Orca: Blocklisting in Sender-Anonymous Messaging

Nirvan Tyagi    Julia Len    Ian Miers    Tom Ristenpart

USENIX Security 2022
Setting: End-to-end encrypted messaging

From: Alice
To: Bob

Hello
Setting: End-to-end encrypted messaging

- Goal: Confidentiality and Integrity
Setting: End-to-end encrypted messaging

- Goal: Confidentiality and Integrity
- Goal: Conversation participant metadata privacy
Metadata privacy is (relatively) expensive!

- 3 classes of approaches for metadata privacy of sender and recipient identity
Metadata privacy is (relatively) expensive!

- 3 classes of approaches for metadata privacy of sender and recipient identity

Alice

Mixnet

[Disent OSDI'12]
[Vuvuzela SOSP’15]
[Stadium SOSP’17]
[Loopix USENIX Sec’17]
[Karaoke OSDI’18]

Bob
Metadata privacy is (relatively) expensive!

- 3 classes of approaches for metadata privacy of sender and recipient identity

**Multiparty Computation (MPC)**
- [MCMix USENIX Sec'17]
- [AsynchroMix CCS'19]
- [Blinder CCS’20]
- [Clarion NDSS’22]

**Mixnet**
- [Dissent OSDI’12]
- [Vuvuzela SOSP’15]
- [Stadium SOSP’17]
- [Loopix USENIX Sec’17]
- [Karaoke OSDI’18]
Metadata privacy is (relatively) expensive!

- 3 classes of approaches for metadata privacy of sender and recipient identity

**Multiparty Computation (MPC)**
- [MCMix USENIX Sec’17]
- [AsynchroMix CCS’19]
- [Blinder CCS’20]
- [Clarion NDSS’22]

**Mixnet**
- [Dissent OSDI’12]
- [Vuvuzela SOSP’15]
- [Stadium SOSP’17]
- [Loopix USENIX Sec’17]
- [Karaoke OSDI’18]

**Private Information Retrieval (PIR)**
- [Riposte S&P’15]
- [Pung OSDI’16]
- [Express USENIX Sec’21]
Signal’s Sealed Sender: Relaxed metadata privacy

- New messaging protocol released by Signal in 2018
- Focuses on metadata privacy of only sender identity
Problem: Platform unable to filter by sender

- Modern messaging platforms are expected to perform various message filtering tasks on behalf of the recipient client (e.g., blocking spam / abuse)
Problem: Platform unable to filter by sender

- Modern messaging platforms are expected to perform various message filtering tasks on behalf of the recipient client (e.g., blocking spam / abuse)

From: Alice
To: Bob

Platform

Online bully
Spammer
Misinformation

$#@%!
Abuse mitigation: Sender blocklists

From: Alice
To: Bob

Platform

$#@%!
Abuse mitigation: Sender blocklists

Alice

From: ?
To: Bob

$#@%! 

Platform

Bob's blocklist

Alice

Bob
Abuse mitigation: Sender blocklists

Sender-anonymity and sender blocklisting are seemingly at odds!
Outline

- Contribution: Blocklisting for sender-anonymous messaging
- Identifying weaknesses in Signal’s sealed sender protocol
  - Requires non-sender-anonymous communication to initialize
  - Admits untraceable battery-draining (griefing) attack
- Orca: a sender-anonymous blocklisting protocol
  - Group signature scheme for sender-anonymous initialization
  - Efficient one-time-use authentication tokens from algebraic MACs
Background: Sealed Sender

Alice

To: Bob

Platform

Bob
Background: Sealed Sender

- Sealed Sender protects recipients by requiring sender to show recipient’s “access key”
Background: Sealed Sender

- Sealed Sender protects recipients by requiring sender to show recipient’s “access key”

1. Bob registers access key ak
2. Bob sends ak to friends (including Alice)
Background: Sealed Sender

- Sealed Sender protects recipients by requiring sender to show recipient’s “access key”

1. Bob registers access key ak

2. Bob sends ak to friends (including Alice)

3. Alice sends message to Bob, authenticating with ak
Background: Sealed Sender

- Sealed Sender protects recipients by requiring sender to show recipient’s “access key”

