Volume

1

QUIKSCRIBE PTY LTD

Whitepaper Draft

Cryptography System

QUIKSCRIBE PTY LTD

Quikscribe Cryptography System Whitepaper

Table of Contents

Quikscribe Cryptography Whitepaper Draft

When data is sent across corporate networks (LANs and WANs) and the internet, It is vulnerable to being intercepted and changed or intercepted and misused by malicious computer / network operators when sent in its plain form, as all data sent over networks in the TCP/IP network protocol is simply converted to hexadecimal before being written into the data packet to be sent.

This means a user could potentially use some form of network packet sniffer and assemble all fragments of a file being transported over the internet into the complete file and open / view the file using the appropriate software.

This may not be a major concern if the data being transferred is of a non sensitive nature, but if the data is of a sensitive nature then security precautions need to be taken.

The most effective form of security for data is encryption, either as the data is being streamed across the network to the remote host(s) or while the file is sitting on some form of storage before being transferred.

To secure IAF files for transport, the most effective and resource efficient manner is to simply encrypt the files while they are sitting on storage, which eliminates having to setup special SSL server infrastructure to deal with transporting data from one point to another securely.

This whitepaper details a proposed file based security system that uses a combination of both symmetric and asymmetric encryption using digital certificates and pass phrases to achieve security.

The proposed system would use the current industry standard algorithms:

- RSA 1024 Bit RC4 Stream (Symmetrical Cipher)
- RSA 1024 Bit Public Key Cryptosystem (Asymmetric Cipher)
- Secure Hash Algorithm (SHA-1)

The RSA Public Key Cryptosystem chosen is IEEE P1363 (Standards for Public Key Encryption) compliant and subsequently conforms to industry standards for Public / Private key Cryptography systems.

SHA-1 is defined under NIST FIPS PUB 180-1 as the standard for digital signature hashing.

Proposed Cryptographic System Solution:

The proposed solution to this cryptography requirement is a file based security system which uses a Public Key Infrastructure (PKI) to allow secure data interchange by means of cryptography between users of a Quikscribe IAF file based system, using a combination of symmetric and asymmetric cryptography to secure data.

Each user in the system would be issued with a digital certificate containing a unique private key that identifies them within the system. This certificate also contains a public key, which other users can use to encrypt data before being sent to the owner of the corresponding private key as only the corresponding private key can decrypt the data encrypted using their public key.

Instead of encrypting data as it is in transit, the IAF file is encrypted and stored within a "carrier" file, that contains all the information about the original file and the data required to decrypt it so that it can be viewed or transcribed.

This means that at all times, all required information to decrypt the file is stored within the file and the file can still be easily identified and the file is still secure even when still stored on some form of disk based storage.

The "carrier" file will also contain a digital signature so that it can be verified that the contents of the file have not been tampered with and that the file is actually from the author it claims to be from.

The rest of this whitepaper outlines how the proposed system would function.

Public Key Infrastructure Setup:

The encryption system would use a Public / Private Key Pair system to provide dynamic encryption keys every time a file is encrypted.

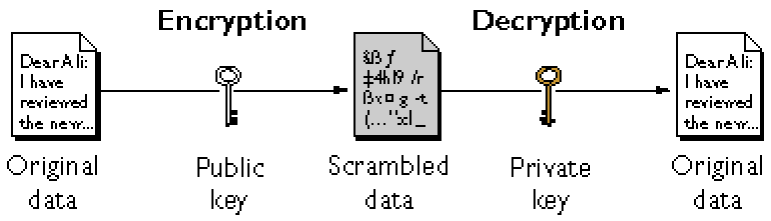

Figure 1 - Public Key (Asymmetric) Cryptographic Process

Each user would be issued with an X.509 Style password protected “certificate” that contains:

- Authority who issued certificate
- Certificate version
- Serial Number
- Name
- Company
- Position
- Validity Start and End Date Time
- Private Key
- Public Key
- Overall Hash Checksum

Authority who issued certificate is the actual organization and person within the organization who issue the certificate to the user.

