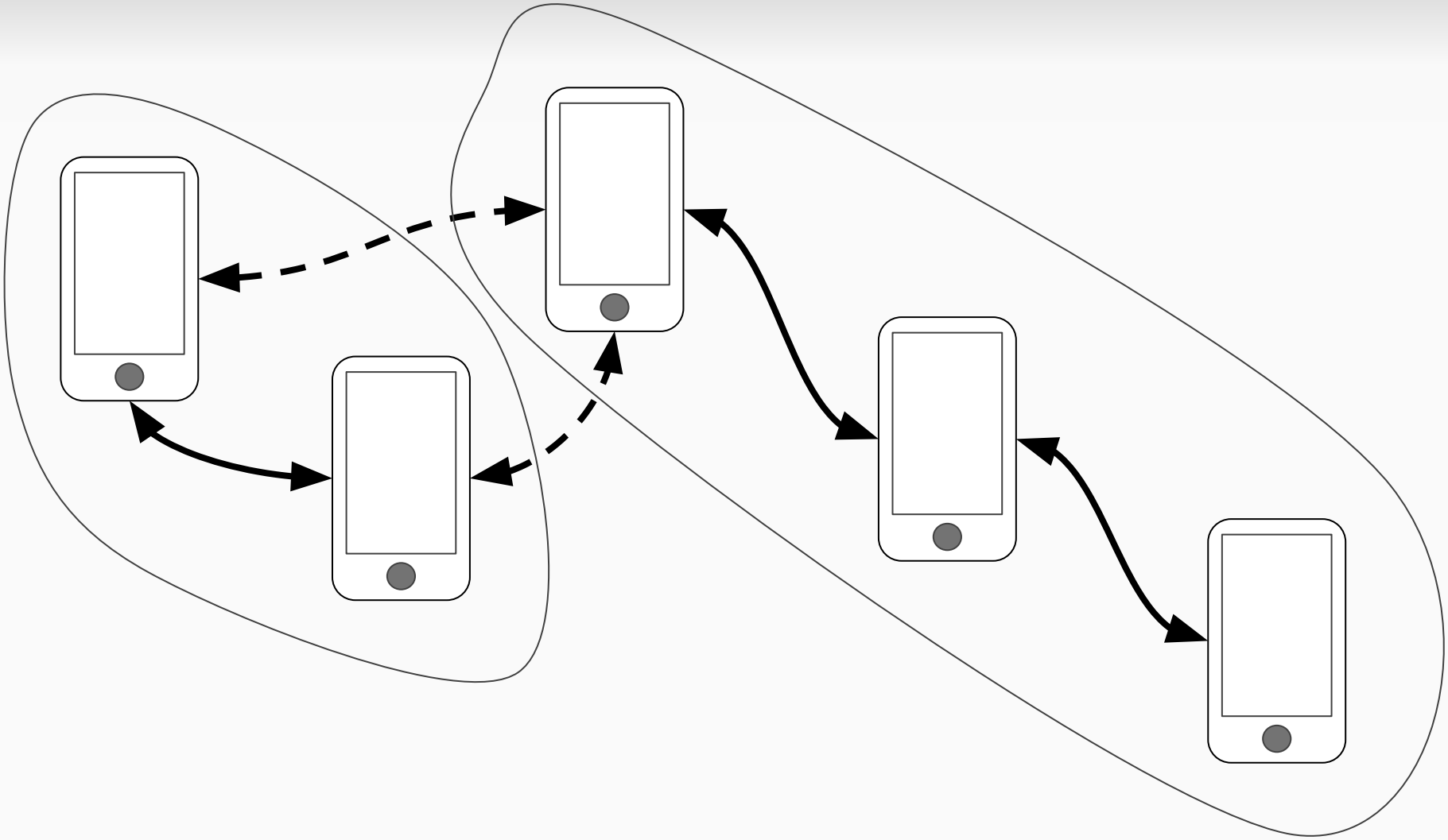
AirMuler

An Anonymous Data Muling Framework

Ankush Gupta, Justin Martinez
github.com/ankushg/AirMuler

Dynamic Multihop Diagram



Multihop Setup

- Message and ACK Buffers
  - Broadcast to new peers as they connect
  - Fixed size so new messages replace old ones
- Sent Message buffer maps Messages sent from this peer with their plaintext
- Messages have a TTL measured in number of hops before expiring