1. Bob registers access key $ak$

2. Bob sends $ak$ to friends (including Alice)

3. Alice sends message to Bob, authenticating with $ak$

4. Platform forwards message to Bob if $ak$ is correct
Weaknesses in Sealed Sender design

1. Initial distribution of access key done over non-sender-anonymous channel
Weaknesses in Sealed Sender design

1. Initial distribution of access key done over non-sender-anonymous channel
2. Timing of sender-anonymous messages can leak conversation patterns
Weaknesses in Sealed Sender design

1. Initial distribution of access key done over non-sender-anonymous channel
2. Timing of sender-anonymous messages can leak conversation patterns
3. Untraceable griefing attack mountable by compromised (or malicious) friend
Weaknesses in Sealed Sender design

1. Initial distribution of access key done over non-sender-anonymous channel
2. Timing of sender-anonymous messages can leak conversation patterns
3. Untraceable grieving attack mountable by compromised (or malicious) friend

We demonstrate battery-draining attack [TLMR USENIX Sec’22]
Weaknesses in Sealed Sender design

1. Initial distribution of access key done over non-sender-anonymous channel
2. Timing of sender-anonymous messages can leak conversation patterns
3. Untraceable griefing attack mountable by compromised (or malicious) friend
Weaknesses in Sealed Sender design

1. Initial distribution of access key done over non-sender-anonymous channel
2. Timing of sender-anonymous messages can leak conversation patterns
3. Untraceable griefing attack mountable by compromised (or malicious) friend

Orca [TLMR USENIX Sec’22] addresses (1) & (3)
Weaknesses in Sealed Sender design

1. Initial distribution of access key done over non-sender-anonymous channel
2. Timing of sender-anonymous messages can leak conversation patterns
3. Untraceable griefing attack mountable by compromised (or malicious) friend

Orca [TLMR USENIX Sec’22] addresses (1) & (3)

[MKARW NDSS’21] evaluates (2) and proposes some partial countermeasures that are compatible with Orca
Outline

- Contribution: Blocklisting for sender-anonymous messaging
- Identifying weaknesses in Signal’s sealed sender protocol
  - Requires non-sender-anonymous communication to initialize
  - Admits untraceable battery-draining (griefing) attack
- Orca: a sender-anonymous blocklisting protocol
  - Group signature scheme for sender-anonymous initialization
  - Efficient one-time-use authentication tokens from algebraic MACs
Building block: Group signatures

- **Group manager** manages membership of group
- **Group members** can sign messages anonymously on behalf of the group
- **Opening authority** can open group signature to learn identity of signer, and revoke
Building block: Group signatures

- **Group manager** manages membership of group
- **Group members** can sign messages anonymously on behalf of the group
- **Opening authority** can open group signature to learn identity of signer, and revoke
Building block: Group signatures

- **Group manager** manages membership of group
- **Group members** can sign messages anonymously on behalf of the group
- **Opening authority** can open group signature to learn identity of signer, and revoke
Building block: Group signatures

- **Group manager** manages membership of group
- **Group members** can sign messages anonymously on behalf of the group
- **Opening authority** can open group signature to learn identity of signer, and revoke

Users register with platform as both group members and opening authorities

Alice  
Group member

Bob  
Opening authority

Platform  
Group manager

Platform manages users, with keys gsk, gpk
Building block: Group signatures

- **Group manager** manages membership of group
- **Group members** can sign messages anonymously on behalf of the group
- **Opening authority** can open group signature to learn identity of signer, and revoke

Users register with platform as both group members and opening authorities.

