CSE 410 Fall 2025
Privacy-Enhancing Technologies

Marina Blanton

Department of Computer Science and Engineering
University at Buffalo

Lecture 17: End-to-End Encryption in Messaging

Secure Communication

We primarily considered building secure communication channels between a client and a server

The situation is different with messaging apps

- securing client-server communication is insufficient for conversation confidentiality
- the server is also in an ideal position to listen to (and tamper with) conversations
  - it is located between two clients and can serve the role of man in the middle

End-to-End Encryption

We start with Diffie-Hellman between two users

Recall that we want an authenticated version resilient to man-in-the-middle attacks

- Alice chooses random a and sends $x_A = g^a$ to Bob
- Bob chooses random b and sends to Alice

$$x_B = g^b,\ y_B = \sigma_B(A||x_B||x_A)$$

- Alice verifies y_B and sends to Bob

$$y_A = \sigma_A(B||x_A||x_B)$$

- Alice and Bob start using $k = g^{ab}$

End-to-End Encryption

This requires each user to have a public-private key pair

This requires each user to reliably obtain keys of other users

Let's look at Signal's design

- it is open-source software with design documents
- other proprietary apps are difficult to study and analyze
- Signal leads in terms of security properties and other apps later adopt similar solutions

Signal's Key Establishment

Signal uses what's called X3DH key agreement

Each user has a long-term identity key IK

Each user also has

- signed prekey SPK, which is changed periodically
- one-time use prekey OPK (multiple are generated)
- also, a new ephemeral key EK is generated during key agreement

Signal's Key Establishment

X3DH between Alice and Bob has three phases:

- Bob publishes key identity key and prekeys to a server
- Alice fetches a "prekey bundle" for Bob from the server and uses it to send an initial message to Bob
- Bob receives and processes Alice's initial message

A central design consideration is to make key exchange non-interactive

- the goal is to compute the key and deliver Alice's message to Bob prior to Bob's response

Signal's Key Establishment

Bob publishes the following key material to the server

- identity key IK_B
- prekey SPK_B and a signature on it $\sigma_{\mathsf{IK}_B}(\mathsf{SPK}_B)$
- a set of one-time prekeys $\mathsf{OPK}_B^1, \mathsf{OPK}_B^2, \mathsf{OPK}_B^3, \ldots$

Alice consequently fetches IK_B, PBK_B, $\sigma_{\mathsf{IK}_B}(\mathsf{SPK}_B)$, and (optionally) one OPK_B

Signal's Key Establishment

For Alice to send her initial message:

- Alice verifies the prekey signature
- she generates an ephemeral key EK_A and computes
  - $\mathsf{DH}_1 = \mathsf{DH}(\mathsf{IK}_A, \mathsf{SPK}_B)$
  - $\mathsf{DH}_2 = \mathsf{DH}(\mathsf{EK}_A, \mathsf{IK}_B)$
  - $\mathsf{DH}_3 = \mathsf{DH}(\mathsf{EK}_A, \mathsf{SPK}_B)$
  - $\mathsf{DH}_4 = \mathsf{DH}(\mathsf{EK}_A, \mathsf{OPK}_B)$
  - $\mathsf{SK} = \mathsf{KDF}(\mathsf{DH}_1 || \mathsf{DH}_2 || \mathsf{DH}_3 || \mathsf{DH}_4)$
  - DH_1 and DH_2 provide mutual authentication, while DH_3 and DH_4 provide forward secrecy
  - here $\mathsf{DH}(K_1, K_2) = \mathsf{DH}(g^{k_1}, g^{k_2})$ computes $g^{k_1 k_2}$

Signal's Key Establishment

Alice sends Bob an initial message containing:

- Alice's identity key IK_A
- Alice's ephemeral key EK_A
- identifiers stating which of Bob's prekeys were used
- a ciphertext using SK or a function of SK

Signal's Communication

Upon receiving the initial message from Alice, Bob

- retrieves Alice's IK_A and EK_A from the message
- repeats the DH and KDF calculations to derive SK
- attempts to decrypt Alice's ciphertext

If decryption is successful, the protocol is complete

- Bob and Alice continue using SK

Signal's Key Establishment

Security considerations associated with Signal's key agreement:

