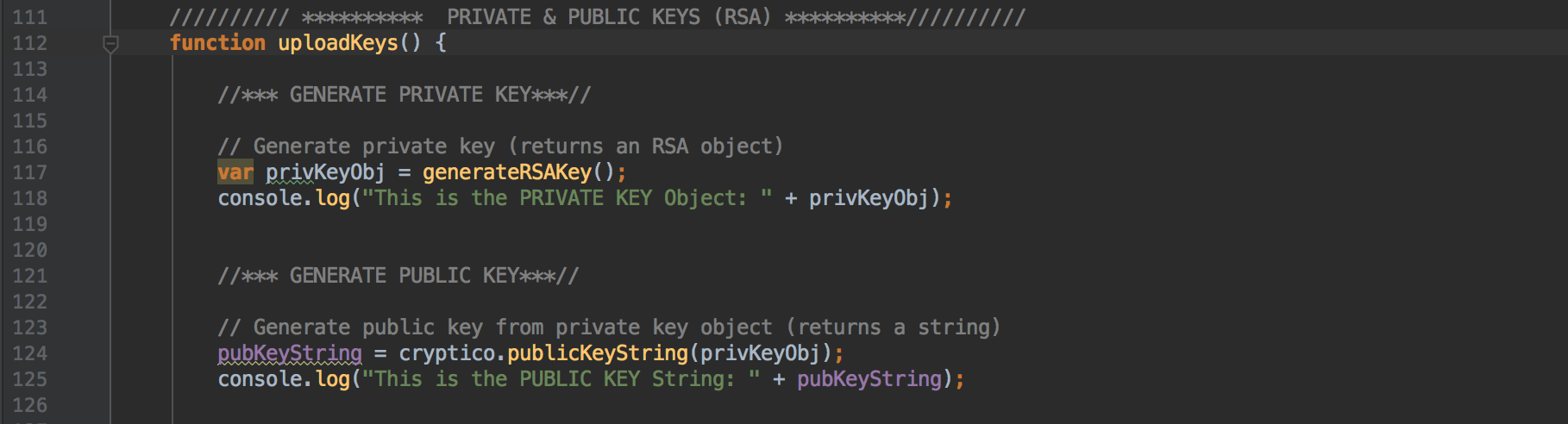
* Aanmaken van private en public key gebeurt één keer, als de user zich aanmeldt:

Via de uploadKeys() functie RSA keys woorden aangemaakt:

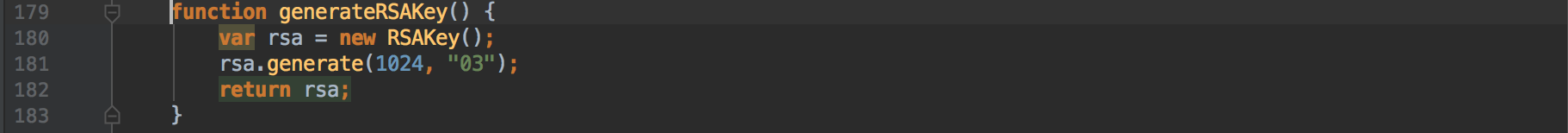


…

Om een public key aan te maken moet eerst de private key gegenereerd woorden. Dit gebeurt aan de hand van de functie generateRSAKey(). Deze functie geeft een object terug en deze object is dan gebruikt oom de public key aan te maken via de cryptico library functie publicKeyString(). Het resultaat is een string dat meegedeeld kan woorden via de Internet.



De generateRSAKey() maakt een nieuwe RSAKey object (private key) aan met een lengte van 1024 bits.



private en public keys aangemaakt

Nu dat we de 2 keys hebben, moeten ze veilig op de Internet geplaatst worden.

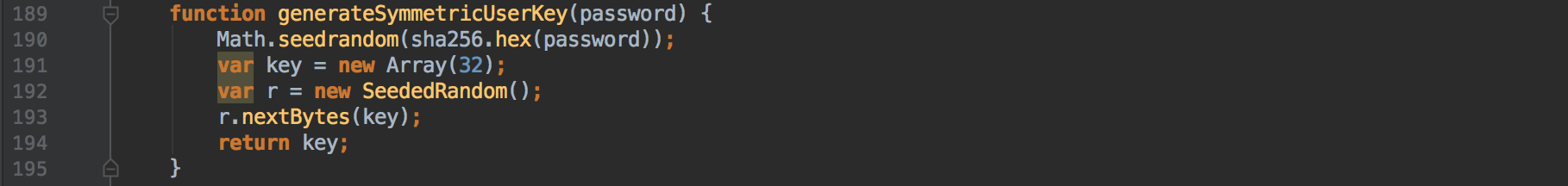
* Public key: iedereen moet toegang hebben tot deze key, dus de **public** key **moet** **niet** geëncrypteerd **worden**.
* Private key: deze key moet alleen bekend zijn voor de private key owner. Dus, de **private** key **moet** geëncrypteerd worden. De encryptie zal gebeuren aan de hand van de user-password en de generatie van een symmetric key.