Alice
Group member

Bob
Opening authority

Platform
Group manager

$gsk, gpk$

$sk_A, pk_A$

$osk_B, opk_B$
Building block: Group signatures

- **Group manager** manages membership of group
- **Group members** can sign messages anonymously on behalf of the group
- **Opening authority** can open group signature to learn identity of signer, and revoke

Users register with platform as both group members and opening authorities

Alice signs ciphertext with group signature

\[ \sigma = \text{Sign}(sk_A, opk_B, gpk, ct) \]
Building block: Group signatures

- **Group manager** manages membership of group
- **Group members** can sign messages anonymously on behalf of the group
- **Opening authority** can open group signature to learn identity of signer, and revoke

\[\sigma = \text{Sign}(sk_A, opk_B, gpk, ct)\]

1. Users register with platform as both group members and opening authorities
2. Alice signs ciphertext with group signature
3. Platform forwards if signature verifies against Bob’s revocation list
Building block: Group signatures

- **Group manager** manages membership of group
- **Group members** can sign messages anonymously on behalf of the group
- **Opening authority** can open group signature to learn identity of signer, and revoke

Users register with platform as both group members and opening authorities

\[
\sigma = \text{Sign}(sk_A, opk_B, gpk, ct)
\]

Alice signs ciphertext with group signature

Platform forwards if signature verifies against Bob’s revocation list

\[
b = \text{Verify}(gsk, opk_B, RL_B, ct, \sigma)
\]

Bob opens signature and may choose to revoke Alice’s sending ability
Building block: Group signatures

- **Group manager** manages membership of group
- **Group members** can sign messages anonymously on behalf of the group
- **Opening authority** can open group signature to learn identity of signer, and revoke

1. Users register with platform as both group members and opening authorities

2. Alice signs ciphertext with group signature

3. Platform forwards if signature verifies against Bob’s revocation list

4. Bob opens signature and may choose to revoke Alice’s sending ability

\[
\sigma = \text{Sign}(sk_A, opk_B, gpk, ct)
\]

\[
b = \text{Verify}(gsk, opk_B, RL_B, ct, \sigma)
\]

\[
pk_A = \text{Open}(osk_B, gpk, ct, \sigma)
\]

\[
\tau_A = \text{Revoke}(osk_B, pk_A, gpk)
\]
Building block: Group signatures

1. Users register with platform as both group members and opening authorities

2. Alice signs ciphertext with group signature

3. Platform forwards if signature verifies against Bob’s revocation list

4. Bob opens signature and may choose to revoke Alice’s sending ability

\[
\sigma = \text{Sign}(sk_A, opk_B, gpk, ct)
\]

\[
b = \text{Verify}(gsk, opk_B, RL_B, ct, \sigma)
\]

\[
\tau_A = \text{Revoke}(osk_B, pk_A, gpk)
\]

\[
pk_A = \text{Open}(osk_B, gpk, ct, \sigma)
\]
Building block: Group signatures

- Group signatures are **anonymous**

Users register with platform as both group members and opening authorities

Alice signs ciphertext with group signature

Platform forwards if signature verifies against Bob’s revocation list

Bob opens signature and may choose to revoke Alice’s sending ability
Building block: Group signatures

- Group signatures are anonymous, **traceable**

1. Users register with platform as both group members and opening authorities

2. Alice signs ciphertext with group signature

3. Platform forwards if signature verifies against Bob’s revocation list

4. Bob opens signature and may choose to revoke Alice’s sending ability
Building block: Group signatures

- Group signatures are anonymous, traceable, and revocable

1. Users register with platform as both group members and opening authorities

2. Alice signs ciphertext with group signature

3. Platform forwards if signature verifies against Bob’s revocation list

4. Bob opens signature and may choose to revoke Alice’s sending ability
Building block: Group signatures

- Group signatures are anonymous, traceable, and revocable
- Group signatures do not require initialization over non-sender-anonymous channels

1. Users register with platform as both group members and opening authorities

2. Alice signs ciphertext with group signature

3. Platform forwards if signature verifies against Bob’s revocation list

4. Bob opens signature and may choose to revoke Alice’s sending ability

\[
s_k_A, p_k_A
\]

\[
s = \text{Sign}(s_k_A, o_k_B, g_k, c)
\]

\[
b = \text{Verify}(g_k, o_k_B, R_L, c, s)
\]

\[
p_k_A = \text{Open}(o_k_B, g_k, c, s)
\]

\[
\tau_A = \text{Revoke}(o_k_B, p_k_A, g_k)
\]
Building block: Group signatures

- Group signatures are anonymous, traceable, and revocable
- Group signatures do not require initialization over non-sender-anonymous channels
- Contribution: **Multi-opener group signatures**

