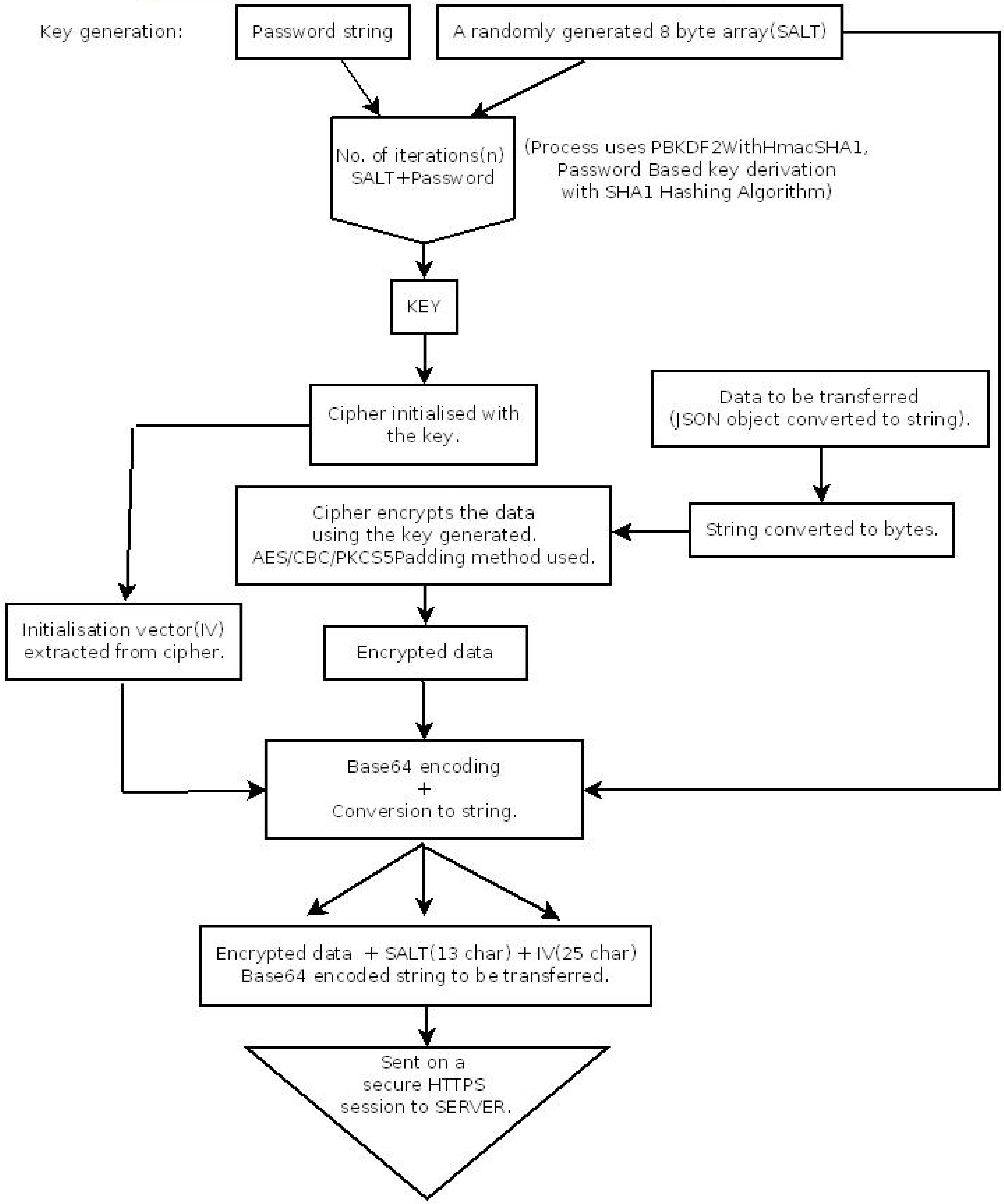
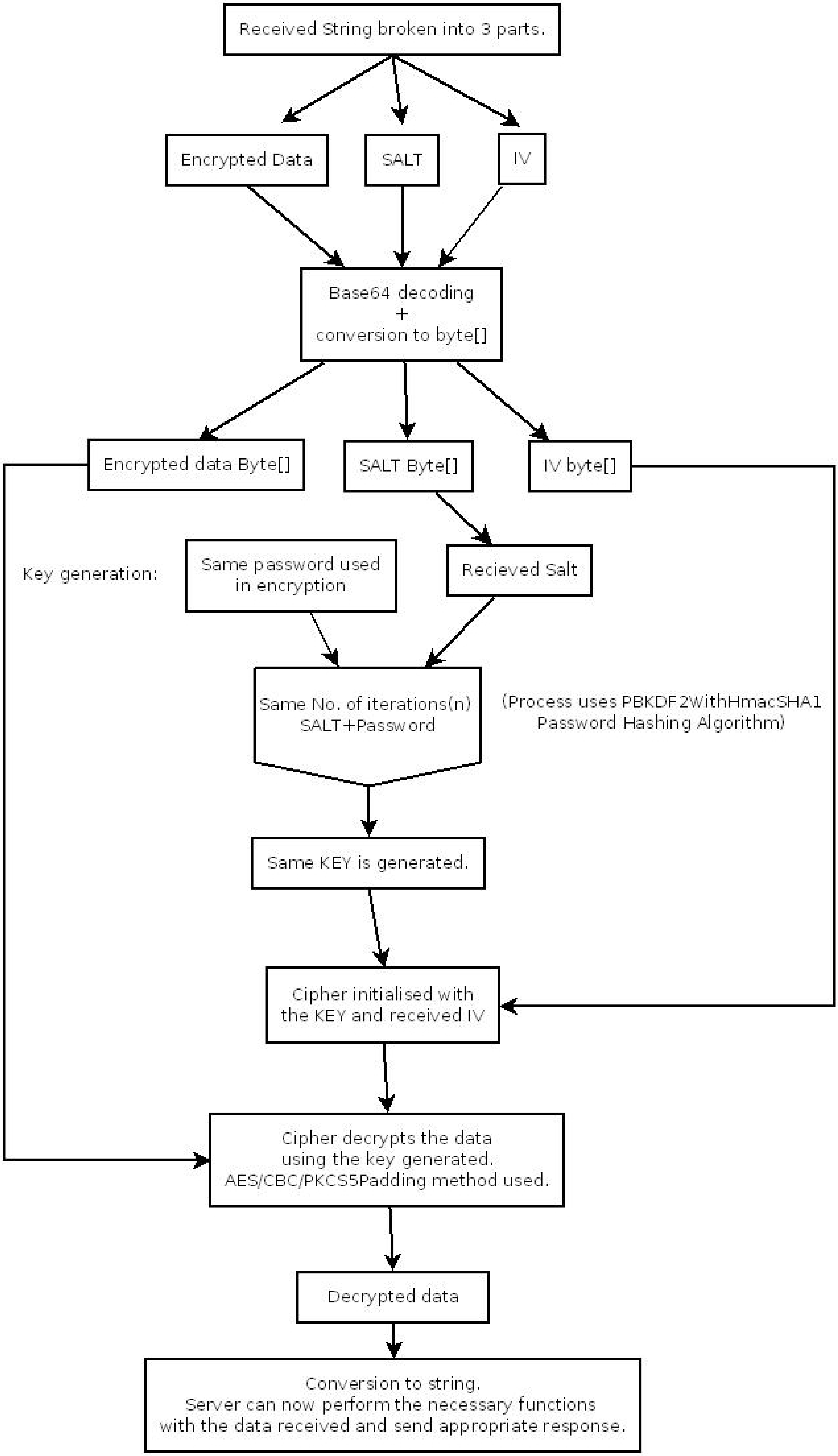
1. Encryption on mobile client side(AES-256 bit encryption).


