WIZZIT
AUTHENTICATOR

WIZZIT
AUTHENTICATOR

AUTHENTICATOR SECURITY WHITE PAPER

AUTHENTICATOR OVERVIEW

The use of instant messaging platforms such, but not limited to, Facebook messenger, WhatsApp, Telegram and WeChat is increasing. Banking and transactional capability for these applications is emerging.

Payments on these platforms cannot trust the inherent security of the Chat platform and need an additional layer of security through a Pin Entry Module that will allow for the encryption of any data entered.

The main validation reason for implementing an additional encryption layer over and above normal SSL transport level encryption would be to provide end to end encryption of data such as a PIN or Password. This is even used inside the server application backend components with the PIN or Password encrypted at all times for added security.

The current solutions of encryption, either requires that an App has a pre-shared key installed at the point of downloading (As done by or in the case of WIG that a key be loaded onto the SIM card. Current app security solutions use either an additional app or downloaded keyboard. There use traditional PKI methods to encrypt the data and can do so because the pre-shared key is contained in this application.

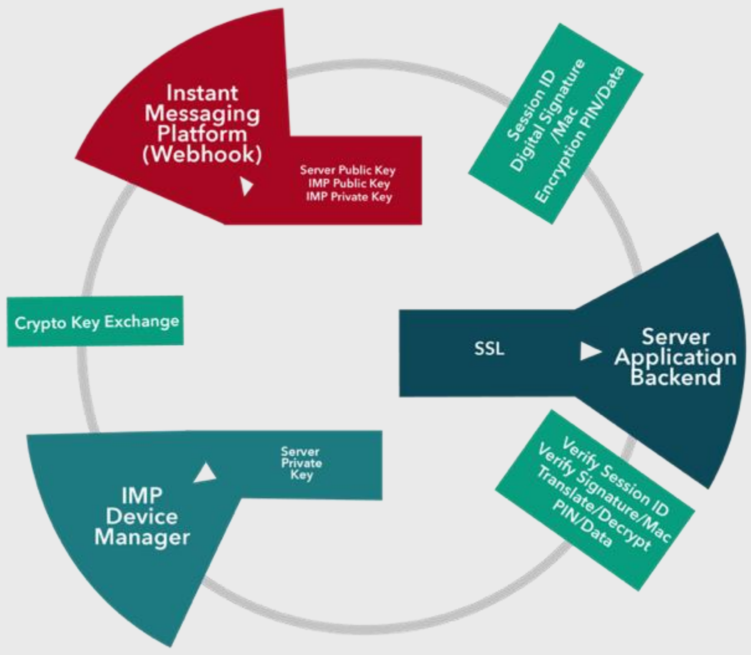
The unique method proposed is not to have to pre-share any key but to negotiate a shared secret between the server and Client using a NIST and FIPS compliant Diffie-Hellman elliptic curve. In order to do this the server needs to trust the client before the negotiation of the shared secret can commence. This is done using a comparison of encrypted values between the client and the server. The secret is shared with the IM provider and not visible to the client.

Once this is now confirmed the public key can now be shared with the client. Based on this standard encryption can now commence.

ARCHITECTURE

The architecture of the securitization supports the following model:

- Instant messaging platform with a Web hook request to the server-side application backend.
- Direct integration model between the Instant messaging application and the sever application backend.
- Please note that the server-side application will adhere to all PCI DSS requirements, which fall outside the scope of this document.



APPLICATION INITIALISATION

The IMP will do some initialization steps on first start-up to generate or derive a unique device identifier and generate any mobile key pairs required for key exchange.

The IMP is initialized with a 32byte pre-shared key, this key will be stored securely on the server side and must not be shared. This key is used to generate a SHA1 HMAC signature for each request, consisting of the JSON body utilizing the 32-byte pre-shared key. The application will on receipt of the request generate the same HMAC, and if the hash values compare will then continue with session key exchange process as discussed below.

APPLICATION INITIALISATION (WEBVIEW)

The key exchange between the IMP and the server device manager ensures that a secure session key is exchanged that can be used for the encryption of sensitive data.

The key exchange consists of 2 phases, first the Diffie-Hellman key exchange followed by a KEK generated by an HSM exchange. This KEK will be the actual encryption key of sensitive data.

