Final Report: Secure Messaging (Signal)

Shashank Singh

Arizona State University
sksing32@asu.edu

Abstract. This project presents the design and implementation of a secure end-to-end messaging system inspired by the Signal protocol. The system enables two clients, Alice and Bob, to communicate securely through an untrusted relay server. The protocol integrates Triple Diffie-Hellman (X3DHE) key exchange, HKDF-based key derivation, AES-GCM authenticated encryption, and a ratcheting mechanism to ensure confidentiality, integrity, authenticity, and forward secrecy. The implementation demonstrates modern secure messaging principles in a modular Python framework.

1 System Architecture

The system consists of three main components, each running as a separate process:

- **Alice (Client 1)**: Initiates communication, performs key exchange, sends and receives encrypted messages.
- **Bob (Client 2)**: Registers keys with the server, participates in key exchange, and sends/receives encrypted messages.
- **Server (Relay)**: Passively relays messages between Alice and Bob without accessing plaintext.

Main cryptographic components:

- **Triple Diffie-Hellman (X3DHE)** for asynchronous key exchange.
- **HKDF-SHA256** for deriving symmetric session keys.
- **AES-GCM** for authenticated encryption.
- **Utility functions (utility.py)** for key generation, encryption/decryption, and KDF operations.

2 Security Intuition

The system achieves the following security goals:

- **Confidentiality**: Only Alice and Bob can decrypt messages; the server cannot read ciphertext.

- **Integrity**: AES-GCM ensures tamper detection.
- **Authenticity**: Long-term keys and X3DHE confirm sender identity.
- **Forward Secrecy**: Ephemeral keys and ratcheting prevent compromise of past communications.

3 Implementation Details

server.py

- Runs a TCP relay server on localhost.
- Stores and distributes Bob's public keys to Alice.
- Relays encrypted messages without inspecting contents.

alice.py

- Generates long-term and ephemeral key pairs.
- Retrieves Bob's public keys from the server.
- Executes X3DHE using static and ephemeral keys.
- Derives shared secrets via HKDF.
- Encrypts outgoing messages with AES-GCM.
- Implements a ratchet mechanism to update keys per message.

bob.py

- Generates long-term and ephemeral keys and registers with the server.
- Awaits Alice's connection and performs X3DHE.
- Derives session keys and decrypts incoming messages.
- Encrypts replies using AES-GCM and advances the ratchet.

utility.py

- Provides core functions:
  - Generate elliptic curve key pairs.
  - Perform Diffie-Hellman key agreement.
  - Derive keys using HKDF with SHA-256.
  - Encrypt/decrypt using AES-GCM.

4 How to Run

```
# Terminal 1
python server.py

# Terminal 2
python bob.py

# Terminal 3
python alice.py
```

Dependencies:

```
pip install cryptography
```

5 Additional Notes

- Runs locally (127.0.0.1).
- Focused on core cryptographic components.
- Advanced Signal features like prekeys and group messaging are not included.
- Code is modular, readable, and well-commented.