Transmission Handling

- Upon Message receipt:
  - If decryptable:
    - ACK is created and buffered for broadcast
    - Client is notified of data receipt
  - If not decryptable, Message is buffered for rebroadcast
- Upon ACK receipt:
  - Corresponding Message is debuffered if present in Message buffer
  - ACK is buffered for broadcast
  - Client is notified of ACK receipt if Sent Message buffer contains a match

Encryption Scheme

Eph_{PK}, Eph_{SK} = Ephemeral Asymmetric Keys
AckKey, BlobKey = Ephemeral Symmetric Keys
S_{PK}, S_{SK}, R_{PK} = Sender & Receiver Asymmetric Keys
M = Client Payload

$\mathrm{UUID} = \mathrm{Eph}_{PK}$
$\mathrm{UUID}_E = \mathrm{Enc}_{\mathrm{AckKey}}(\mathrm{UUID})$
$\mathrm{M}_E = \mathrm{Seal}_{\mathrm{R}_{PK},\, \mathrm{S}_{SK}}(\mathrm{M})$
$\mathrm{Blob}_E = \mathrm{Enc}_{\mathrm{BlobKey}}(\mathrm{M}_E, \mathrm{S}_{PK}, \mathrm{AckKey})$
$\mathrm{BlobKey}_E = \mathrm{Seal}_{\mathrm{R}_{PK},\, \mathrm{Eph}_{SK}}(\mathrm{BlobKey})$

Transmit Message $(\mathrm{UUID}, \mathrm{UUID}_E, \mathrm{Blob}_E, \mathrm{BlobKey}_E)$

Encryption Scheme

- All transmitted data is encrypted such that neither the sender nor the receiver's identity is known
  - $UUID_E$ is encrypted using the AckKey
  - $BlobKey_E$ is sealed using the Ephemeral secret key and the Receiver's public key
  - $Blob_E$ is encrypted using the BlobKey

Decryption Scheme

R_{PK}, R_{SK} = Receiver Asymmetric Keys

Receive Message (UUID, UUID_E, Blob_E, $\mathrm{BlobKey}_E$)

Eph_{PK} = UUID
$\mathrm{BlobKey} = \mathrm{Open}_{\mathrm{Eph}_{PK},\, \mathrm{R}_{SK}}(\mathrm{BlobKey}_E)$
$(\mathrm{M}_E, \mathrm{S}_{PK}, \mathrm{AckKey}) = \mathrm{Dec}_{\mathrm{BlobKey}}(\mathrm{Blob}_E)$
$\mathrm{M} = \mathrm{Opens}_{\mathrm{S}_{PK},\, \mathrm{R}_{SK}}(\mathrm{M}_E)$

Transmit Ack (UUID, AckKey)

Decryption Scheme

- The receiver cannot determine the sender's identity unless the message is intended for him
  - BlobKey_E can only be opened using the intended Receiver's secret key and the UUID (Ephemeral public key)
  - Blob contains the Sender's public key, allowing the Receiver to open M_E

ACK Scheme (Zero Knowledge Proof)

- After decryption, the receiver transmits the ACK (UUID, AckKey)
- Any node who receives this ACK can check if their Message buffer contains a message with a matching UUID
  - Anyone verify that ACK is valid by encrypting UUID with the AckKey, and checking against $UUID_E$
  - This Zero Knowledge Proof does not expose the Receiver's identity, as the ACK could have been retransmitted from any node

