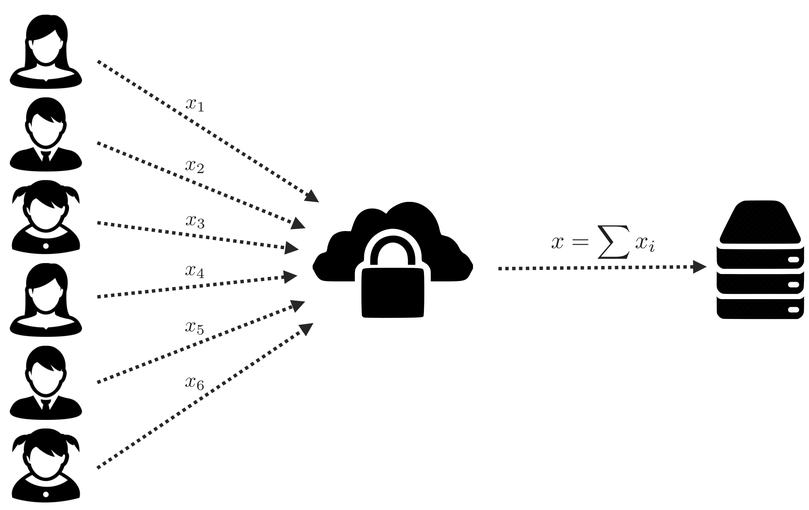
Private Data Aggregation on a Budget

Morten Dahl[1] Valerio Pastro[2] Mathieu Poumeyrol[1]

[1]Snips [2]Yale University

Private Aggregation

A single company running a **server** (right) wishes to learn aggregate $x = \sum x_i$ of inputs $x_i \in \mathbb{Z}_p^d$ held by a set of **users** (left). Secure computation in the middle to ensure server does not learn anything about the individual inputs.



Challenges

Weak devices: *Users only have computationally weak devices such as smart phones and web browsers.* We keep requirements as low as possible and spread the workload.

Limited connectivity: *Devices have unreliable and low bandwidth communication channels.* We keep interaction, session length, and presence to a minimum.

Single server: *Only company investing significant resources in the computation (standard MPC not applicable).* We optimise outsourcing in server-aided model.

Robustness: *User device availability is sporadic and might disappear entirely before completion.* We employ tweakable redundancy to guarantee output delivery.

Application: *Aggregation through summation requires suitable representation of user inputs, which may significantly increase their dimension.* We investigate impact and techniques in several real-world use-cases.