Users register with platform as both group members and opening authorities

\[ \sigma = \text{Sign}(sk_A, opk_B, gpk, ct) \]

Alice signs ciphertext with group signature

Platform forwards if signature verifies against Bob’s revocation list

\[ b = \text{Verify}(gsk, opk_B, RL_B, ct, \sigma) \]

Bob opens signature and may choose to revoke Alice’s sending ability

\[ \tau_A = \text{Revoke}(osk_B, pk_A, gpk) \]
Building block: Group signatures

- Group signatures are anonymous, traceable, and revocable
- Group signatures do not require initialization over non-sender-anonymous channels
- Contribution: Multi-opener group signatures, Keyed-verification group signatures

1. Users register with platform as both group members and opening authorities
2. Alice signs ciphertext with group signature
3. Platform forwards if signature verifies against Bob’s revocation list
4. Bob opens signature and may choose to revoke Alice’s sending ability

\[ \sigma = \text{Sign}(sk_A, opk_B, gpk, ct) \]

\[ b = \text{Verify}(gsk, opk_B, RL_B, ct, \sigma) \]
Outline

- **Contribution**: Blocklisting for sender-anonymous messaging
- **Identifying weaknesses in Signal’s sealed sender protocol**
  - Requires non-sender-anonymous communication to initialize
  - Admits untraceable battery-draining (griefing) attack
- **Orca: a sender-anonymous blocklisting protocol**
  - Group signature scheme for sender-anonymous initialization
  - Efficient one-time-use authentication tokens from algebraic MACs
Building block: One-time use access tokens
Building block: One-time use access tokens

Alice

Platform

Bob

1. Bob sends token key to platform $tk_B$
Building block: One-time use access tokens

1. Bob sends token key to platform

2. Bob distributes initial tokens to Alice by some process

We will revisit this shortly... Hint: Group signatures!

Unspent tokens

Alice - $t_1 t_2 t_3$

Bob
Building block: One-time use access tokens

Tokens are generated by sampling a random input $x$ and computing the MAC

$$y = \text{MAC}_{tk}(x)$$

$$t = (x, y)$$

1. Bob sends token key to platform
2. Bob distributes initial tokens to Alice by some process

$tk_B$

Alice - $t_1, t_2, t_3$

Unspent tokens
Building block: One-time use access tokens

Tokens are generated by sampling a random input $x$ and computing the MAC:

$$y = \text{MAC}_{tk}(x)$$
$$t = (x, y)$$

1. Bob sends token key to platform
2. Bob distributes initial tokens to Alice by some process
3. Alice sends message to Bob authenticating with a token

Alice - $t_1 \ t_2 \ t_3$
Bob
Building block: One-time use access tokens

Tokens are generated by sampling a random input $x$ and computing the MAC

$$y = \text{MAC}_{tk}(x)$$
$$t = (x, y)$$

1. Bob sends token key to platform
2. Bob distributes initial tokens to Alice by some process
3. Alice sends message to Bob authenticating with a token
4. Platform forwards to Bob if token is valid and not present on strikelist, then adds token to strikelist
Building block: One-time use access tokens

Tokens are generated by sampling a random input $x$ and computing the MAC

$$y = \text{MAC}_{tk}(x)$$

$$t = (x, y)$$

1. Bob sends token key to platform

2. Bob distributes initial tokens to Alice by some process

3. Alice sends message to Bob authenticating with a token

4. Platform forwards to Bob if token is valid and not present on strikelist, then adds token to strikelist

5. Bob identifies sender based on token and may choose to revoke Alice’s remaining tokens

Unspent tokens

Alice - $t_1$, $t_2$, $t_3$
Building block: One-time use access tokens

Tokens are generated by sampling a random input $x$ and computing the MAC:

$$y = \text{MAC}_{tk}(x)$$

$$t = (x, y)$$

1. Bob sends token key to platform
2. Bob distributes initial tokens to Alice by some process
3. Alice sends message to Bob authenticating with a token
4. Platform forwards to Bob if token is valid and not present on strikelist, then adds token to strikelist
5. Bob identifies sender based on token and may choose to revoke Alice’s remaining tokens