AirMuler Framework

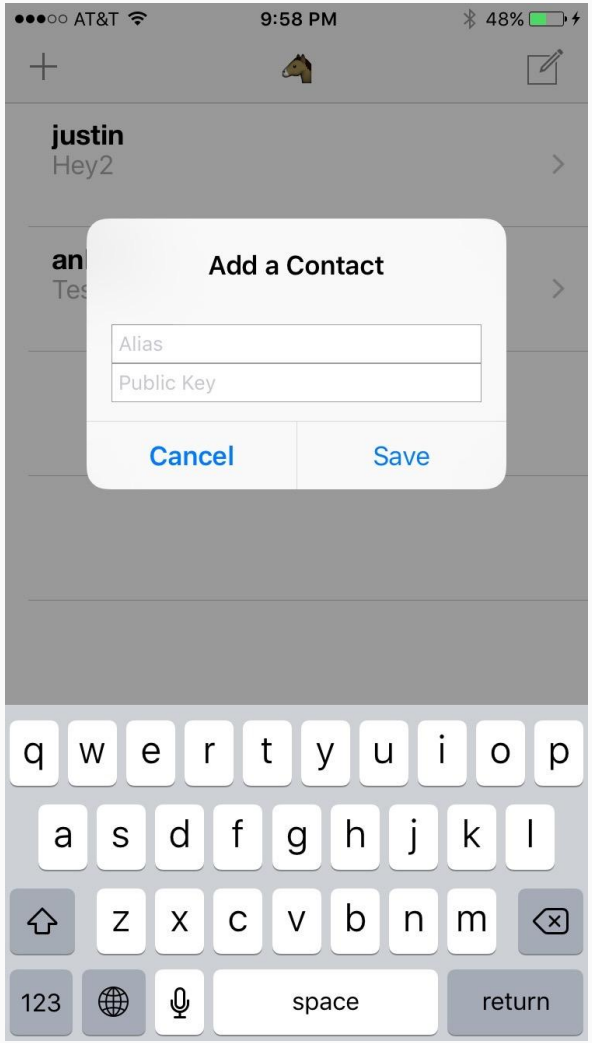
- Custom crypto can be supplied by implementing `CryptoProvider`
  - **`encryptMessage`**`(message: NSData, with keyPair: KeyPair, to recipient: PublicKey) throws -> NSData?`
  - **`decryptMessage`**`(message: NSData, with keyPair: KeyPair) throws -> (payload: NSData?, from: PublicKey, ackMessage: NSData?)`
  - **`checkBuffer`**`(buffer: [NSData], against ackMessage: NSData) throws -> Int?`
- `SodiumCryptoProvider`
  - Uses `libsodium` to compute all cryptographic functions as detailed above