Certificate Version identifies the internal format version of the certificates internal data format and layout.

The Serial Number is the unique serial number of the certificate that is used to uniquely identify it among other certificates that have been issued.

Name is the Name of the Individual that the Certificate was issued to, used to identify who owns a certificate.

Company is the company that the named individual is employed by and has had the certificate issued to by.

Position is the individual’s position within the organization that issued the certificate.

Validity Start and End contains the date time that the certificate is valid from and the point in that the certificate is valid until.

The private key is used to decrypt information sent to a specific user, as the random encryption key used to encrypt the data would be encrypted using that user's public key and stored within the "carrier" file before being sent.

When this key is decrypted using the correct private key, it allows the intended recipient of the file to decrypt and view the contents of the file.

The public key is used to encrypt information before being sent to a specific user, specifically the random key used when encrypting the IAF data before it is attached to the carrier IAF file. The public key encryption process is a one way process; the data cannot be decrypted using anything but the private key from the key pair.

The overall hash checksum is a signature that is calculated based on all the data within the file, to allow integrity checking so that changes cannot be made to certificates after they have been issued to a client.

Certificate Internal File Layout:

This is the format of a certificate file that would be issued to a user when they are first issued with a certificate to use encryption.

```
-----BEGIN CERTIFICATE-----
Certificate:
    Data:
        Version: v1 (0x0)
        Serial Number: 3 (0x3)
        Signature Algorithm: PKCS #1 MD5 With RSA Encryption
        Issuer: OU=Quikscribe Certificate Authority, O=Quikscribe Pty Ltd, C=AU
        Validity:
            Not Before: Thursday Aug 1 08:00:00 2002
            Not After: Thursday Aug 1 08:00:00 2003
        Subject: CN=Joshua McAdam, OU=Development, O=Quikscribe Pty Ltd, C=AU
        Subject Public Key Info:
            Public Key:
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAJ1Ojvd2YVMtOzK0oFByAYAF+AvwuvMr
T6DM7KChvpARAgMBAAE=
-----END PUBLIC KEY-----

        Subject Private Key Info:
           Private Key:
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,A2CD5635C0499A34

1ib+VjL8rTA2kheubHFu7l+/ZiajB8CS6ma7ezIi8Y4K3iWidOjp1dlzeEGo72sN
l8+3mmJvv1pySDPcHZhU6uyFTCmVazqsqEmqMG5gFJEWUNeQhy5tJkkH39AQR6Gm
NHrbOZBVj1ddjg4oJtcrLAYldOf2nqYFdh/JYstj4WOvo7GFawueXmZMfNPi4Qtx
75x2KsJn9vHQsMxQGN4313xWuvBF1d8URv0retHjn+U=
-----END RSA PRIVATE KEY-----

        Extensions:
            Identifier: Certificate Type
                Critical: no
                Certified Usage:
                    Quikscribe File Encryption

    Signature:
        Algorithm: PKCS #1 MD5 With RSA Encryption
        Signature:
            6d23aff3d3b67adf90dfcd7e186c01698e5465fc06
            304334d1631f067dc340a82a82c1a4832afb2e8ffb
            f06dff75a378f752474662971dd9c6110a02a2e0cc
            2a756c8bb69b87007d7c847679baf8b4d26258c3c5
            b6c143ac634442fdafc80f2f38856dd659e84142a5
            4ae52638ff3278a138f1eddc0d31d1b06d67e946a8
            ddc4

-----END CERTIFICATE-----
```

The certificate is X.509 standard and contains all necessary data for them to begin exchanging encrypted data with another user who has been provided with an appropriate public key.

A private key backup file would have the same format, with the private key password protected to ensure its security and integrity.

A public key export file would be the same, but with the private key section removed so that only the owner of the private key can decrypt files encrypted with their public key and the key is not compromised.

Certificate (Key) Issue / Management:

Certificates and their component key pairs would be issued by the network administrator to the users who required them, to prevent Trojan self signed certificates from being used to attempt security breaches within the system.