Unspent tokens

Alice - $t_1 t_2 t_3$
Building block: One-time use access tokens

Bob sends token key to platform

Tokens are generated by sampling a random input $x$ and computing the MAC:

$$y = \text{MAC}_{tk}(x)$$

$$t = (x, y)$$

2. Bob distributes initial tokens to Alice by some process.

3. Alice sends message to Bob authenticating with a token.

4. Platform forwards to Bob if token is valid and not present on strikelist, then adds token to strikelist.

5. Bob identifies sender based on token and may choose to revoke Alice’s remaining tokens.

Unspent tokens

Alice - $t_1, t_2, t_3$

“Add to strikelist”
Building block: One-time use access tokens

Do not want to rely on non-sender-anonymous channels!

1. Bob sends token key to platform
2. Bob distributes initial tokens to Alice by some process
3. Alice sends message to Bob authenticating with a token
4. Platform forwards to Bob if token is valid and not present on strikelist, then adds token to strikelist
5. Bob identifies sender based on token and may choose to revoke Alice’s remaining tokens

**Unspent tokens**

\[ y = \text{MAC}_{tk}(x) \]
\[ t = (x, y) \]

Tokens are generated by sampling a random input \( x \) and computing the MAC.
Building block: One-time use access tokens

Do not want to rely on non-sender-anonymous channels!

After initial batch is distributed, future tokens can be replenished in the regular flow of conversation

1. Bob sends token key to platform

2. Bob distributes initial tokens to Alice by some process

3. Alice sends message to Bob authenticating with a token

4. Platform forwards to Bob if token is valid and not present on strikelist, then adds token to strikelist

5. Bob identifies sender based on token and may choose to revoke Alice’s remaining tokens
Building block: One-time use access tokens

Do not want to rely on non-sender-anonymous channels!

Bob distributes initial tokens to Alice by some process

1. Bob sends token key to platform

Alice sends message to Bob authenticating with a token

3. Alice sends message to Bob authenticating with a token

Platform forwards to Bob if token is valid and not present on strikelist, then adds token to strikelist

4. Platform forwards to Bob if token is valid and not present on strikelist, then adds token to strikelist

Bob identifies sender based on token and may choose to revoke Alice’s remaining tokens

5. Bob identifies sender based on token and may choose to revoke Alice’s remaining tokens

How to distribute initial batch?

“Add to strikelist”

Unspent tokens

After initial batch is distributed, future tokens can be replenished in the regular flow of conversation

“Here are some tokens to respond with”
Orca: Hybrid of OTU tokens + Group signatures

- Use group signature to send initial batch of one-time-use (OTU) tokens
Orca: Hybrid of OTU tokens + Group signatures

- Use group signature to send initial batch of one-time-use (OTU) tokens

\[
\begin{align*}
&< m_1, t_1, t_2, t_3 > \\
&\text{Authenticated to platform} \\
&\sigma \text{ to } oapk_A
\end{align*}
\]
Orca: Hybrid of OTU tokens + Group signatures

- Use group signature to send initial batch of one-time-use (OTU) tokens

Alice - \( \frac{t_1}{t_2} t_3 \)

Bob - \( \frac{t_1'}{t_2'} t_3' \)

Platform

Content
\(< m_1, t_1 t_2 t_3 >\)

Authenticated to platform
\(\sigma to oapk_A\)

\(< m_2, t_1' t_2' t_3' >\)

Bob - \( \frac{t_1'}{t_2} t_3 \)

Alice - \( \frac{t_1}{t_2} t_3 \)