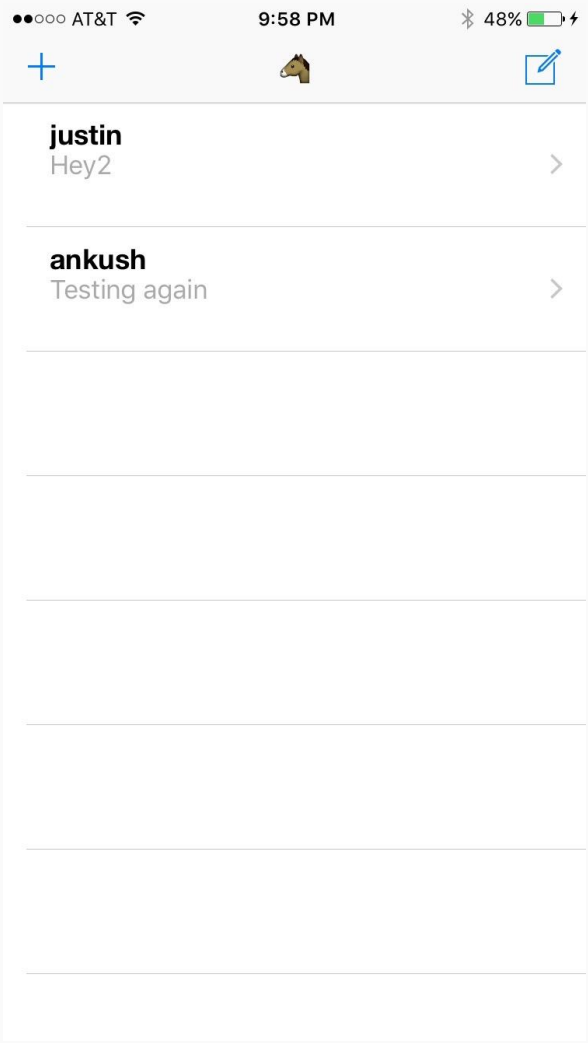
AirMuler Framework

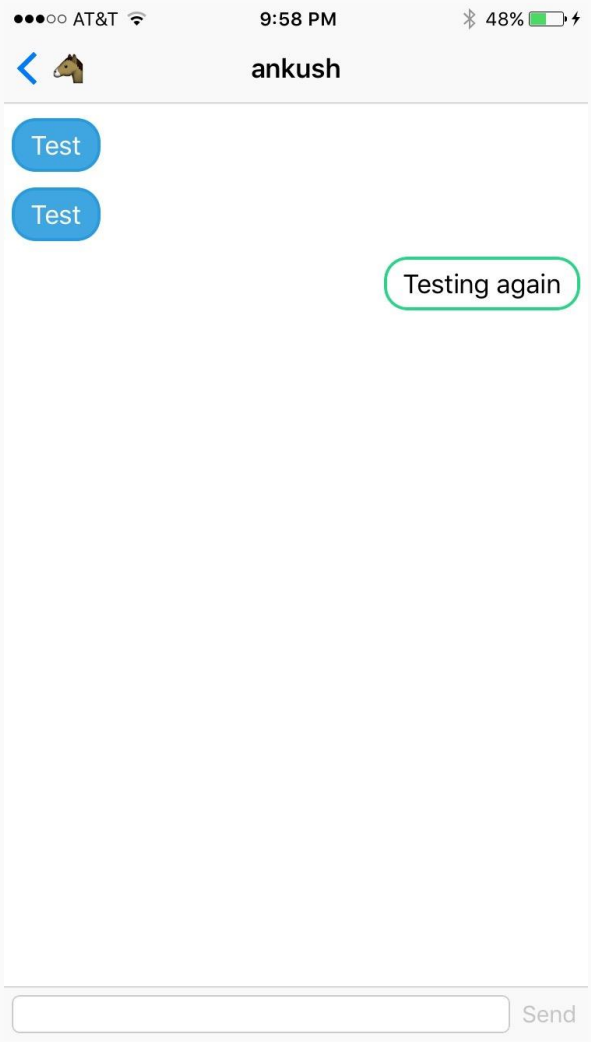
- `NetworkLayer`
  - Initialized with a subclass of `CryptoProvider`
  - Implements Multipeer Connectivity APIs
  - Maintains state for Message, ACK, and Sent Message buffers
  - Handles broadcasting messages to new connections
    - Utilizes a randomized TTL to eliminate exposure based on observing messages
  - Allows client to register as a `NetworkLayerDelegate`
  - Allows client to call **`sendMessage`**`(message: NSData, to recipient: PublicKey)`
- `NetworkLayerDelegate` methods
  - **`connectedWithKey`**`(key: PublicKey)`
  - **`receivedMessage`**`(message: NSData?, from publicKey: PublicKey)`
  - **`acknowledgedDeliveryOfMessage`**`(message: NSData?, to publicKey: PublicKey)`