- authentication: the parties may compare their identity public keys IK_A and IK_B prior to communication
  - if this is not done, there are no guarantees with respect to with whom they are communicating
- replay: if Alice's initial message doesn't use a one-time prekey, the message can be replayed
  - there are mitigations outside X3DH
- deniability: X3DH doesn't give Alice or Bob publishable cryptographic proof of their communication
  - if one party cooperates with a third party during protocol execution, the third party can obtain a proof
  - this limitation on online deniability is intrinsic to asynchronous protocols

Signal's Key Establishment

Security considerations associated with Signal's key agreement:

- signatures: the present design achieves the desired security properties
  - skipping signature on SPK weakens forward secrecy
  - signing DH values using identity keys reduces deniability
- key compromise: compromise of private keys unavoidably has disastrous effect on security, but some mitigations are in place
  - identity key compromise allows impersonation
  - compromise of a prekey private key may affect security of older or newer computed keys SK

Signal's Key Establishment

Security considerations associated with Signal's key agreement:

- server trust: a malicious server could cause communication between Alice and Bob to fail
  - if Alice and Bob don't verify key, there is no guarantee they are talking to each other
  - otherwise, any tampering by the server is detected
- identity binding: identity misbinding is possible even when checking the key
  - dishonest Charlie can present a false (e.g., Bob's) key to Alice

Signal's Double Ratchet

After key agreement, Signal uses the Double Ratchet algorithm to send and receive encrypted messages

- the parties derive a new key for each Double Ratchet message
- earlier keys cannot be calculated from later keys
- the computation also mixed DH values into key derivation
- this gives protection in case of key compromise

Signal's Double Ratchet

Double Ratchet relies on a key derivation function (KDF)

A KDF is a cryptographic function that takes a random secret KDF key and input data and produces output

- the output is indistinguishable from random if the key is not known
- if the key is not secret and random, the KDF should still produce a secure cryptographic hash
- HMAC and HKDF construction fulfill this goal

Signal's Double Ratchet

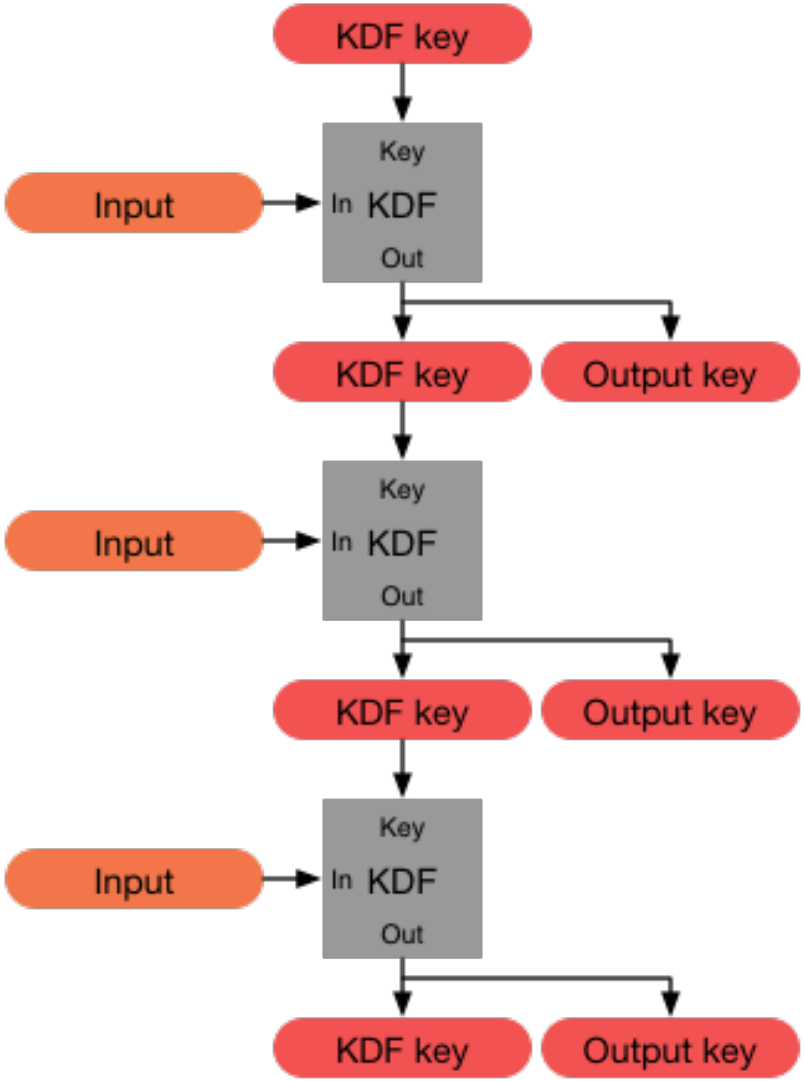
The concept of KDF chain is central to Double Ratchet