Key generation:
Password string
A randomly generated 8 byte array(SALT)
No. of iterations(n)
SALT+Password
(Process uses PBKDF2WithHmacSHA1,
Password Based key derivation
with SHA1 Hashing Algorithm)
KEY
Cipher initialised with the key.
Data to be transferred
(JSON object converted to string).
Cipher encrypts the data
using the key generated.
AES/CBC/PKCS5Padding method used.
String converted to bytes.
Initialisation vector(IV)
extracted from cipher.
Encrypted data
Base64 encoding
+
Conversion to string.
Encrypted data + SALT(13 char) + IV(25 char)
Base64 encoded string to be transferred.
Sent on a
secure HTTPS
session to SERVER.


Received String broken into 3 parts.
Encrypted Data
SALT
IV
Base64 decoding
+
conversion to byte[]
Encrypted data Byte[]
SALT Byte[]
IV byte[]
Key generation:
Same password used in encryption
Recieved Salt
Same No. of iterations(n) SALT+Password
(Process uses PBKDF2WithHmacSHA1 Password Hashing Algorithm)
Same KEY is generated.
Cipher initialised with the KEY and received IV
Cipher decrypts the data using the key generated. AES/CBC/PKCS5Padding method used.
Decrypted data
Conversion to string. Server can now perform the necessary functions with the data received and send appropriate response.

AES/CBC/PKCS5Padding explained: AES is in the Cipher Block Chaining cipher mode, with padding defined in PKCS#5.

-This algorithm accepts keys of 128, 192, or 256 bits(256 in this case).

-CBC is a cipher mode where each block of plaintextis combined (through XOR) with the previous (encrypted) block before encrypting, and the first block is combined (through XOR) with a so-called initialization vector (or IV) before encrypting.

-In the Java implementation, a random IV is generated (with the IvParameterSpec class) and placed at the beginning of the cipher text.

-Same format to be followed for server - mobile app communication.

-A generated key will be valid for one session only.

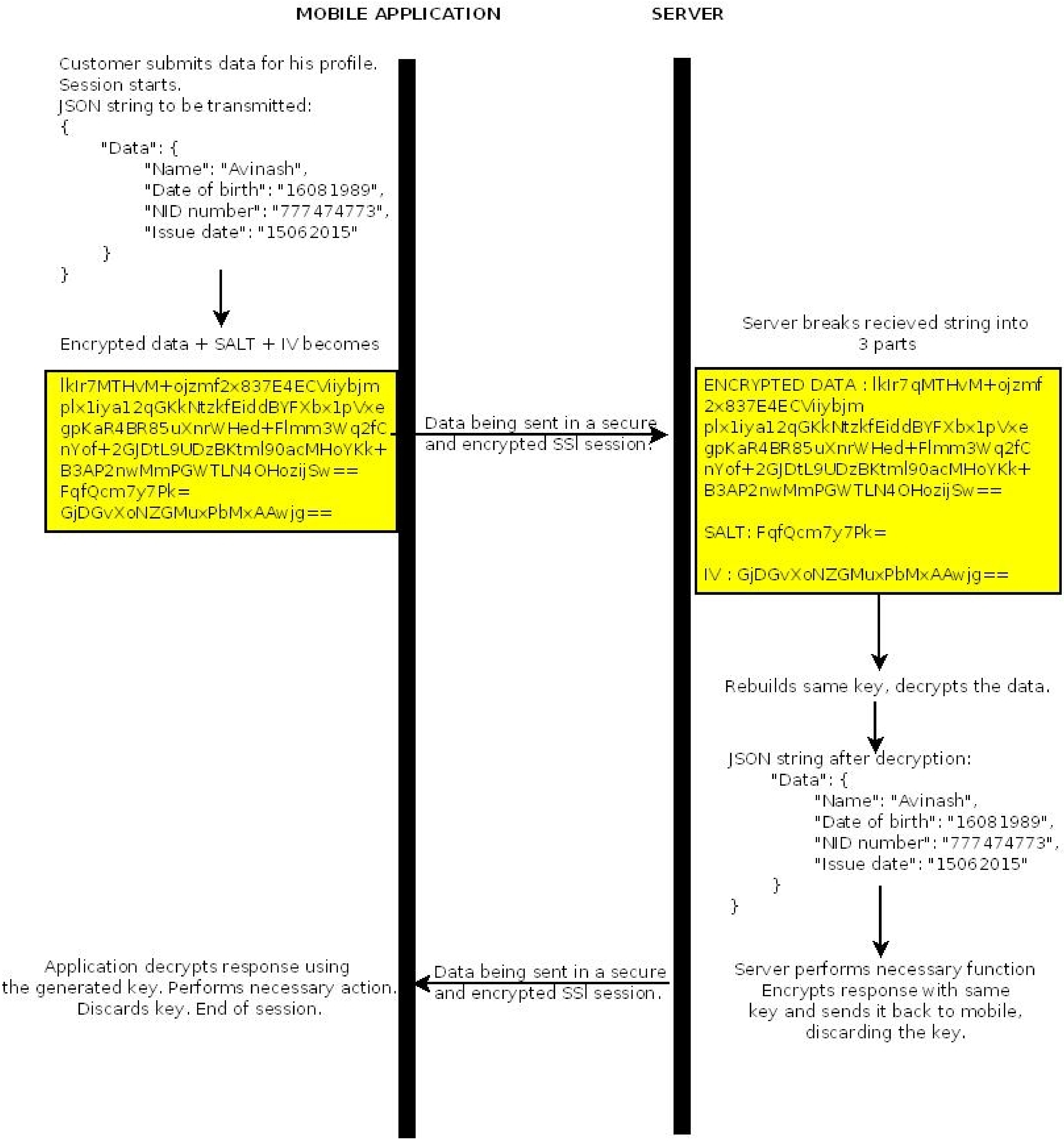
-The server and application would have the same password and iterations number (to be kept after discussion).

