[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2023-2024

Scribed by: Raghav Agiwal (202151124) — Lecture (Week 12)

1 Signal Protocol Overview

The Signal Protocol is a comprehensive framework designed to provide end-to-end encryption for secure messaging. It encompasses several cryptographic components and procedures to ensure confidentiality, integrity, and authenticity of communication.

- **Objective:** Ensure secure and private communication between users.
- **End-to-End Encryption:** Messages are encrypted on the sender's device and can only be decrypted by the intended recipient's device.
- **Authentication:** Validates the identity of communicating parties and ensures message integrity.
- **Forward Secrecy:** Ensures that even if long-term keys are compromised, past messages remain secure.

2 Main Cryptographic Modules

2.1 X3DH (Extended Triple Diffie-Hellman)

Purpose: Key agreement protocol for secure communication initiation.

- **Key Generation:**

Alice:	Bob:
IKA, nIA	IKB, nIB
SPKA, nSP A	SPKB, nSP B
	(signed prekey pairs)
Shared Secret:	Shared Secret:
SK_AB = DH(IKA, nIB) —— DH(IKA, nIBP) —— DH(IKA, nSP B) —— DH(IKA, nSP BP)	SK_BA = DH(IKB, nIA) —— DH(IKB, nIAP) —— DH(IKB, nSP A) —— DH(IKB, nSP AP)
Master Key Derivation	Master Key Derivation

2.2 ECDSA (Elliptic Curve Digital Signature Algorithm)

Purpose: Digital signature algorithm for signing public keys.

- **Key Generation:**

Alice:	Bob:
Sign(IKA, nIA)	Sign(IKB, nIB)

2.3 Double Ratchet

Purpose: Key management algorithm for forward secrecy and message encryption.

- **Session Key Derivation:**
  - A new key is derived for each message exchange using a key schedule algorithm.
  - Previous session keys are combined with new key material to derive subsequent session keys.

3 Registration Process

3.1 Alice Registration

Key Generation:

- Alice generates key material including:
  - Identity key pair (IKA, nIA)
  - Signed prekey pair (SPKA, nSP A)
  - Prekey signature: Sig(IKA, Encode(SPKA))
  - One-time prekeys (OPK1A, OPK2A, OPK3A, ...)

Packet Transmission: Alice uploads the generated key material (PacketA) to the server.

3.2 Bob Registration

Key Generation:

- Bob follows a similar process as Alice, generating his own key material (PacketB).
- Key material includes Bob's identity key pair, signed prekey pair, prekey signature, and one-time prekeys.

Packet Transmission: Bob uploads his key material (PacketB) to the server.

Conclusion

The Signal Protocol employs a combination of cryptographic techniques such as key agreement, digital signatures, and key management to ensure secure and private messaging. Through rigorous key generation, signing, and encryption procedures, it establishes and maintains secure communication channels between users, offering a high level of privacy and security.

4 Diffie-Hellman Computation of Alice

4.1 1. Ephemeral Key Generation

Alice generates an ephemeral key EKA, which is a temporary key used for the current session.

4.2 2. DH1 Calculation

Alice computes $DH1 = DH(IKA, SPKB)$, where IKA is Alice's identity key and $SPKB$ is Bob's signed prekey. $DH1$ represents the shared secret between Alice's identity key and Bob's signed prekey.

4.3 3. DH2 Calculation

Alice computes $DH2 = DH(EKA, IKB)$, where IKB is Bob's identity key. $DH2$ represents the shared secret between Alice's ephemeral key and Bob's identity key.

4.4 4. DH3 Calculation

Alice computes $DH3 = DH(EKA, SPKB)$. $DH3$ represents the shared secret between Alice's ephemeral key and Bob's signed prekey.

4.5 5. One-time Prekey Check (Optional)

- If Bob's key bundle contains a one-time prekey:
  - $DH4 = DH(EKA, OPKB)$ is computed.
  - SKA (Alice's shared secret key) is derived using a Key Derivation Function (KDF) with inputs $DH1||DH2||DH3||DH4$.
- Otherwise, SKA is derived using $DH1||DH2||DH3$.

5 Initial Encryption in Alice's Side

5.1 1. Associated Data (AD) Computation

Alice constructs associated data $AD = \text{Encode}(IKA)||\text{Encode}(IKB)$, representing the identities of both Alice and Bob.

5.2 2. Encryption

Alice encrypts an initial message using AD, SKA, and generates the corresponding initial ciphertext. The ciphertext along with IKA, EKA, and Bob's prekeys are sent to the server.