The idea is that the output from a KDF is used

- as the output key and
- to replace the KDF key

The updated KDF key is used again with another input

KDF Chain


KDF key
Key
Input
In KDF
Out
KDF key
Output key
Key
Input
In KDF
Out
KDF key
Output key
Key
Input
In KDF
Out
KDF key
Output key

Signal's Double Ratchet

A KDF chain achieves the following properties:

- resilience: output keys appear random to an adversary even if the adversary controls input data
- forward secrecy: past output keys appear random to an adversary
- break-in recovery future output keys appear random to an adversary (if sufficiently random inputs were added)

Signal's Double Ratchet

In a Double Ratchet session between Alice and Bob, there is a KDF key for each of the following chains:

- root chain
- sending chain
- receiving chain

Alice's sending chain is Bob's receiving chain and vice versa

Signal's Double Ratchet

Diffie-Hellman ratchet:

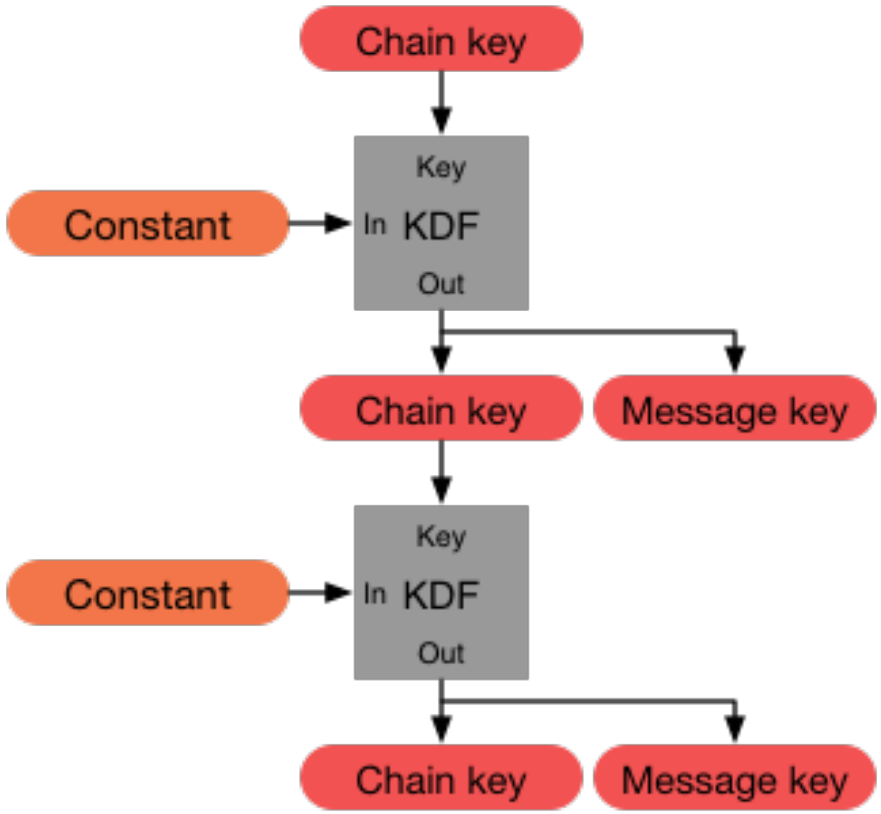
- Diffie-Hellman output secrets become inputs to the root chain
- the output keys from the room chain become new KDF keys for the sending and receiving chains

Symmetric-key ratchet:

- the sending and receiving chains advance with each sent/received message
- the output keys are used to encrypt and decrypt messages

Signal's Double Ratchet

The symmetric-key ratchet is simpler



Signal's Double Ratchet

If an attacker steals a party's sending and receiving chain keys, it can compromise all future message keys