-Iterations should be greater than 1000 but to an extent where it doesnt slow down the processing. A higher number makes the system more secure but anything above 1000 is considered good.
Source- http://www.rfc-editor.org/rfc/rfc2898.txt (section 4.2)

-A different salt for each key and the HTTPS encryption makes the system very robust against any kind of attack specially a brute force attack which is done using a hash table of passwords and trying for a match.

-Password can be kept anything even a blank string, and will have the option of being remotely changed on both server and application in case its compromised.

A working example of the above model.(Server is stimulated in a different class)



MOBILE APPLICATION

SERVER

Customer submits data for his profile.
Session starts.
JSON string to be transmitted:

```
{
      "Data": {
            "Name": "Avinash",
            "Date of birth": "16081989",
            "NID number": "777474773",
            "Issue date": "15062015"
      }
}
```

Encrypted data + SALT + IV becomes

```
lkIr7MTHvM+ojzmf2x837E4ECViiybjm
plx1iya12qGKkNtzkfEiddBYFXbx1pVxe
gpKaR4BR85uXnrWHed+Flmm3Wq2fC
nYof+2GJDtL9UDzBKtml90acMHoYKk+
B3AP2nwMmPGWTLN4OHozijSw==
FqfQcm7y7Pk=
GjDGvXoNZGMuxPbMxAAwjg==
```

Data being sent in a secure
and encrypted SSl session.

Server breaks recieved string into
3 parts

```
ENCRYPTED DATA : lkIr7qMTHvM+ojzmf
2x837E4ECViiybjm
plx1iya12qGKkNtzkfEiddBYFXbx1pVxe
gpKaR4BR85uXnrWHed+Flmm3Wq2fC
nYof+2GJDtL9UDzBKtml90acMHoYKk+
B3AP2nwMmPGWTLN4OHozijSw==

SALT: FqfQcm7y7Pk=

IV : GjDGvXoNZGMuxPbMxAAwjg==
```

Rebuilds same key, decrypts the data.

JSON string after decryption:

```
      "Data": {
            "Name": "Avinash",
            "Date of birth": "16081989",
            "NID number": "777474773",
            "Issue date": "15062015"
      }
}
```

Server performs necessary function
Encrypts response with same
key and sends it back to mobile,
discarding the key.

Data being sent in a secure
and encrypted SSl session.

Application decrypts response using
the generated key. Performs necessary action.
Discards key. End of session.