5.3 Initial Ciphertext Decryption in Bob's Side

5.3.1 1. Key Recovery

Bob retrieves IKA and EKA from the received data.

5.3.2 s DH and KDF

Bob performs Diffie-Hellman and Key Derivation Function (KDF) using the shared secret derived from his own keys.

5.3.3 Shared Key Construction

Due to the properties of DH and KDF, Bob's derived shared key SKB matches Alice's derived shared key SKA.

5.3.4 Associated Data Reconstruction

Bob constructs associated data $AD = \text{Encode}(IKA)||\text{Encode}(IKB)$ using the recovered identity keys.

5.3.5 Decryption

Bob decrypts the initial ciphertext using SKB and AD to recover the original message.

6 Zero Knowledge Proof using DLP

Both prover and verifier agreed on Z_p^* where p is prime. Prover wants to prove the knowledge of x satisfying $y = g^x$ without revealing x.

6.1 Prover

1. **Agreement on Parameters:** Both parties agree on a prime number p and a generator g in the group Z_p^*, where p is prime.
2. **Knowledge Proof:** Prover wants to prove knowledge of an exponent x satisfying $y = g^x \mod p$, where y is known to both parties.
3. **Computations:**
   (a) Prover computes $y = g^x \mod p$ and sends y to the verifier.
   (b) Prover selects a random value r from the interval $[0, p-1]$.
   (c) Prover computes $m = g^r \mod p$ and sends m to the verifier.
4. **Challenge:** Verifier chooses a random challenge e from the interval $[0, p-1]$.
5. **Response:** Prover computes $s = ex + r \mod p$ and sends s to the verifier.

Verifier

1. **Verification:** Verifier computes $g^s \mod p$ and $(y \cdot m^e) \mod p$. If $g^s \mod p = (y \cdot m^e) \mod p$, the proof is considered valid.

6.2 Explanation

- The prover demonstrates knowledge of x by providing evidence that y is the result of raising g to the power of x, without revealing x itself.
- The challenge e ensures that the prover cannot precompute s, as e is chosen by the verifier after seeing the prover's initial message.
- The verification step ensures that s is calculated correctly based on the randomness introduced by both r and e, and that it corresponds to the claimed exponent x.

This protocol demonstrates zero-knowledge because the verifier is convinced of the prover's knowledge of x without learning any information about x itself.

7 RC4 Stream Cipher

RC4 (Rivest Cipher 4) is a stream cipher algorithm used for encryption and decryption of data. It was developed by Ronald Rivest in 1987 and gained popularity due to its simplicity and efficiency. The RC4 algorithm is widely used in various protocols and applications, including SSL/TLS, WEP, and Bluetooth.

7.1 RC4 Stream Cipher Overview

7.1.1 Key Scheduling Algorithm (KSA)

- **Data Structure:** RC4 uses an S-Box, denoted as S, consisting of N elements, where $N = 2^8$.
- **Initialization:** Initially, the S-Box is initialized with values from 0 to $N - 1$ (i.e., $S[i] = i$).
- **Scrambling:** During key scheduling, the S-Box is scrambled based on the secret key provided.
  - For each iteration from 0 to $N - 1$:
    - Update the variable j using the formula: $j = (j + S[i] + K[i]) \mod N$, where K is the key array.
    - Swap the values of $S[i]$ and $S[j]$.

7.1.2 Pseudo-Random Generation Algorithm (PRGA)

- **Initialization:** Initialize two pointers i and j to 0.
- **Keystream Generation Loop:** Generate a pseudo-random keystream indefinitely.
  - Increment i by 1.
  - Update j by adding $S[i]$ to it.
  - Swap the values of $S[i]$ and $S[j]$.
  - Calculate $t = S[i] + S[j]$.
  - Output the keystream byte $z = S[t]$.

7.2 Explanation

- **Key Scheduling:** The KSA initializes the S-Box based on the secret key provided, ensuring that each element of the S-Box is influenced by the key.
- **Pseudo-Random Generation:** The PRGA generates a pseudo-random keystream by iteratively shuffling the S-Box and generating output bytes. The output byte z is produced by selecting an element from the S-Box based on the current state of S.
- **Security:** The security of RC4 relies on the unpredictability of the keystream generated. However, RC4 has been subject to vulnerabilities, such as biases in the initial keystream bytes, which can weaken its security.

RC4 is a fast and simple stream cipher, but its use has been deprecated in many applications due to vulnerabilities discovered over time. Despite this, it remains a crucial historical cipher and is still used in some legacy systems.