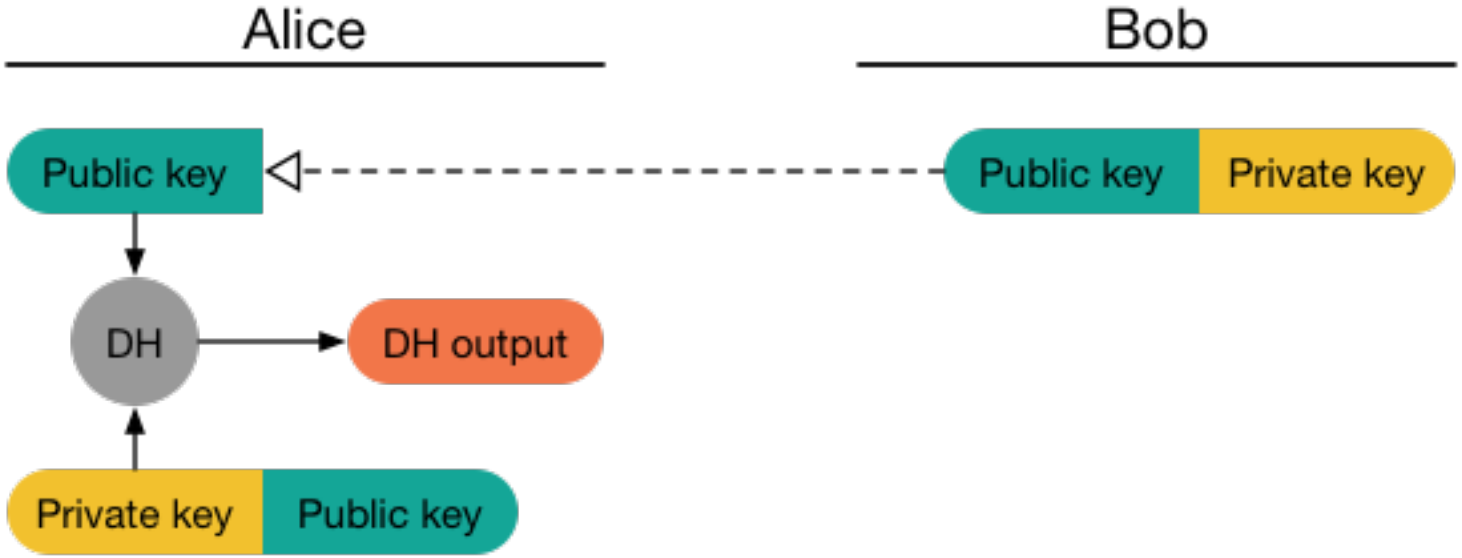
To prevent this, the Double Ratchet includes:

- DH key exchange is executed periodically
- DH ratchet updates chain keys based on DH outputs