\(t_1\) for \(tk_B\)
Orca: Hybrid of OTU tokens + Group signatures

- Use group signature to send initial batch of one-time-use (OTU) tokens

Alice - $t_1 \ t_2 \ t_3$

$< m_1, t_1 \ t_2 \ t_3 >$

Authenticated to platform

$\sigma$ to oapk$_A$

$t_1$ for tk$_B$

Bob - $t'_1 \ t'_2 \ t'_3$

$< m_2, t'_1 \ t'_2 \ t'_3 >$

$< m_3 >$

$t_2$ for tk$_B$
Orca: Hybrid of OTU tokens + Group signatures

- Use group signature to send initial batch of one-time-use (OTU) tokens

Alice - $t'_1 \ t'_2 \ t'_3$

Bob - $t'_1 \ t'_2 \ t'_3$

Platform

< $m_1, t_1 t_2 t_3$ >

Authenticated to platform

$\sigma$ to $oapk_A$

$tk_B$

$t_1$

< $m_2, t'_1 t'_2 t'_3$ >

$t_2$

< $m_3$ >

$t'_1$ for $tk_A$

< $m_4, t_4 t_5 t_6$ >

$t'_2$ for $tk_B$
Orca: Hybrid of OTU tokens + Group signatures

- Use group signature to send initial batch of one-time-use (OTU) tokens

Best of both worlds!
Sender-anonymous initialization via group signatures and efficiency via one-time-use tokens
Orca: Hybrid of OTU tokens + Group signatures

- Use group signature to send initial batch of one-time-use (OTU) tokens

Group signatures until first response? Remedied by oblivious token minting protocol. See paper.

Best of both worlds!
Sender-anonymous initialization via group signatures and efficiency via one-time-use tokens
Impact

- Open source: Implemented to confirm practicality of solution
  - “Steady-state” costs of authenticating via OTU tokens add little overhead
  - Initialization costs of group signature and token minting
    - ~ 200 ms computation for both platform and client
- Disclosed findings to Signal
  - Advising on possible partial mitigations

Open source: https://github.com/nirvanyagi/orca
Archive: https://ia.cr/2021/1380
Summary

- **Contribution**: Blocklisting for sender-anonymous messaging
- Identifying weaknesses in Signal’s sealed sender protocol
  - Requires non-sender-anonymous communication to initialize
  - Admits untraceable battery-draining (griefing) attack
- **Orca**: a sender-anonymous blocklisting protocol
  - Group signature scheme for sender-anonymous initialization
  - Efficient one-time-use authentication tokens from algebraic MACs

Open source: https://github.com/nirvantyagi/orca
Archive: https://ia.cr/2021/1380
Back-up slides
Inference attack: Interleaving messages

<table>
<thead>
<tr>
<th>Time</th>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021-8-18 8:08:59</td>
<td>Bob</td>
</tr>
<tr>
<td>2021-8-18 8:14:02</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:20:19</td>
<td>Alice</td>
</tr>
<tr>
<td>2021-8-18 8:22:25</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 8:29:55</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:31:38</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 8:46:24</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:57:11</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:06:41</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:09:56</td>
<td>Bob</td>
</tr>
<tr>
<td>2021-8-18 9:14:39</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:17:29</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:20:12</td>
<td>Alice</td>
</tr>
<tr>
<td>2021-8-18 9:20:30</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:28:54</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:37:15</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:42:08</td>
<td>Bob</td>
</tr>
</tbody>
</table>
Inference attack: Interleaving messages

<table>
<thead>
<tr>
<th>Time</th>
<th>Recipient</th>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021-8-18 8:08:59</td>
<td>Bob</td>
<td>Bob</td>
</tr>
<tr>
<td>2021-8-18 8:14:02</td>
<td>Claire</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:20:19</td>
<td>Alice</td>
<td>Alice</td>
</tr>
<tr>
<td>2021-8-18 8:22:25</td>
<td>Dave</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 8:29:55</td>
<td>Claire</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:31:38</td>
<td>Dave</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 8:46:24</td>
<td>Claire</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:57:11</td>
<td>Dave</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:06:41</td>
<td>Claire</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:09:56</td>
<td>Bob</td>
<td>Bob</td>
</tr>
<tr>
<td>2021-8-18 9:14:39</td>
<td>Dave</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:17:29</td>
<td>Claire</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:20:12</td>
<td>Alice</td>
<td>Alice</td>
</tr>
<tr>
<td>2021-8-18 9:20:30</td>
<td>Dave</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:28:54</td>
<td>Claire</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:37:15</td>
<td>Dave</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:42:08</td>
<td>Bob</td>
<td>Bob</td>
</tr>
</tbody>
</table>
Inference attack: Interleaving messages

