Privacy-Preserving Payment Splitting

Saba Eskandarian
Stanford University

Mihai Christodorescu
Visa Research

Payman Mohassel
Facebook
Payment Splitting Apps

Splitwise, Receipt Ninja, Billpin, SpotMe, Conmigo, Settle Up, …

Convenient way to keep track of costs and debts between groups of friends or colleagues
Payment Splitting Apps

Privacy Policy – Data We Collect:

“This data includes, for example, group names, expense descriptions and amounts, payments and their confirmation numbers, comments and reminders, receipt images, notes, and memos, in addition to any other information that you attach or share while using …”

“… the types of expenses you add, the features you use, the actions you take, and the time, frequency and duration of your activities”
Payment Splitting Apps

Privacy Policy – Data We Collect:

“This data includes, for example, group names, expense descriptions and amounts, payments and their confirmation numbers, comments and reminders, receipt images, notes, and memos, in addition to any other information that you attach or share while using …”

“… the types of expenses you add, the features you use, the actions you take, and the time, frequency and duration of your activities”

“Personal Information collected may include your name, age, gender, zip code, e-mail address, cell phone number, occupation, hometown, college, personal interests, nickname, friend’s list and information about personal finances”

“… we may collect and process information about your actual location, like GPS signals sent by a mobile device. We may also use various technologies to determine location, such as sensor data …”

“We also use this information to offer you tailored content – like giving you more relevant search results and ads.”
Payment Splitting Apps

Privacy Policy – Data We Collect:

“This data includes, for example, group names, expense descriptions and amounts, payments and their confirmation numbers, comments and reminders, receipt images, notes, and memos, in addition to any other information that you attach or share while using …”

“… the types of expenses you add, the features you use, the actions you take, and the time, frequency and duration of your activities”

“Personal Information collected may include your name, age, gender, zip code, e-mail address, cell phone number, occupation, hometown, college, personal interests, nickname, friend’s list and information about personal finances”

“… we may collect and process information about your actual location, like GPS signals sent by a mobile device. We may also use various technologies to determine location, such as sensor data …”

“We also use this information to offer you tailored content – like giving you more relevant search results and ads.”

“The app does use third party services that may collect information used to identify you.”
Goal: cash-like privacy for payment splitting
Generic Solutions

Homomorphic encryption based solutions
  [e.g. Gen09, BGV11, GSW13]

Server-aided MPC solutions
  [e.g. FKN94, KMR11/12, HLP11]

Zero-Knowledge Log Server
  [e.g. zkLedger (NVV’18)]

Metadata-hiding anonymous group messaging?
  [e.g. Riposte, Vuvuzela, Stadium, Pung, Atom]
Generic Solutions

Homomorphic encryption based solutions
  [e.g. Gen09, BGV11, GSW13]

Server-aided MPC solutions
  [e.g. FKN94, KMR11/12, HLP11]

Zero-Knowledge Log Server
  [e.g. zkLedger (NVV’18)]

Metadata-hiding anonymous group messaging?
  [e.g. Riposte, Vuvuzela, Stadium, Pung, Atom]

Goal 2: Strong performance and scalability
Our Solution

Same functionality as today’s payment splitting apps

Hides user data from provider

Runs very fast:
  <50ms/round on phone
  <300μs/round on server
  (for realistic group sizes)

Consists mainly of AES and addition
Informal User Survey

Sent to ~250 employees in Visa Palo Alto office, got 51 responses

Some takeaways:
• Groups tend to be small
• Groups have only a few transactions a day
• Transaction amounts are usually fairly small amounts of money
Architecture Overview

Group members connect to server via app
Architecture Overview

Group members connect to server via app

Group members share secret key during setup
Architecture Overview

Group members connect to server via app

Group members share secret key during setup

System proceeds in a series of rounds
Architecture Overview

Group members connect to server via app

Group members share secret key during setup

System proceeds in a series of rounds

Users send vectors of encrypted data each round – either transactions or cover traffic
Architecture Overview

Group members connect to server via app

Group members share secret key during setup

System proceeds in a series of rounds

Users send vectors of encrypted data each round – either transactions or cover traffic

Server *blindly* sums values and sends results (New balance, charger identity, integrity check)
Security Properties

**Server Privacy:** any two sets of transactions indistinguishable to server

**Debtor Privacy:** transaction hides who it puts into debt to others
Security Properties

Server Privacy: any two sets of transactions indistinguishable to server

Debtor Privacy: transaction hides who it puts into debt to others

User Integrity:
   1) No user can create or destroy money (assume >0 honest users)
   2) No user can undetectably frame an honest user for making a charge

Server Integrity: Malicious server can only cause denial of service
Security Properties

**Server Privacy**: any two sets of transactions indistinguishable to server

**Debtor Privacy**: transaction hides who it puts into debt to others

**User Integrity**:

1) No user can create or destroy money (assume >0 honest users)
2) No user can undetectably frame an honest user for making a charge

**Server Integrity**: Malicious server can only cause denial of service

**Limitations**:

We do not hide group membership from the server

We do not protect against collusion between a malicious user and server
Making a Request

Example: Alice requests $1 from Bob in their friend group 🧚‍♀️ 🧚‍♂️ 🧙‍♂️ 🧙‍♀️
Making a Request

Example: Alice requests $1 from Bob in their friend group 🙋‍♀️ 🙋‍♂️ 🙈 🙊 🙏

Alice sets her vector to all 0s except a 1 in Bob’s position

-appointed-doll-face-0-1-0-0-0
Making a Request

Example: Alice requests $1 from Bob in their friend group.