The administrator would use a Certificate generation Application that sets all attributes contained within the certificate file.

The certificate generation app would have the following functionality:

- Issue New Certificate(s)
  - Issue New Certificates to users that have not had a certificate issued before, generate new key pair and certificate file
- Generate Key Pairs
  - Reissue certificate with new key pair in the event of a private key being compromised
- Regenerate Public Key from Specified private key
  - Regenerate a public key from a users private key in case of public key loss
- Reissue Certificate with updated Validity Period
  - Reissue a certificate with a new validity period, when a certificate expires and user requires a new certificate to continue using cryptographic functions
- Track Certificate Issue, Revoking and Expiry
  - Track when certificates were issued and to who, track when certificates were revoked and why, track when currently valid issued certificates will expire

The Administrator would be responsible for:

- Issue of new user Certificates
- Reissue of user certificates
- Revocation of certificates as period expires or become unnecessary for reasons

Workstation/Client Certificate (Key) Management & Interchange:

On a workstation with Quikscribe software installed, a “keychain’ would be established. The function of the “keychain” is to store information about all the keys available on that particular workstation and manage import / export of certificate data files.

The “keychain” would contain:

- Users Identification Information
- Users Public Key
- Users Private Key (Password Protected)
- Other Imported Public Keys

The import functionality of the keychain would be used to import certificate files into the keychain so that a user could import their private key and associated data, allowing them to decrypt data sent to them or import other user’s public keys and associated data into the keychain so that they can send encrypted data to other users.

The export functionality of the keychain would be used to export certificate data out of the keychain for purposes of private key backup or public key exchange.

The export functionality for the private key requires a password to ensure that the exported file is properly secured once exported and cannot be reopened without that password. Note: if a private key is lost, it cannot be recovered!

The export functionality for the public key would not require this as it is only able to encrypt information and can be freely distributed. Note: public keys can easily be regenerated from the private key if required.

Another important function of the “keychain” is to ensure that all key pairs and certificates in use are valid. This is achieved by checking the hash checksum stored in the certificate against a calculated checksum and also by checking that the certificate used is within its explicitly specified date and time period of validity.

When a user configures cryptography on their workstation, the following steps would be taken:

- **Quikscribe Recorder:**
    1. Install user Certificate.
    2. Import Public Key(s) for destination typists.
    3. Select Optional File Signing / Verification Options

- **Quikscribe Player:**
    1. Install user certificate
    2. Select Optional File Signing / Verification Options

This configures all required data and files on the client workstations to allow the cryptography system to function as described.

File Encryption Methodology:

The actual IAF file data will be encrypted using the RC4 Stream Symmetrical Cipher. This means that the same key used to encrypt the data is required in order to decrypt the data when it reaches intended recipient(s).

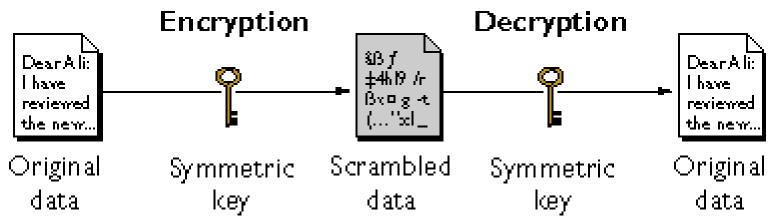

Figure 2 - Symmetrical Cryptographic Process

The IAF file data will be encrypted using a randomly generated 1024 bit key that is internally generated within the Cryptographic Service Provider Module (CSP), so that the cryptographic level of the encryption performed on the data is relatively strong because the key is of an ensured length and composed of completely random data.

Once the key is generated, the IAF data will be encrypted and stored inside a temporary “carrier” IAF file that will act as the transport for the encrypted file until it reaches its destination and it decrypted.

The “carrier” IAF file will contain all the information stored in IAF fields that existed in the original file before it was encrypted allowing the file to still be easily identified and tracked as it passes through various transport mechanisms and file stores.