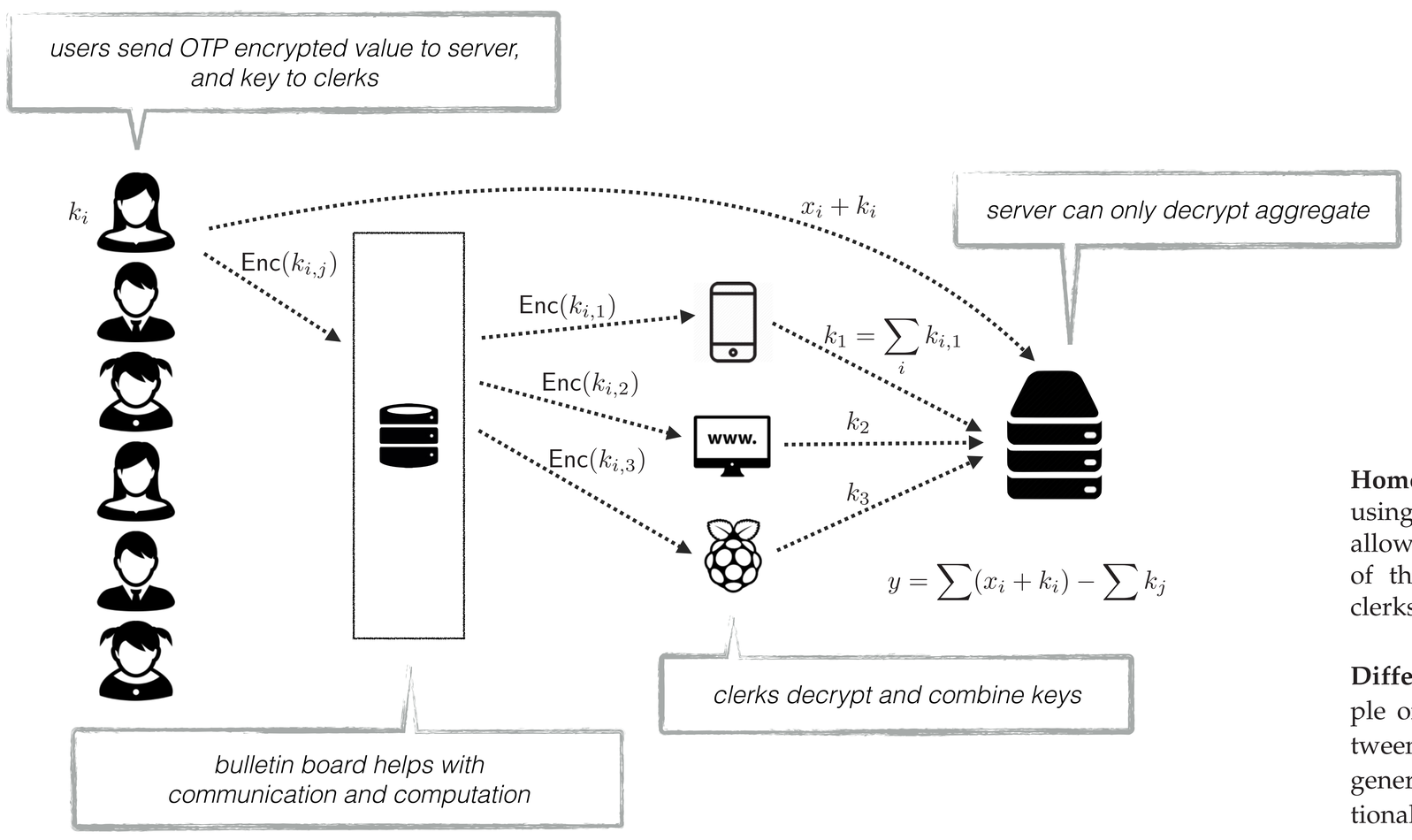
Protocol

From left to right: **users**, **bulletin board**, **clerks**, and **server**.

User U_i picks random key $k_i \in \mathbb{Z}_p^d$, sends $x_i + k_i$ to the server, and sends a secret sharing of k_i to the clerks; latter step is done by posting the shares to the bulletin board, encrypted under the public key of each clerk.

To aggregate, each clerk downloads its encrypted shares from the bulletin board, decrypts, and sends the sum of them to the server.

Finally, the server can recover the key and decrypt only the aggregate.

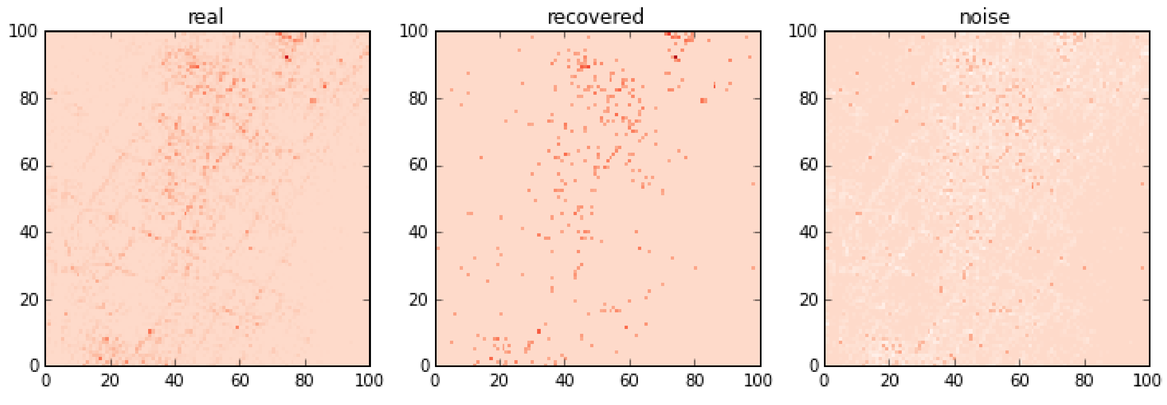


Homomorphic encryption: using an additive scheme allows for outsourcing most of the work done by the clerks to the bulletin board.

Differential privacy: a simple one-round protocol between clerks allows them to generate "noise" for additional input privacy.

Use-case: Heatmaps

We seek to discover areas of interest by a specific demographic in a region the size of New York City. By dividing it into a grid of approximately 40×40 meters we get $160,000$ boxes and an equivalent input dimension d. To lower the workload on clerks we instead approximate the result using a *Min-Count Sketch*, thereby reducing the dimension to $20,000$. We use a Foursquare dataset to test the accuracy of this approach on a realistic place distribution, with the aim of identifying heavy hitters. This is illustrated for a simpler case below, giving the real heatmap (right) and the heatmap recovered from the sketch (middle).



Summary

We develop a practical secure multi-party computation protocol allowing a company to privately aggregate inputs from mobile devices and web browsers. We account for the characteristics of these devices and outsource as much as possible to the company. We benchmark against several realistic data sets.

Use-case: Analytics

Goal is to count occurrences of 100 different events in web or mobile applications, and test how many users the protocol (using a non-homomorphic scheme) supports under the requirement that the *download* for each clerk remains small, i.e. no more than 1MB. Note that as the number of users increases the download size can be kept below the limit by simply increasing the number of clerks, without significant affecting the *upload* size for each user.

#users	#clerks	upload	download
$25,000$	26	1KB	977KB
$80,000$	80	1KB	938KB
$250,000$	728	3KB	977KB

Use-case: Ratings

The Netflix Prize aggregation gives ratings to $17,770$ movies using roughly 100 million inputs. This amount of data results in each clerk spending a significant amount of time *decrypting*, but by increasing the number of clerks we can spread the workload and bring it down to a reasonable level, without significantly affecting the time each user spends *sharing* its input.

device	#clerks	sharing	decrypting
iPhone 6	26	1186ms	$15,057$ms
iPhone 6	80	791ms	$3,174$ms
iPhone 6	728	877ms	414ms