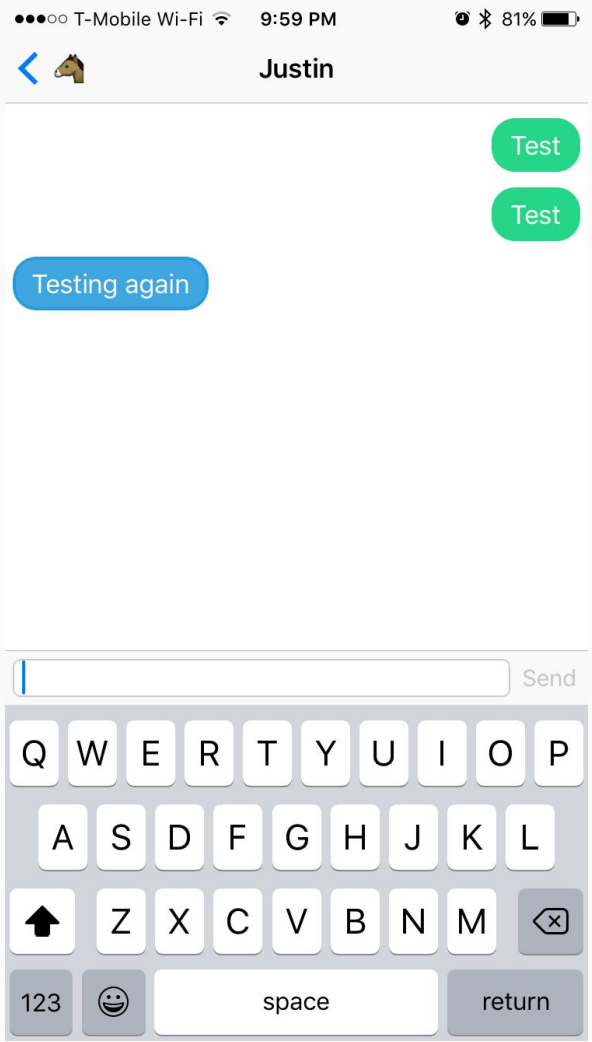
HeeHaw

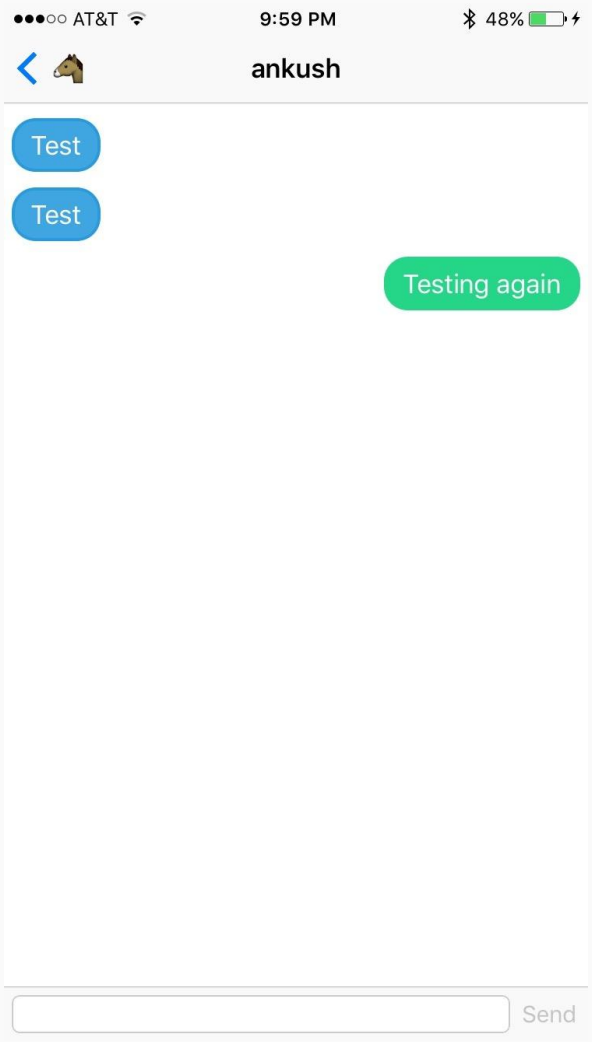
- Example application that utilizes the AirMuler Framework
- Multipeer encrypted chat client
- Allows users to add contacts with an Alias + Public Key combination
  - Creates separate message thread for each alias
- Sends JSON `{ messageText, timestamp }` through `NetworkLayer`
- Saves sent/received messages through CoreData
- Provides visual feedback of message receipt when ACK is received


●●●○○ AT&T
9:58 PM
48%
justin
Hey2
an
Tes
Add a Contact
Alias
Public Key
Cancel
Save
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
return


●●○○○ AT&T
9:58 PM
48%
justin
Hey2
ankush
Testing again


●●●○○ AT&T
9:58 PM
48%
ankush
Test
Test
Testing again
Send


●●●○○ T-Mobile Wi-Fi
9:59 PM
81%
Justin
Test
Test
Testing again
Send
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return


●●●○○ AT&T
9:59 PM
48%
ankush
Test
Test
Testing again
Send