# **Securing API data via aes\_encryption/aes\_decryption using secret key**

**Step 1**

**Web Service api call to save user sensitive data:**

Need to config a secret key (like 'xxxxxxx') on both mobile and web end for secure encryption and decryption of parameters.

Send all the parameters from mobile app by aes encryption process for secure connection with web api end point like below:

**http://www.xxxxx.com/api/v1/save\_data?auth="ABF%$JH78787878ABF%$JH78787878ABF%$JH78787878ABF% $JH78787878ABF%$JH78787878ABF%$JH78787878ABF%$JH78787878ABF%$JH78787878ABF%$JH78787878ABF%$JH78787878ABF%$JH78787878ABF%$JH78787878"**

Auth will contain all the parameters in encryption format with the combination of secret key,

**Step 2**

Create a file name **key.rb** in **config**/**initializers** folder and set the same key that have used in mobile app for encryption.

**Config/intializers/key.rb**

**PRIVATE\_KEY="xxxxxxxxxxx"**

**Step 3**

In application controller write down the following code to decrypt the data coming from the mobile app api call.

**App/controllers/application\_controller.rb**

***# Check user with valid authenticity token to access API***

***def restrict\_access***

***params = prepare\_params***

***@params = params***

***if !params.has\_key?(:auth\_token) && !params.has\_key?(:access\_code)***

***error\_response(LOGIN\_FAILED, 'Need authenticity token and/or access code')***

***elsif params[:auth\_token].present? && params[:access\_code].present?***

***If (@participant=Participant.includes(:user).where("participants.access\_code =? and participants.auth\_token=? and participants.approved =?", params[:access\_code], params[:auth\_token], true).first).present?***

***MakeUserAvailableViaThread.user\_in\_thread = @participant.user***

***return @participant***

***else***

***error\_response(INVALID\_ACCESS\_TOKEN, 'Invalid authenticity token/access code')***

***end***

***else***

***error\_response(INVALID\_ACCESS\_TOKEN, 'Invalid authenticity token/access code')***

***end***

***end # End of method***

***# preparing the params in array structure***

***def prepare\_params***

***param\_array = decrypt\_API\_params***

***params = {}***

***param\_array.each do |p|***

***puts "@@@@@@@@@@@@@@@@@@@@@!!!!!!!!!!!!!!!!"***

***key, value = p.split("=")***

***params[key.to\_sym] = value***

***end***

***decrypt\_log.info "Params after preparing it"***

***decrypt\_log.debug params***

***return params***

***end # End of method***

***# Setting up the paramters for decryption process***

***def decrypt\_API\_params***

***parameters = {}***

***decrypt\_log.info "encrypted params[:auth].present?"***

***decrypt\_log.debug params[:auth].present?***

***decrypt\_log.info " Encrypted of params[:auth] is "***

***decrypt\_log.debug params[:auth]***

***if params[:auth].present? && (encrypted\_data = params[:auth].gsub(' ', '+')).present?***

***parameters = aes\_decryption(encrypted\_data)***

***decrypt\_log.info "Decrypted value of parameters are "***

***#decrypt\_log.debug parameters = parameters.split("--separator--").first.split("&")***

***decrypt\_log.debug parameters = parameters.split("&")***

***end***

***return parameters***

***end # End of method***

***# Decrypting the paramters by using the PRIVATE KEY***

***def aes\_decryption(encrypted\_data)***

***encrypted\_parameters, publickey\_encrypted\_date = AESCrypt.decrypt(encrypted\_data, PRIVATE\_KEY).split("--separator--")***

***decrypt\_log.info "Decrypted by private key"***

***decrypt\_log.info "value of encrypted parameters is "***

***decrypt\_log.debug encrypted\_parameters***

***decrypt\_log.info "value of encrypted publikey and date is "***

***decrypt\_log.debug publickey\_encrypted\_date***

***publickey, date = AESCrypt.decrypt(publickey\_encrypted\_date, PRIVATE\_KEY).split("--separator--")***

***decrypt\_log.info "Decrypted value of public key is "***

***decrypt\_log.debug publickey***

***decrypt\_log.info "Decrypted value of date is "***

***decrypt\_log.debug date***

***@publickey = publickey unless @publickey***

***parameters = AESCrypt.decrypt(encrypted\_parameters, publickey)***

***end# End of method***

Now all the data are decrypted securely through aes\_decryption process using a secret key from web section.