Secure and Private Service Discovery over Low Energy Bluetooth
Square Register
- Hardware credit card reader
- Developed for businesses
- Android and iOS app
Square Wallet

• Android and iOS app
• Works with Square Register
• Discovery, payments and loyalty
Register and Wallet - Discovery

- At startup the app gets its GPS location
- Contacts server for nearby merchants
- Displays merchants to the user
Register and Wallet - Payment

• User opens a tab
• Wallet app tells the server
• Server checks Wallet and Register locations
• If close, Server tells Register about Wallet
• Register displays the user’s name and photo
• Customer pays by saying their name
Register and Wallet - Payment

- User opens a tab
- Wallet app tells the server
- Server checks Wallet and Register locations
- If close, Server tells Register about Wallet
- Register displays the user’s name and photo
- Customer pays by saying their name
Register and Wallet - Payment

- User opens a tab
- Wallet app tells the server
- Server checks Wallet and Register locations
- If close, Server tells Register about Wallet
- Register displays the user's name and photo
- Customer pays by saying their name
Challenges

- Getting geo-location at start up takes time
- Keeping app running in background drains battery
- Discovery requires round-trips to the server
- Overall time adds friction
Discovery and payment with BLE
Bluetooth low energy

- Sometimes called Bluetooth Smart
- Not backward-compatible with the previous Bluetooth protocol
- Proximity sensing through signal strength
- Support on Apple products, gaining support on Android
## BLE specs

<table>
<thead>
<tr>
<th>Spec</th>
<th>Bluetooth Classic</th>
<th>BLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power consumption</td>
<td>1 (reference)</td>
<td>0.01 to 0.05</td>
</tr>
<tr>
<td>Wakeup time</td>
<td>100+ ms</td>
<td>&lt; 6ms</td>
</tr>
<tr>
<td>Throughput</td>
<td>0.7-2.1 Mbps</td>
<td>0.27 Mbps</td>
</tr>
<tr>
<td></td>
<td>(meas. ~10-20kbps)</td>
<td></td>
</tr>
</tbody>
</table>
Three key exchange modes, two are completely insecure, the third is not particularly practical

- No public key crypto
- Link layer must be considered in-clear
- Security left to the application layer
Our protocol goals

- Untraceability, Wallets cannot be traced by the content of their transmissions over BLE
- Provide off-line Wallets with a secure channel to our servers over BLE via Register
- Very short messages over BLE
Discovery

- Wallet wants to discover nearby Registers
- Wallet periodically broadcasts anonymous “Square Hello” message (only protocol version number)
- Register replies with BLE certificate
- Certificates are periodically issued to authenticated Registers via HTTPS
Properties

- Register does not learn anything about a Wallet (apart from the rotating BLE layer information)
- Wallets learns about the signed identity of nearby Registers
- ECDSA chosen as signature algorithm (good implementation support on multiple platform)
- ECDSA also allows for compact signatures
Wallet has no Internet connection

But wants to send data to server through a Register (e.g., make a payment)

Protocol to allow Wallet to send confidential, anonymous and authenticated messages to a Square server through an online Register
Setup

- The Wallet has a valid and current session token
- The Wallet has information about nearby Registers (gathered through the discovery protocol above)
- The Server has an encryption key (pk_server, sk_server) and the Wallet has pk_server
Wallet constructs the following message to send over BLE

- \( m = (\text{wallet\_session\_token}, \text{timestamp}, \text{message\_to\_server}) \)
- \( c = \text{ECIES encryption of } m \) using \( \text{pk\_server} \)

Wallet sends \( c \) to nearby Register over BLE

Register forwards \( c \) to Server over HTTPS

Server checks the validity of the session token and the timestamp
Server and Wallet can derive a shared key $k_e$ using the shared wallet_session_token

- $c = \text{ECIES}(\text{wallet_session_token}, \text{timestamp}, \text{message_to_server})$
- $c' = \text{AES-GCM}(k_e, (\text{timestamp_from_wallet}, \text{message_from_server}))$

Register forwards $c'$ to Wallet

Wallet receives $c'$ and checks the message
Payment over BLE

- Wallet message_to_server contains the following information:
  - register_id
  - ble_certificate_hash
  - message_type (e.g., ‘open_tab’)

- Server checks whether the information for register_id matches the ble_certificate_hash

- It opens a tab for the Wallet and reveals the Wallet identity to the Register