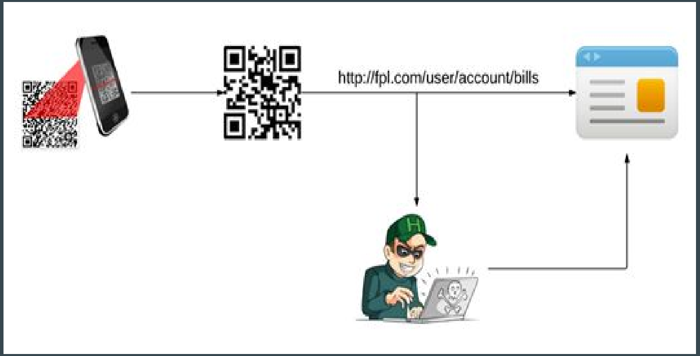
CS428 Course Project: Secure QR Code

Michael Roy
Brandon Bielefeld
Thomas Levine

Overview of Project



- This project was oriented around the use case of secure communication between a financial institution and their customer from the viewpoint of an individual customer receiving a notification.

- In this case the bank must send a notification of repossession to their user.

- To do this, the bank and user both make use of Public and Private keys to ensure that the user knows the bank is the legitimate entity sending the notice and that the user's information is protected.

Threat Model and Defense

- **This system has been designed to handle explicitly defined threats**
  - Entities obtaining the customer's sensitive information.
  - Entities generating QR codes to elicit customer’s sensitive information.
  - Entities tampering with message integrity between bank and customer.
- **Possible Threat Agents**
  - Identity thieves
  - Marketing companies gathering advertisement data
  - Insurance companies assessing risks
  - Inside attacks from bank employees
  - Non-malicious financial snooping by family members or co-workers

Threat Model and Defense (continued)

- **Existing Countermeasures:**
  - Public/Private key encryption:
    - Ensures message confidentiality.
  - Digital Signatures:
    - Ensures message authenticity: The bank sent the message.
    - Ensures message non-repudiation: The bank cannot deny they sent the message.
    - Ensures message integrity: The message contents have not been altered in any way.
  - QR encoding and generation of messages:
    - Convenience for user ensures that the scheme will actually be used.

Threat Model - Limitations of Defense

- This system protects against the security issues raised in the threat model:
  - The message is encrypted with the customer's public key to protect information.
  - The message is signed with the bank’s private key to ensure integrity.
  - The message is verified by customer with bank’s public key to ensure integrity.
  - The message is decrypted with the customer’s secret key to protect information.

- This system is unable to protect against certain threats:
  - Outside/Inside attacks from embedded software in the QR code image.
  - Outside attacks resulting from customer’s carelessness of private key storage.
  - Inside attacks from internal personnel with access to the bank’s private key.

Scheme

Basic Scheme: Asymmetric encryption using two key pairs for encryption and digital signature.

$$\begin{tikzcd}
\mathrm{Sign}_{\mathrm{Bank(private)}}\left(\mathrm{Enc}_{\mathrm{Customer(public)}}(m)\right) \arrow[r] & \text{QR Encode } (c) \arrow[dl] \\
\text{Customer Decode } \left(\text{QR Encode}(c)\right) \arrow[d] & \\
\mathrm{Verify}_{\mathrm{Bank(public)}}\left(\mathrm{Dec}_{\mathrm{Customer(private)}}(c)\right) = m &
\end{tikzcd}$$

Implementation

- **Operating environment**
  - Ubuntu virtual machine provided by "Lab 1" in class
  - GnuPG v1.4.11 (Gnu/Linux)
  - Bing QR code generator
  - Zbarimg (QR code reader and decoder)
- **Scripting**
  - Bash scripts were created for logical groups of operations
    - Script1: Setup key pairs
    - Script2: Encrypt and sign message
    - Script3: Encode encrypted and signed message into QR code, then decode back into encrypted and signed message.
    - Script4: Verify signature and decrypt cipher

Generate Keys (bash script1)

```
gpg --gen-key --batch mikeKeyGenInfo

gpg --gen-key --batch bankKeyGenInfo
```

- GPG allows for the generation of public and private keys together from the information provided in the batch (.txt) file provided to the command.
- The output of these commands are:
  - Customer Private Key File
  - Customer Public Key File
  - Bank Private Key File
  - Bank Public Key File

Encrypt and Sign (bash script2)

```
# encrypt message.txt using public key of Michael. Save to cipher.pem
gpg --output cipher.pem --encrypt --recipient 38E45B2A message.txt

# digitally sign cipher.pem with private key of sender
# -u xxxxxxx designates the private key to be used, --passphrase xxxxx is to enter
the passphrase
# -clearsign preserves the ASCII integrity of the message formatting
gpg -u 9F027B2E --passphrase password --output cipher.txt --clearsign cipher.pem
```

- The input files for this script:
  - Plaintext Message File: (.txt)
  - Customer Public Key File
  - Bank Private Key File
- The output files of this script:
  - Ciphertext Message File: (.pem)
  - **Signed Ciphertext Message File: (.txt)**

Generate and Interpret QR Code (bash script3)

Now we take the encrypted and signed message and generate a QR code. We will be able to see both the QR code and read from it the same cipher.txt data from the decryption process in script2.

- **Input files of this script:**
  - cipher.txt

- **Output files of this script:**
  - qrcode.png
  - qrcipher.txt

Verify and Decrypt (bash script4)

```
gpg --output qrcipherverified.pem --decrypt qrcipher.txt
gpg -u 3896EED0 --passphrase password --output qrcipherdec.txt
--decrypt qrcipherverified.pem
echo Message Decrypted: Printing Message
more qrcipherdec.txt
```

- The input files for this script:
  - Signed Ciphertext Message File: (.txt)
  - Customer Private Key File
  - Bank Public Key File
- The output files for this script:
  - Ciphertext Message File: (.pem)
  - **Plaintext Message File: (.txt)**

Demonstration

Questions?