De functie generateSymmetricUserKey(password) maakt een hash-256 van de password en gebruikt deze als een seed om een symmetric key aan te maken.

Omdat de password altijd dezelfde is, dan zal de symmetric key ook altijd dezelfde zijn.

Door deze methode te gebruiken, hoeft de password nergens opgeslagen te worden.



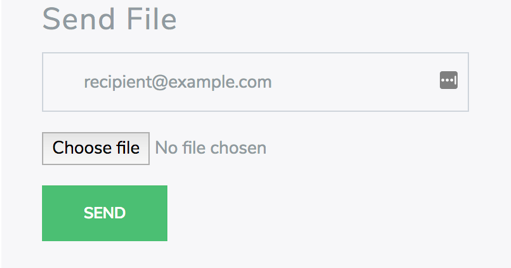
De encryptie van de private key gebeurt a.d.h.v. de cryptico library functie encryptAESCBC() die als parameters de private-key en de symmetric-key neemt:



private key geëncrypteerd met symmetric key (hash-password)

Alles is klaar en de upload zal gebeuren naar de Firebase Database.

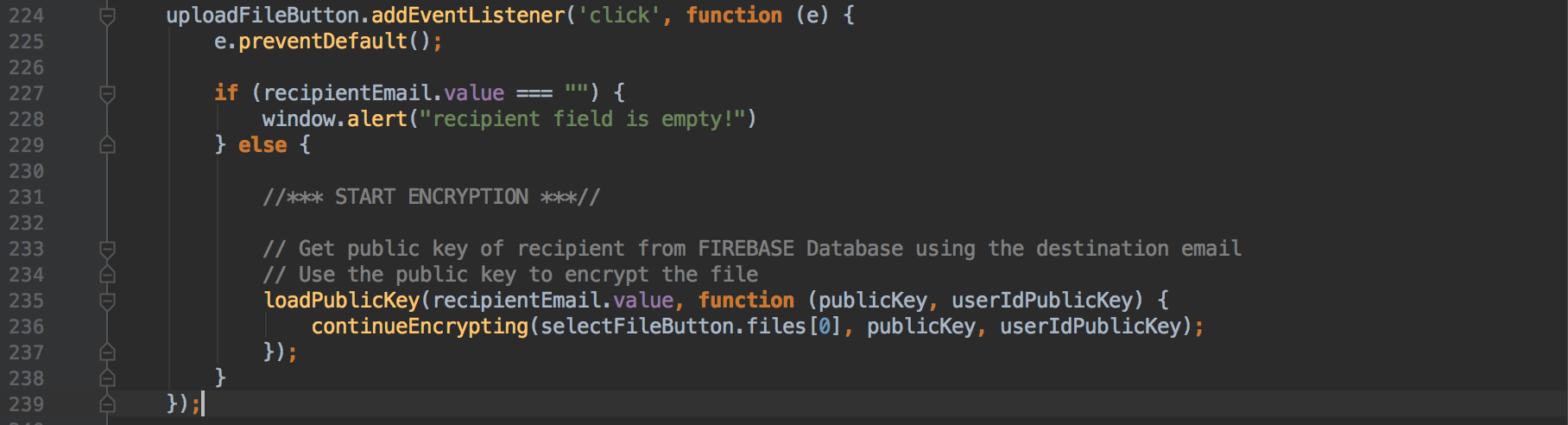
1. File encrypteren, symmetric key encrypteren, file hashing



* De encriptie van de file (ook symmetric key en file hassing) gebeurt als de user op de “send” knop drukt:

**FILE ENCRYPTIE**

Om iets te kunnen encrypteren zodat de ontvanger de file kunt lezen, moeten we zijn public-key (die op Internet staat, in dit geval op Firebase) gaan halen. We doen dat aan de hand van de email van de ontvanger en de functie loadPublicKey() die de public-key Object teruggeeft en de userID:



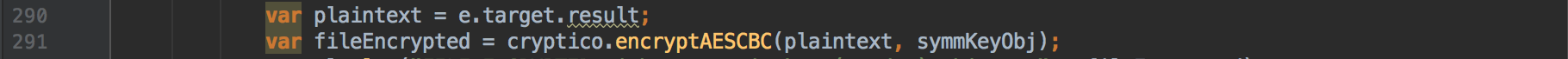
Met de public-key object en de userId kunnen we verder met de encryptie proces a.d.h.v. de functie continueEncrypting() die als parameters heeft de file, de public-key van de ontvanger en zijn userId.