The parties alternate in generating ratchet keys

Signal's Double Ratchet

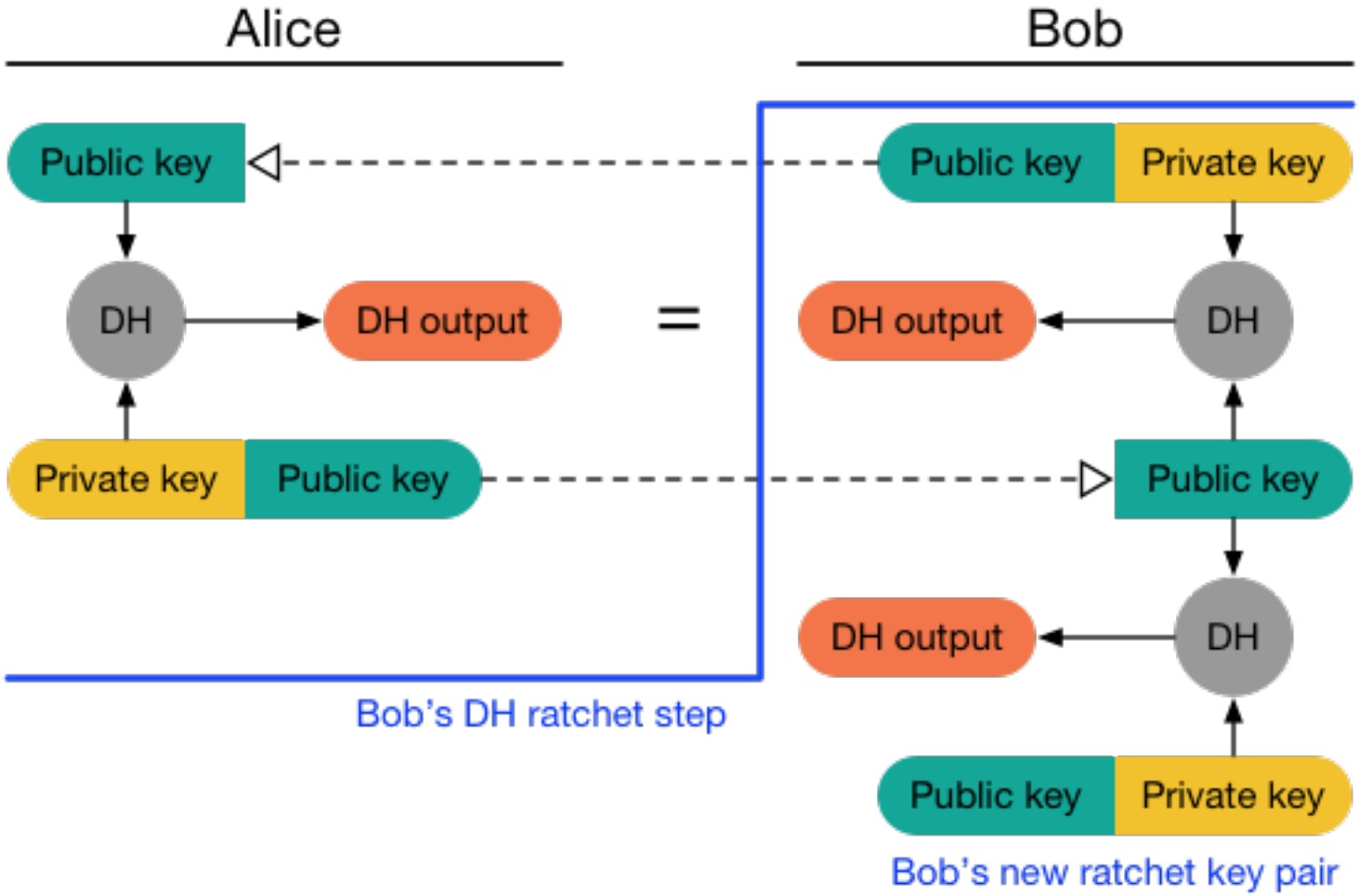
Alice's initial Diffie-Hellman ratchet computation