Alice sets her vector to all 0s except a 1 in Bob’s position:

Anyone not making a charge puts a 1 in their own position:
Making a Request

Example: Alice requests $1 from Bob in their friend group

Alice sets her vector to all 0s except a 1 in Bob’s position

Anyone not making a charge puts a 1 in their own position

Each user encrypts his/her vector and sends the result to the server
Making a Request

Example: Alice requests $1 from Bob in their friend group

Alice sets her vector to all 0s except a 1 in Bob’s position

Anyone not making a charge puts a 1 in their own position

We’ll start by showing the protocol without encryption
Making a Request

The server adds up everyone’s values and subtracts 1

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

+       +       +       +       +

| -1     | -1     | -1     | -1     |       |

=       =       =       =       =

| -1     | 1      | 0      | 0      | 0      |
Making a Request

The server adds up everyone’s values and subtracts 1

The result is added to users’ existing balances

Note: server tracks *debt*, so negative is less debt
Tracing Charges

How does Bob know it was Alice who charged him?
Tracing Charges

How does Bob know it was Alice who charged him?

For each user, server takes “(input in user’s own position) – 1”
Tracing Charges

How does Bob know it was Alice who charged him?
For each user, server takes “(input in user’s own position) – 1”

0 → -1
1 → 0
1 → 0
1 → 0

Multiplies by a power of 2 assigned to that user

x1 = -1 + x2 = 0 + x4 = 0 + x8 = 0
Tracing Charges

How does Bob know it was Alice who charged him?

For each user, server takes “(input in user’s own position) – 1” Multiplies by a power of 2 assigned to that user

\[
x_1 = -1 + x_2 = 0 + x_4 = 0 + x_8 = 0
\]

And sums up the results to identify the charger(s)

\[
= -1 \rightarrow \text{Alice}
\]
Tracing Charges

For each user, server takes “(input in user’s own position) – 1” Multiplies by a power of 2 assigned to that user
And sums up the results to identify the charger(s)

Examples

-1  0  0  0
x1 + x2 + x4 + x8 = -1 →

-1  0  0 -1
x1 + x2 + x4 + x8 = -9 →
Tracing Charges

For each user, server takes “(input in user’s own position) – 1” 

*Multiplies* by a power of 2 assigned to that user 

And sums up the results to identify the charger(s) 

**Examples**

-1         0           0          0
x1   +   x2   +    x4   +   x8 = -1 →

-1         0           0          -1
x1   +   x2   +    x4   +   x8 = -9 →

What to do for collisions? Roll back and repeat one by one (server is oblivious)
Adding Server Privacy

Observation 1: server does the same fixed set of additions every round.

No data-dependent operations
Adding Server Privacy

**Observation 1:** server does the same *fixed* set of additions every round.

No data-dependent operations

**Observation 2:** all the clients share a secret key $k$.

They can independently generate the same PRF outputs
Adding Server Privacy

Solution:
Instead of actually encrypting, users mask values with a PRF output.
They send $v_i + r_i$, where $r_i = PRF(k, \text{group, user, round, } i)$

Users calculate sum of masks and remove them from server responses.
Calculating/removing masks fast because it’s just AES and addition.
Extensions

• Integrity
• Larger transactions
• Multiple charges per Round
• Identifying misbehaving users
• Handling framing
• Handling users going offline
• Improving usability for charge requests
• Integration with payment systems
• Payment splitting with collateral

See paper for details!
Extensions

- Integrity
- **Larger transactions**
- **Multiple charges per Round**
- Identifying misbehaving users
- Handling framing
- Handling users going offline
- Improving usability for charge requests
- Integration with payment systems
- Payment splitting with collateral

See paper for details!
Extensions

- Integrity
- Larger transactions
- Multiple charges per Round
- Identifying misbehaving users
- Handling framing
- Handling users going offline
- Improving usability for charge requests
- Integration with payment systems
- Payment splitting with collateral

See paper for details!
Performance
Client Performance

<50ms/round for realistic groups (realistic based on user survey)

Malicious server overhead <20ms

Only computes AES and addition

Client bandwidth for group size:
- 10 (≥69% of groups in survey): 160 Bytes
- 25 (≥92% of groups in survey): 400 Bytes
- 100 (≥100% of groups in survey): 1.6Kb
Server Performance

<300 microseconds for realistic groups (realistic based on user survey)

No changes for malicious security

Only computes addition

Server memory requirements small – can handle user inputs as they arrive, no need to keep in memory

See paper for more evaluation details
Summary

Our system allows payment-splitting groups to hide
- Who pays,
- Who is paid,
- How much is spent,
- When transactions are made,
- And more

From a potentially malicious server at minimal performance cost

Contact: saba@cs.stanford.edu