The “carrier” IAF file would contain the following additional data stored in IAF private fields:

- **Encryption Flag**
  This would flag a private field within the IAF as 1 or 0 to indicate whether the contents of the current IAF file was an encrypted IAF file stored as an attachment

- **Encrypted Key used to encrypt the IAF data**
  This would store the actual random 1024 bit key used in the RC4 encryption process used to encrypt the IAF file before it was inserted into the carrier in its encrypted form.

  This key would be stored in encrypted form, as it would be encrypted using the intended recipient’s public key so that only the intended recipient's private key could decrypt the key and allow decryption of the attached IAF file.

- **Intended Recipient Name**
  This would store the name of the intended recipient in DN (Distinguished Name) format, e.g. CN=Joshua McAdam, OU=Development, O=Quikscribe Pty Ltd, C=AU

  This is done so that the private key that should be used to decrypt the file is identifiable when it reaches the intended recipient.

- **Sender Name**
  This would store the name of the sender in DN format, e.g.
  CN=Joshua McAdam, OU=Development, O=Quikscribe Pty Ltd, C=AU
  This is done so that the sender can be identified and the public key used for response or in digital signature verification can be easily identified and used.

- **Optional Digital Signature Encrypted Hash**
  This would store the SHA1 Thumbprint calculated on the file and then encrypted using the authors private key; this would allow the recipient to verify that the file was not altered during transit and is genuine.

Data Encryption Process:

The Encryption Process would Consist of a 9 Step Process:

1. **Generate Random 1024 Bit Key:**
   A 1024 Bit Character string is internally generated by the system to be used as the encryption key for encrypting the IAF File using the RC4 Stream Cipher.

   This is internally generated to provide a strong level of encryption through prevention of weak keys from being used.

2. **Export IAF Internal Fields to temporary file:**
   The IAF Internal Fields that are stored in the file header are saved to a temporary file so that they can be used later in the encryption process.

3. **Encrypt Contents of IAF File using 1024 bit RC4 Stream Cipher:**
   The whole file is encrypted using RC4 Stream Cipher with the Randomly Generated Key produced in step 1. The RC4 Stream Cipher is used because the output encrypted stream is the same size as the input stream, which is important in the audio application as audio files can be already very large. The stream cipher is also much quicker than using a block cipher when processing large amounts of data.

4. **Create New IAF File:**
   A new “blank” IAF file is created, which will be used as the transport mechanism for the encrypted file until it is decrypted.

5. **Import IAF Fields from temporary file:**
   The fields from the encrypted file that where exported to a temporary file in step 2 are imported into the new IAF file. This is done so that all relevant information about the file is still available even though the content within the file is currently encrypted and unavailable.

   An example of where this information is needed would be if the encrypted file passes through a transcription system running the Quikscribe Manager, which uses the Fields to Display information about workload levels and workflow.

6. **Import Encrypted IAF File as file attachment into new IAF file:**
   The Encrypted file from step 3 is imported into the “carrier” IAF file as a file attachment.

7. **Encrypt the random key used to encrypt IAF contents with RSA Public Key Cryptosystem using Recipients Public Key:**
   The random key generated in step 1 and used to encrypt the data in step 3, is now encrypted using RSA Public Key encryption, using the Recipients Public Key as the encryption key. This is done so that only the intended recipient of the file could decrypt the key using their private key.

8. **Write Encrypted Key into IAF Private Field.**
   The encrypted key value generated in step 7 is written to an IAF private field within the "carrier" IAF File.

   This is done so that all information needed to decrypt the file is within the file at any one time, also when the file reaches its intended recipient, they can decrypt the key and use the key to decrypt the file contents.

9. **Flag New IAF as containing encrypted data using IAF Private Field.**
   The "carrier" IAF file has a private field flag written to it to signify that it is not in fact the actual content, but contains the content in its encrypted form.