Alice sends her ratchet public key with the initial message

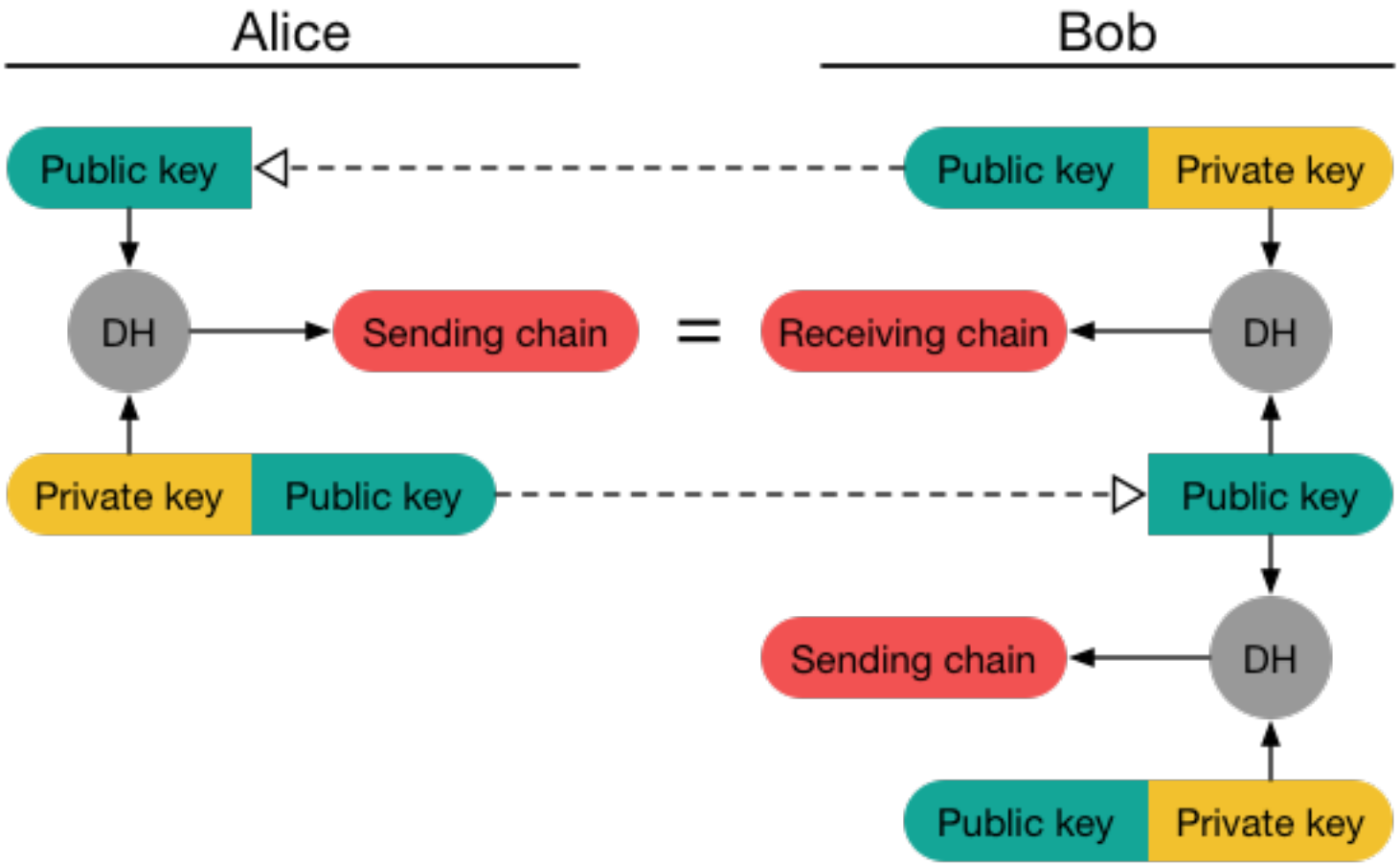
Signal's Double Ratchet

Bob replaces his ratchet key pair and calculates new output



Signal's Double Ratchet

The DH ratchet outputs are used to derive sending and receiving chain keys



Signal's Sealed Sender

Signal provides meta-data protection in the form of a sealed sender

- this goes in line with other efforts to minimize data retained about Signal users

The idea is that when sending a message, the identity of the sender is protected

- when you send physical mail, the envelope contains both from and to names and addresses
- we remove the from address and place it inside the envelope
- when the recipient opens the mail, he/she learns the sender's identity

Signal's Sealed Sender

Previously, Signal required each sender to authenticate prior to delivering an encrypted message to the supplied destination

The authentication served two functions:

- validating the sender's identity to prevent spoofing and provide some assurances to the recipient
- apply rate limiting and abuse protection using the sender identity

When the sender's identity is not known, different solutions are needed

Signal's Sealed Sender

One protection mechanism is sender certificates

- to prevent spoofing, clients periodically retrieve a short-lived sender certificate
- a certificate attests to the client's identity
- it contains the client's phone number, public identity key, and the certificate's expiration time

A certificate is included with a message sent

- the receiver can easily check the sender's identity

Signal's Sealed Sender

Another protection mechanism is delivery tokens

- clients derives a delivery token from their profile key and register it with the service
- Signal profiles are transmitted end-to-end encrypted
- they are accessible only to your contacts and conversations
- when sending a sealed sender message, the service requires to prove knowledge of the recipient's delivery token
- this restricts the use of sealed sender to contacts
- contacts are less likely to require rate limiting and abuse protection

Signal's Sealed Sender

Now, we are ready to talk about sealed sender encryption

- a message is end-to-end encrypted as before
- the envelope now contains the sender certificate
- the envelope is also encrypted
- it uses the recipient's identity key together with a new sender's ephemeral key
- it also uses the sender and recipient identity keys

Signal's Sealed Sender

Putting everything together, to send a sealed sender message:

- encrypt the message as usual
- include a sender certificate in the envelope
- encrypt the envelope to the recipient
- without authenticating, give the envelope to the service together with the recipient's delivery token

Signal's Group Messaging

The last piece we'll discuss is Signal's private group messaging

For conventional two-party conversations, we have attractive security and usability properties

- forward secrecy
- deniability
- asynchronous message delivery

We would like to maintain those for group chats

It is also desirable to have transcript consistency: all members see the same messages in the same order

Signal's Group Messaging

A group conversation could be encrypted using a single key

- e.g., the group originator performs a key agreement with each member and forms the key

Prior state of the art had undesirable properties

- no forward secrecy
- rounds of interaction with each group member

Signal's solution is more secure and zero-round-trip for the setup

Signal's Group Messaging

Signal designer instead chose to use pairwise keys for group communication

Instead of encrypting and sending a single message, a client sends a different ciphertext to each member of the group

- for a group with n members, this means sending n messages instead of 1
- the server's load does not change: it is sending n messages regardless
- for long messages, the client's load can be optimized to n short ciphertexts and 1 long ciphertext

Signal's Group Messaging

Today Signal also provides private groups

- Signal has no record of group membership, titles, avatars or other attributes

But how we ensure that messages are delivered if the service does not know where to send a message?