DIFFIE-HELLMAN KEY EXCHANGE (IMP) CLIENT SIDE

1. IMP generates an ECDH (Elliptic Curve Diffie-Hellman: NamedCurve P-256) key pair, it is important that this key pair is not stored beyond the actual exchange agreement.
2. The IMP (WebView) is rendered with the server public key (Elliptic Curve Diffie-Hellman: NamedCurve P-256).
3. The IMP will derive a Shared Secret using IMP-Priv Key and Server Pub Key.
4. A SHA256 cipher will be generated from the derived Shared Secret and used as an encryption key for the duration of that specific session.
5. The IMP will receive a KEK encrypted under the SHA256 cipher from server application.
6. The IMP will decrypt the encrypted KEK using an RSA256 algorithm.
7. On data entry the value will be encrypted with 3DES CBC encryption algorithm, using the decrypted KEK.
8. This result generated in step 7 will then be encrypted (RSA256) with cipher calculated in step 4 and submitted to the server with the IMP-Public key.
9. All keys will be cleared from memory.

DIFFIE-HELLMAN KEY EXCHANGE (SERVER)

1. Application server generates an ECDH (Elliptic Curve Diffie-Hellman: NamedCurve P-256) key pair, the public key will be shared with the IMP on WebView render. It's important to note that for each new rendering of the WebView a new key pair will be generated.
2. On each WebView request a SPID (IMP Unique Identifier) will be stored with corresponding IMP-Public key and Server-side Private key for the duration of that session.
3. The application will derive the Shared Secret from keys as per step 2.
4. A SHA256 cipher will be generated from the derived key from step 3.
5. The Application request a new KEK form the HSM, this key will be stored securely for a period of 24 hours or N amount of transactions.
6. The KEK key will be encrypted (RSA256) with cipher from step 4 and send to the IMP.

SECURITY THREATS AND MITIGATION

Phishing

A big threat is persons with malicious intend pushing fraudulent messages to handsets pretending to be valid WIZZIT PIN requests. The customer will not be pushed any PIN requests but enter these on the APP only.

Man-in-the-middle attack

The aim of this attack is to convince a customer, called Alice, to use a rogue WIZZIT mobile application. The rogue application, similar in look-and-feel to the real WIZZIT mobile application, talks to the 'man-in-the-middle' application (operated by Eve), and in turn the 'man-in-the-middle' application talks to the EVERET server. The 'man-in-the-middle' application emulates the real WIZZIT mobile application's interface to the WIZZIT server.

Every time Alice establishes a session to Eve, Eve establishes her own session to the WIZZIT server. Eve decrypts all information received from Alice with the session key she shares with Alice, and then re-encrypts the information with the session key she shares with the WIZZIT server. Eve follows a similar process when receiving information from the WIZZIT server to pass on to Alice.

The WIZZIT system is immune to this type of attack. An out-of-band authentication message is needed to authenticate Alice. Eve can persuade Alice to send the out-of-band message, but the message will be signed with a session key only known to Alice and Eve (and not the WIZZIT server), and for that reason the server will not accept the registration attempt.

Internal threats

The current WIZZIT system is PCI-DSS Level 1 ready. Account information is stored against a linked MSISDN, but this Information is kept in an encrypted format inside a database with restricted access.

No database or other system passwords are stored in the clear on the system.

SYSTEM REVIEW AND RISK ANALYSIS

This review is to identify any vulnerabilities and weaknesses of network or systems. It will focus on operating system, administration and monitoring tools on different platforms.

Risk Analysis helps to determine the value of the assets and their associated risks.

In general, this process is divided into several sub-processes.

They are:

- Asset Identification and Valuation
- Threat Analysis (continuously revised)
    - Human errors
    - Disgruntled employees
    - Malicious or careless personnel
    - Misuse of systems and computer resources
    - Computer fraud
    - Theft
    - Industrial espionage
    - Environmental disasters
- Vulnerability Analysis
    - Vulnerability Scanning
    - Penetration Testing
- Asset/Threat/Vulnerability Mapping
- Impact and Likelihood Assessment
- Risk Results Analysis – using Qualitative & Quantitative Methods, and Matrix Approach.

CONCLUSION

The WIZZIT system has been designed to accommodate the banking industries requirements for Security and risk management. We comply with the latest standards and are constantly updating the system with new security features. If there are any additional regional or banking requirements we will accommodate these.

WIZZIT INTERNATIONAL

- Dirkb@wizzit-int.com
- Davep@wizzit-int.com
- Charlesr@wizzit-int.com
- Brianr@wizzit-int.com

Europe

- Gideonv@wizzit-int.com

Australasia

- Nicholasr@wizzit-int.com

London

- Simonellis89@icloud.com