Nu dat we de ontvanger public-key, zijn userId en de file hebben, kunnen we de file encrypteren:

* We genereren een pseudo-random symmetric-key om de file contents te kunnen encrypteren. We gebruiken de library functie generateAESKey() die een object teruggeeft:



* We plaatsen de contents van de file in a variable “plaintext”.
* De encryptie van de plaintext gebeurt a.d.h.v. de cryptico library functie encryptAESCBC() die als parameters de plaintext en de symmetric-key neemt:



file is geëncrypteerd!!!

**SYMMETRIC KEY ENCRYPTIE**

**(antwoord vraag:** “*De ontvanger zeker is dat hij de enige is die de file terug kan decrypteren“)*

* We gaan de symmetric-key encrypteren met de public-key van de recipient (de public-key object heeft een functie encrypt() om dit te kunnen doen), en deze encryptie nog langer maken (base64).
* Omdat we de public-key van de ontvanger gebruiken, alleen de ontvanger zal de symmetric-key kunnen decrypteren (met zijn private-key).



symmetric-key is geëncrypteerd!!!

**FILE HASHING**

* De functie SHA256() maakt een hash van de plaintext



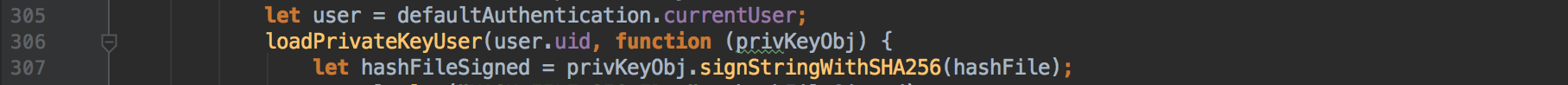
file hash !!!

**FILE HASH SIGNEREN**

**(aantwoord vraag: “***De ontvanger zeker is dat de file van A afkomstig is***“)**

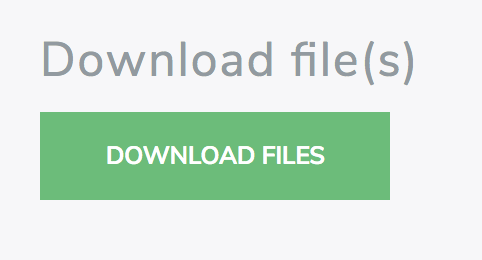
**(aantwoord vraag: “***De zender niet meer kan ontkennen dat hij de file gestuurd heeft***”)**

* Een file signeren gebruikt met de private-key van de gebruiker.
* Firebase gaat de current user ID gebruiken om de private-key (encrypted) vanaf de Database te gaan halen via de loadPrivateKeyUSer() functie.
* Met de private key (gedecrypteerd aan de client kant) krijgen we weer toegaan naar de functies van de private-key, en ëën van deze functies is signStringWithSHA256() die als parameter de hash file gebruikt:

****

file hash is gesigneerd !!!

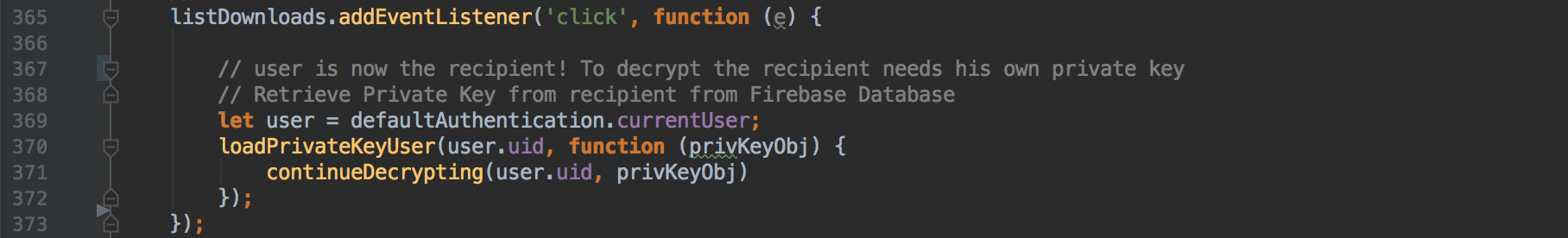
1. File decrypteren



* De decryptie van de file (ook symmetric key en file hassing) gebeurt als de user op de “download files” knop drukt:

**FILE DECRYPTIE**

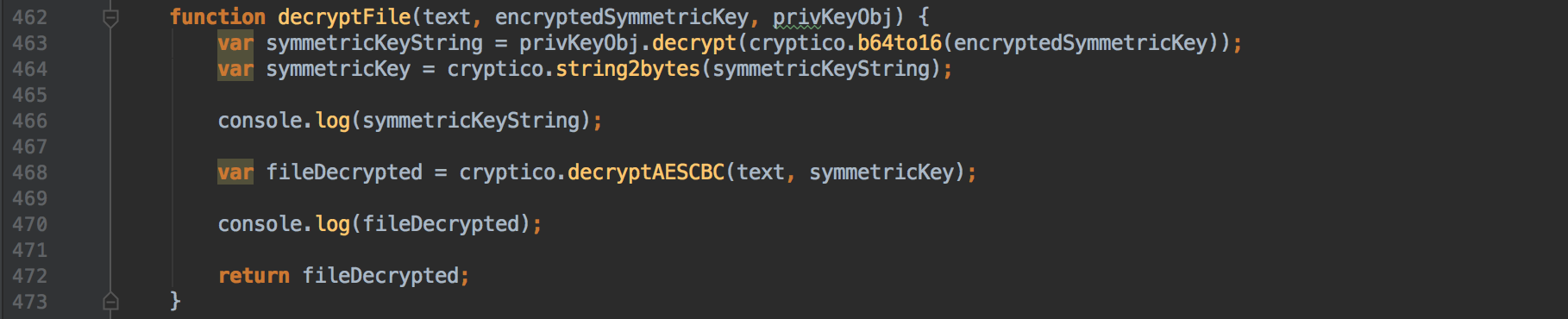
* Om iets te kunnen decrypteren moet de ontvanger gebruik maken van zijn private-key (in dit geval staat de private-key op Firebase, maar geëncrypteerd!).
* We halen de private-key aan de hand van de functie loadPrivateKeyUser() die de user ID gebruikt (deze info is doorgegeven naar de functie via Firebase) om de key te gaan halen uit de Database:



* Met de private-key object kunnen we verder gaan met de decryptie proces via de functie continueDecrypting().
* Deze functie gaat de data uit de Firebase Database halen en gaat de tekst, de file symmetric-key en de private-key gebruiken om de file te kunnen decrypteren. We gebruiken de functie decryptFile() ervoor:



* Werking van de functie decryptFile():
  + Decrypt symmetric key:
    - De symmetric key van de file was geëncrypteerd met de public-key van de ontvanger (De ontvanger is de enige die de symmetric key kunt decrypteren, omdat hij de enige is die zijn eigen private-key heeft).
    - We gebruiken de functie van de private-key obejct “decrypt()” om de symmetric key te kunnen decripteren.
  + Decrypt file:
    - We gebruiken de cryptico library functie “decryptAESCBC()” die als argumenten de tekst en de symmetric key gebruikt.
    - Deze functie geeft terug de file decrypted!



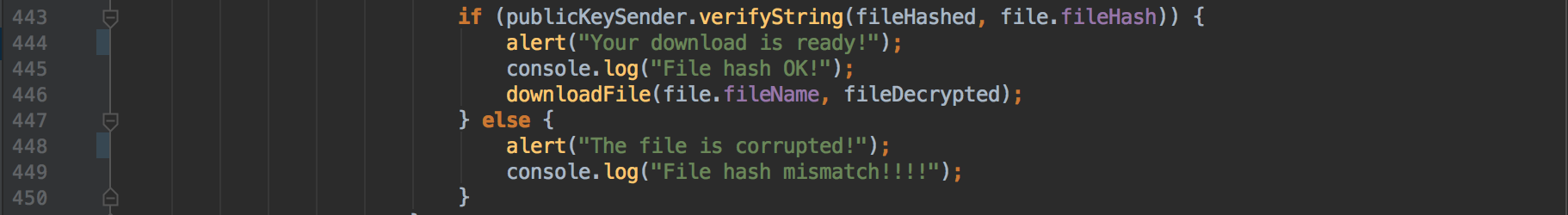
file gedecrypteerd !!!

* We moeten nue verifieren dat nieks is veranderd in de file. We gaan een hashing doen van de gedecrypteerde file. We gebruiken de cryptico functie SHA256():



file gedecrypteerd hashed !!!

* We verifieren dat als de hash file van de sender en de file hash van de ontvanger dezelfde zijn, dan kunnen we de file downloaden, anders gaan we alleen maar een alert bericht laten zien:
* We doen dit aan de hand van de criptico library functie verifyString() van de publi-key object:



* Deze laatste step aantwoordt de vraag “*De ontvanger zeker is dat er niets veranderd is aan de file tijdens transport*”.