- this is difficult to do if clients use pairwise secure communication channels
- pairwise communication also makes synchronization challenging

Signal's solution is to change the way message delivery takes place

Signal's Group Messaging

The traditional approach to managing group state is to store it in a database on a server

id	group_name	avatar	members	welcome_message
765	"Surprise party!"	"party-hat.jpg"	sofia (admin), wei, hugo	"Don't tell Alice!"
766	"Book Club Ideas"	"library.png"	jakob, saanvi (admin)	"Les Mis again?"
767	"Finals Week"	"crying.gif"	bob, lucia (admin), leo	"Only 4 days left"

- this obviously allows the server (and an attacker who breaks in) to see all information
- this also permits the server to modify group membership and attributes

Signal obviously wants to do better

Signal's Group Messaging

Suppose group members hold a shared (symmetric) group key

Group information is stored on the service in a way that it cannot read it

id	group_name	avatar	members
765	"E2jP6c8LB8ESLdTNy/tingqAZFj2so8s="	encrypted-file	"2wb5fLpn1yun0acrEbVifXAf566
766	"vNea+GD5bETfL4YS2g/Flhz2lrikq6vY="	encrypted-file	"Y+bt/qjxu8pacBVYyBGJAoVPlIH
767	"dil09QiiGNlt5TajrmfqVs/6VZnk6eSE="	encrypted-file	"+v2Hh1Sh0X64kpAyhXvsn7tuhrN

This creates new issues:

- the service cannot authenticate users
- it does not know who is allowed to fetch and modify group entries
- it does not know who is allowed to read content

Signal's Group Messaging

This type of problem is what anonymous credentials were designed to solve

Anonymous credentials is a token issued to a user by an authority

- it grants the user access or other privileges based on their attributes
- users prove their right or privilege without disclosing other information
- the service does not know anything about the user beside the access outcome
- multiple uses of the same credential cannot linked together
- credentials cannot be forged by an attacker

Signal's Group Messaging

Anonymous credentials are typically issued by one authority, but get verified by a different service

This means that they come in a form of (randomizable) signature on user attributes

When using the credentials, the user proves that her signed (private) credentials satisfy some policy

- in the current context, e.g., a user proves that her ID on the list of group members

This involves zero-knowledge proofs of knowledge

Signal's Group Messaging

A zero-knowledge proof of knowledge (ZKPK) provides a proof of a certain statement

It has to satisfy two properties

- proof of knowledge: the verifier is convinced that the statement of the proof holds
- zero knowledge: nothing beyond the validity of the statement is revealed

Examples

- the prover is of age 21 or older
- the prover is a student at UB

Signal's Group Messaging

This time, anonymous credentials are issued and verified by the same entity

This means we can use more efficient (algebraic) MACs instead of signatures

- prior MAC-based construction did not meet Signal's need
- a proof that a credential attribute matches some encrypted plaintext is needed
- a new solution was designed for Signal's private groups

Signal's Group Messaging

Now a user obtains credentials containing the user's ID

The user authenticates by proving that the identity matches one of the encrypted group member IDs

Some issues still remain

- clients have to prove their credential matches some ciphertext before they can download the ciphertext
- a malicious user can add themselves many times using different ciphertexts

Luckily, both issues can be solved with a single solution: making encryption deterministic

Signal's Group Messaging

We obtain the following:

- suppose Alice has group access credential and the group key
- the server stores member IDs encrypted with the group key
- to add Bob to the group, Alice first proves that she has the corresponding credentials
- if this is successful, she send the server a new entry encrypting Bob's ID
- Alice also sends Bob the group key as an encrypted Signal message
- Bob now can prove group membership, download and decrypt group entries

Signal's Private Messaging

Designing a privacy-preserving application is fascinating

It involves a number of different tools and must consider speed and usability

We discussed

- key agreement between two users (X3DH)
- key derivation and update mechanisms (Double Ratchet)
- protecting metadata by means of sealed sender
- group messaging and private groups