<table>
<thead>
<tr>
<th>Time</th>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021-8-18 8:08:59</td>
<td>Bob</td>
</tr>
<tr>
<td>2021-8-18 8:14:02</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:20:19</td>
<td>Alice</td>
</tr>
<tr>
<td>2021-8-18 8:22:25</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 8:29:55</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:31:38</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 8:46:24</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:57:11</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:06:41</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:09:56</td>
<td>Bob</td>
</tr>
<tr>
<td>2021-8-18 9:14:39</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:17:29</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:20:12</td>
<td>Alice</td>
</tr>
<tr>
<td>2021-8-18 9:20:30</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:28:54</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:37:15</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:42:08</td>
<td>Bob</td>
</tr>
</tbody>
</table>

Bob & Claire?
- Bob
- Claire
- Dave
- Claire
- Claire
- Dave
- Claire
- Bob
- Dave
- Claire
- Alice
- Alice
- Dave
- Claire
- Dave
- Bob

Bob & Alice?
- Bob
- Claire
- Alice
- Dave
- Claire
- Dave
- Claire
- Alice
- Dave
- Claire
- Dave
- Bob
Inference attack: Interleaving messages

Platform

Observations

<table>
<thead>
<tr>
<th>Time</th>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021-8-18 8:08:59</td>
<td>Bob</td>
</tr>
<tr>
<td>2021-8-18 8:14:02</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:20:19</td>
<td>Alice</td>
</tr>
<tr>
<td>2021-8-18 8:22:25</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 8:29:55</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:31:38</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 8:46:24</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:57:11</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:06:41</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:09:56</td>
<td>Bob</td>
</tr>
<tr>
<td>2021-8-18 9:14:39</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:17:29</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:20:12</td>
<td>Alice</td>
</tr>
<tr>
<td>2021-8-18 9:20:30</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:28:54</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:37:15</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:42:08</td>
<td>Bob</td>
</tr>
</tbody>
</table>

Bob & Claire?

<table>
<thead>
<tr>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Alice</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Bob</td>
</tr>
</tbody>
</table>

Bob & Alice?

<table>
<thead>
<tr>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Alice</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Alice</td>
</tr>
</tbody>
</table>

Dave & Claire?

<table>
<thead>
<tr>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Alice</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Claire</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Alice</td>
</tr>
</tbody>
</table>

[MKARW NDSS'21]
Inference attack: Interleaving messages

Attack efficacy varies based on number of conversation participants, frequency of response, participation balance, etc.

Platform

<table>
<thead>
<tr>
<th>Time</th>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021-8-18 8:14:02</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:20:19</td>
<td>Alice</td>
</tr>
<tr>
<td>2021-8-18 8:22:25</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 8:29:55</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:31:38</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 8:46:24</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 8:57:11</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:06:41</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:09:56</td>
<td>Bob</td>
</tr>
<tr>
<td>2021-8-18 9:14:39</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:17:29</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:20:12</td>
<td>Alice</td>
</tr>
<tr>
<td>2021-8-18 9:20:30</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:28:54</td>
<td>Claire</td>
</tr>
<tr>
<td>2021-8-18 9:37:15</td>
<td>Dave</td>
</tr>
<tr>
<td>2021-8-18 9:42:08</td>
<td>Bob</td>
</tr>
</tbody>
</table>

[MKARW NDSS’21]
Orca: Hybrid of OTU tokens + Group signatures

- Oblivious token minting protocol authenticated by group signatures to obtain first batch of tokens without use of non-sender-anonymous channels
Orca: Hybrid of OTU tokens + Group signatures

