PrivateDrop: Practical Privacy-Preserving Authentication for Apple AirDrop

Alexander Heinrich, Matthias Hollick, Thomas Schneider, Milan Stute, and Christian Weinert

privatedrop.github.io
Roadmap to PrivateDrop

Discover
Contact Identifier Leakage by Apple AirDrop

Design
Privacy-Preserving Authentication via Private Set Intersection

Demonstrate
Native Prototype with Excellent User Experience

Discover
SHA256("...@....com")
SHA256("+49...")

Design

Demonstrate
less than 1 second
AirDrop Authentication

[SNMHKNH19]

Find out whether we are mutual contacts via Wi-Fi/AWDL [SKH18]

Sender

I want to find other people, so I tell them who I am

** HTTP POST /Discover with sender’s validation record**

Receiver

* Apple-signed cert including
  $H_i = SHA256(+49 \ 123 ...)$
  $H_j = SHA256(... @icloud.com)$

TLS connection with client and server certificates
AirDrop Authentication

[SNMHKHN19]

∃ \( H_i \in VR: H_i \in address\ book \) 
(+ check validation record 
+ check TLS certificate)

I know the sender, 
so I tell them who I am

* Apple-signed cert including 
\( H_i = SHA256(+49 123 ...) \) 
\( H_j = SHA256(...@icloud.com) \)
AirDrop Authentication

[SNMHKNH19]

*Apple-signed cert including

\[
H_i = SHA256(\{+49 123\})
\]

\[
H_j = SHA256(\{\ldots@icloud.com\})
\]
AirDrop Authentication: What can go wrong here?

Sender

I want to find other people, so I tell them who I am

Receiver

I know the sender, so I tell them who I am

“Hashing vs. Hiding” (Sender Leakage)

\[ H_{S,i} = SHA256(+49 \ 123...) \]

“Celebrity Issue” (Receiver Leakage)

\[ H_{R,i} = SHA256(+1 \ 234...) \]

I know the receiver, so I present him to the user
Exploiting the Vulnerabilities in Practice

Requirements
- Physical proximity to target
- Wi-Fi-capable device

Proof-of-concept
- “AirCollect”
  https://github.com/seemoo-lab/opendrop/blob/poc-phonenumbe-leak/README.PoC.md
- Makes use of optimized rainbow tables [HWSDS21]

Impact
- Recover phone numbers of AirDrop users in real-time
Roadmap to PrivateDrop

Discover
Contact Identifier Leakage by Apple AirDrop

Design
Privacy-Preserving Authentication via Private Set Intersection

Demonstrate
Native Prototype with Excellent User Experience

SHA256("...@....com")
SHA256("+49...")

less than 1 second
Privacy requirements:

1. Disclose contact identifiers only if both parties are mutual contacts.
2. Only disclose those contact identifiers that the other party already knows.

Apply private set intersection (PSI) to achieve private mutual authentication.
Private Set Intersection (PSI)

Sender’s contact identifiers $IDs$

Receiver’s address book $AB$

Intersection $IDs \cap AB$
AirDrop: Semantics

Problems:

- Malicious receivers
- Online complexity depends on AB (large)

\[
\begin{align*}
\text{PSI} & & \text{PSI} \\
\text{IDs} & \rightarrow & \text{AB} & \rightarrow \\
\text{} & \leftarrow & \text{Z} = \text{AB} \cap \text{IDs} & \leftarrow \\
\text{I know R} & & \text{I know S} & \\
\text{} & \leftrightarrow & \text{} & \leftrightarrow \\
\end{align*}
\]

AB: address book
IDs: contact identifiers
PrivateDrop: Semantics

AirDrop Sender $S$  |  AirDrop Receiver $R$

$AB$  |  PSI  |  $IDs$  |  $Z = AB \cap IDs$

$IDs$  |  $Z = AB \cap IDs$  |  PSI  |  $AB$

Changed Semantics:

- Receivers in check
- Online complexity depends on IDs (small)

AB: address book
IDs: contact identifiers

Next: $S$ and $R$ can disclose their known identities, i.e., $IDs \cap AB$
PrivateDrop Design and Implementation

- Maliciously Secure PSI Protocol
- Protection against Malicious Inputs
- Integration of PSI into AirDrop

PrivateDrop Implementation
AirDrop Implementation

Backwards Compatibility
Choice of PSI Protocol

OT-based PSI
[PSZ14,PSZ15,KKRT16,RR17,PRTY20]

Unbalanced PSI

Precomputation Form
[KLS+17,RA18,KRSSW19]

FHE
[CLR17,CHLR18]

PIR-PSI
[DRRT18]

Public-Key Crypto-based PSI
[Sha80,Mea86]

Semi-Honest
[JL09,CKT10a,BBC+11]

Maliciously Secure
[CKT10b,JL10]
Optimized PSI Protocol of [JL10]

**AirDrop Sender** $S$

\[ AB = \{c_1, \ldots, c_n\} \]

**AirDrop Receiver** $R$

\[ Z = AB \cap IDS \]

---

**Precomputation**

\[ k \leftarrow \mathbb{Z}_q \]

For $j = 1$ to $n$:

\[ u = H(H(c_j), H(c_j)^k) \]

For $j = 1$ to $m$:

\[ \alpha_i \leftarrow \mathbb{Z}_q \]

\[ h_i = H(ID_i) \]

\[ y_i = (h_i)^{\alpha_i} \]

Obtain $\checkmark$ for $y_i$

---

**Online**

For $i = 1$ to $m$:

\[ z_i = y_i^k \]

\[ \{u_1, \ldots, u_n\} \]

For $i = 1$ to $m$:

\[ v_i = H(h_i, (z_i)^{1/\alpha_i}) \]

Output $\{ID_i \in IDS | \exists j : u_j = v_i\}$

---

(simplified version, omits ZK proofs for malicious security)
Roadmap to PrivateDrop

**Discover**
Contact Identifier Leakage by Apple AirDrop

**Design**
Privacy-Preserving Authentication via Private Set Intersection

**Demonstrate**
Native Prototype with Excellent User Experience

SHA256("…@….com")
SHA256("+49...")

less than 1 second
PrivateDrop Results: Authentication Delay

• Native implementation for macOS and iOS

• There is some (expected) overhead

• But, authentication delay is well below 1 second (“immediate response”)

Setup: MacBook and iPhone connected via USB cable (results for Wi-Fi connection with stronger variance in the paper)
PrivateDrop: Privacy-Preserving Mutual Authentication for Apple AirDrop

Open-Source Software
Native implementation for macOS and iOS as open-source software available at privatedrop.github.io

Press and Media
International and national coverage

Responsible Disclosure
Apple users are still vulnerable to the discovered privacy leaks
References (1/3)


[HWSDS21] Christoph Hagen, Christian Weinert, Christoph Sendner, Alexandra Dmitrienko, Thomas Schneider. All the Numbers are US: Large-scale Abuse of Contact Discovery in Mobile Messengers. NDSS, 2021.

References (2/3)


References (3/3)


Acknowledgements

• Icons from Font Awesome, Apple Keynote, and Microsoft PowerPoint
• Photo by Ann-Kathrin Braun and Daniela Fleckenstein