Data Decryption Process:

The Decryption Process would Consist of a 4 Step Process:

1. **Extract encrypted IAF file attachment from IAF File to temporary location:**
   The encrypted IAF file attachment is saved to a temporary location on disk for use during decryption.

2. **Use Recipients Private Key to decrypt Encryption Key stored in IAF Private Field:**
   The recipients private key is used to decrypt the file encryption key that is stored in the IAF private field, this key will be used in step 3 to decrypt the IAF file attachment.

3. **Use decrypted key to decrypt contents of extracted IAF File:**
   The encrypted IAF file attachment is now decrypted using the key decrypted in step 3 to its original form.

4. **Save decrypted IAF File to disk and remove temporary and source files:**
   The decrypted IAF file is saved to disk, all temporary files and source files are removed after checksum and cyclic redundancy checks verify that the file is valid, complete and non corrupt.

Digital File Signing Methodology:

Digital File Signing and verification would allow recipients to authenticate that a file was sent by the author that it claims to be from and that the file contents have not been altered in any way during transit.

The digital file signing and verification would be an optional component in the cryptographic system that would use the SHA-1 algorithm to generate hashes of data to be sent for use in digital signatures that would be embedded in an encrypted format in the carrier IAF file.

This signature would then be decrypted at the recipients end and then compared to a hash generated of the received data, to check that the data was not tampered with and that the origin of the file was genuine.

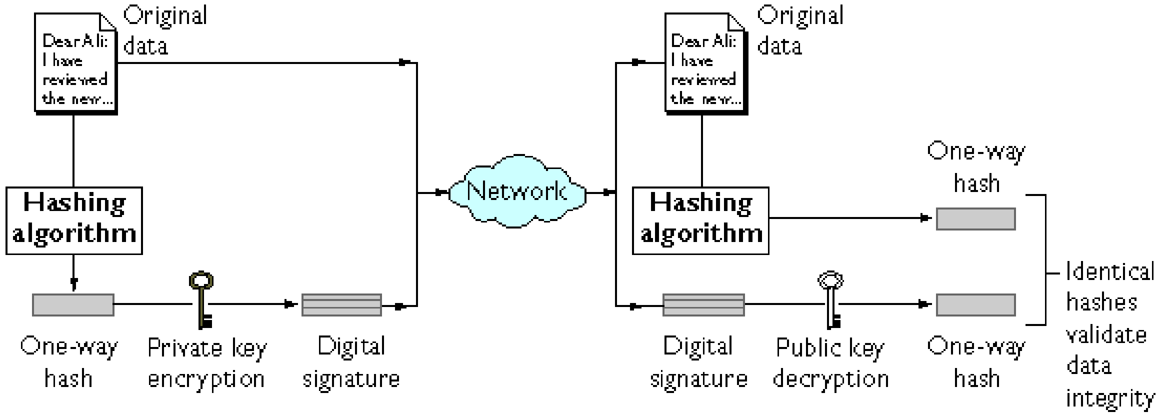

Figure 3 - Digital Signature Creation and Verification Process

Digital Signature Creation Process:

The creation of the digital signature for insertion into the file would be a 6 step process:

1. Export IAF Fields to temporary file
2. Load exported IAF fields from file into memory
3. Append the file length in milliseconds to the IAF fields stored in memory
4. Calculate SHA-1 Hash on IAF field data stored in memory
5. Encrypt calculated hash using RSA Public Key Cryptosystem using Authors Private Key
6. Store encrypted hash ("signature") in private IAF field within "carrier" IAF file.

Digital Signature Verification Process:

The verification process for digital signatures embedded in an IAF file would be a 6 step process:

1. Load encrypted hash from “carrier” IAF private field
2. Decrypt hash using RSA Public Key Cryptosystem using Authors public key
3. Load Fields from Decrypted IAF file into memory
4. Append file length in milliseconds to IAF data in memory
5. Calculate hash on decrypted file IAF data stored in memory
6. Compare hashes for verification

If the file was unaltered and genuine then the hashes would match exactly.