- Oblivious token minting protocol authenticated by group signatures to obtain first batch of tokens without use of non-sender-anonymous channels
Orca: Hybrid of OTU tokens + Group signatures

- Oblivious token minting protocol authenticated by group signatures to obtain first batch of tokens without use of non-sender-anonymous channels.
Orca: Hybrid of OTU tokens + Group signatures

- Oblivious token minting protocol authenticated by group signatures to obtain first batch of tokens without use of non-sender-anonymous channels

Alice engages in oblivious minting protocol, including group signature over ciphertext of proposed tokens encrypted to Bob

\[ \sigma = \text{Sign}(sk_A, \text{opk}_B, gpk, ct_t) \]

1. Alice engages in oblivious minting protocol, including group signature over ciphertext of proposed tokens encrypted to Bob
Orca: Hybrid of OTU tokens + Group signatures

- Oblivious token minting protocol authenticated by group signatures to obtain first batch of tokens without use of non-sender-anonymous channels

1. Alice engages in oblivious minting protocol, including group signature over ciphertext of proposed tokens encrypted to Bob

\[ \sigma = \text{Sign}(sk_A, opk_B, gpk, ct) \]

2. If group signature verifies, platform engages in oblivious minting protocol, and sends signature and ciphertext to Bob
Orca: Hybrid of OTU tokens + Group signatures

- Oblivious token minting protocol authenticated by group signatures to obtain first batch of tokens without use of non-sender-anonymous channels

1. Alice engages in oblivious minting protocol, including group signature over ciphertext of proposed tokens encrypted to Bob

   \[ \sigma, \mathit{ct}_t \]

   \[ t_1 t_2 t_3 \]

   \[ \mathit{tk}_B \]

   \[ \mathit{gsk}, \mathit{gpk} \]

   \[ \sigma = \mathsf{Sign}(sk_A, \mathit{opk}_B, \mathit{gpk}, \mathit{ct}_t) \]

2. If group signature verifies, platform engages in oblivious minting protocol, and sends signature and ciphertext to Bob

3. Bob decrypts ciphertext to get tokens, opens signature to identify Alice, then stores tokens to identify future messages from Alice

   \[ \mathit{pk}_A = \mathsf{Open}(\mathit{osk}_B, \mathit{gpk}, m, \sigma) \]

   \[ t_1 t_2 t_3 = \mathsf{Decrypt}(\mathit{osk}_B, \mathit{ct}_t) \]

Alice - \( t_1 t_2 t_3 \)

Unspent tokens
Orca: Hybrid of OTU tokens + Group signatures

- Oblivious token minting protocol authenticated by group signatures to obtain first batch of tokens without use of non-sender-anonymous channels

1. Alice engages in oblivious minting protocol, including group signature over ciphertext of proposed tokens encrypted to Bob

   $\sigma = \text{Sign}(sk_A, opk_B, gpk, ct_t)$

2. If group signature verifies, platform engages in oblivious minting protocol, and sends signature and ciphertext to Bob

3. Bob decrypts ciphertext to get tokens, opens signature to identify Alice, then stores tokens to identify future messages from Alice

   $pk_A = \text{Open}(osk_B, gpk, m, \sigma)$

   $t_1 t_2 t_3 = \text{Decrypt}(osk_B, ct_t)$

Unspent tokens

Alice - $t_1 t_2 t_3$
Orca: Hybrid of OTU tokens + Group signatures

- Oblivious token minting protocol authenticated by group signatures to obtain first batch of tokens without use of non-sender-anonymous channels

Alice engages in oblivious minting protocol, including group signature over ciphertext of proposed tokens encrypted to Bob

σ, ct_t

σ = Sign(sk_A, opk_B, gpk, ct_t)

Bob decrypts ciphertext to get tokens, opens signature to identify Alice, then stores tokens to identify future messages from Alice

pk_A = Open(osk_B, gpk, m, σ)

t_1 t_2 t_3 = Decrypt(osk_B, ct_t)

Unspent tokens

Alice - t_1 t_2 t_3

Note: We used algebraic MACs for OTU tokens instead of OPRF-based tokens for algebraic proving properties