Private Set Intersection (PSI)

Introduction, ORPF/Diffie-Hellman based PSI, and the Apple CSAM detection system

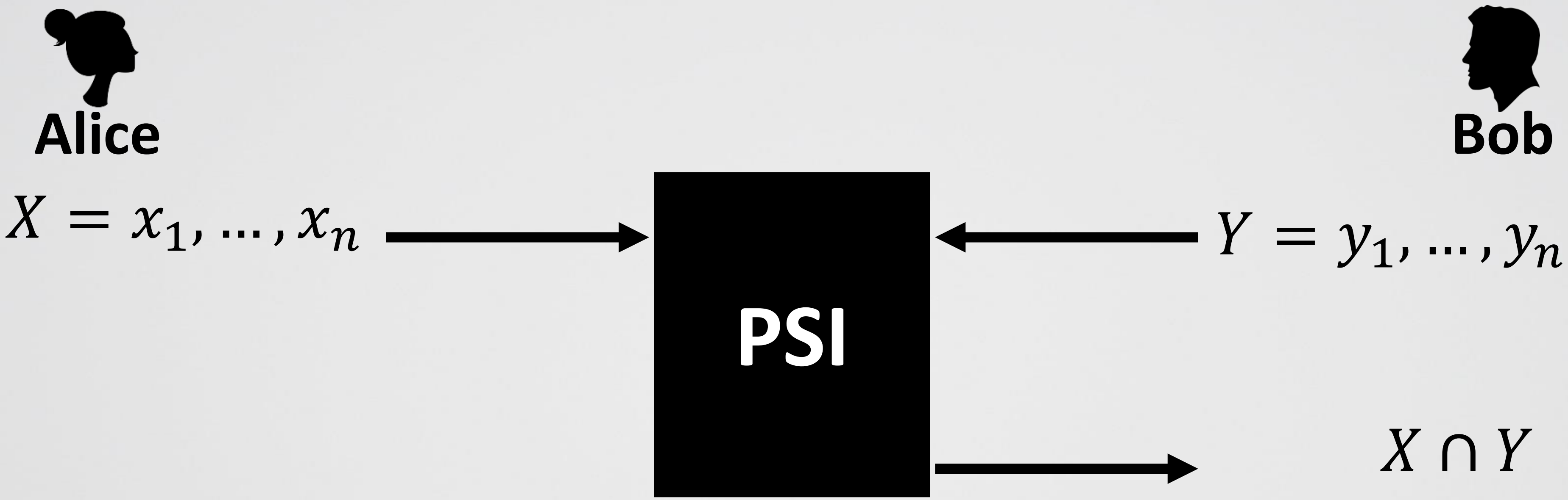
Benny Pinkas, Bar-Ilan University

In this talk

- PSI Introduction
- Diffie-Hellman based PSI, and PSI from OPRF
- The PSI system used by Apple for CSAM detection

Private Set Intersection (PSI)

Definition



Can also compute a function of the intersection, e.g. size (cardinality), or whether the intersection size is great than some threshold

Applications of PSI

- **Information sharing**, e.g., intersection of threat information
- **Matching**, e.g., testing compatibility of different properties (preferences, genomes…)
- **Join** DB operations
- **Analytics:** $Pr(A / B) = Pr(A \cap B) / Pr(B)$
- **Identifying mutual contacts** (Signal app)
- **Computing ad conversion rates** (Google)

PSI Example – simple analytics for COVID vaccinations

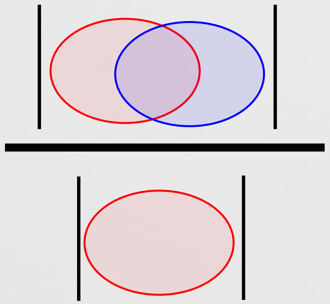
University

workers & students ids

Ministry of Health

vaccinated people

Need to compute



Multi-party PSI

How can multiple fertility clinics compare their donor records?

Customer Risk Analysis

small merchant
(Alice)

merchants, banks

Input: new customer

customer databases

Output: do you know this guy?

Customer Risk Analysis

small merchant
(Alice)

merchants, banks

Here Alice's set only contains one item!

In more detail:

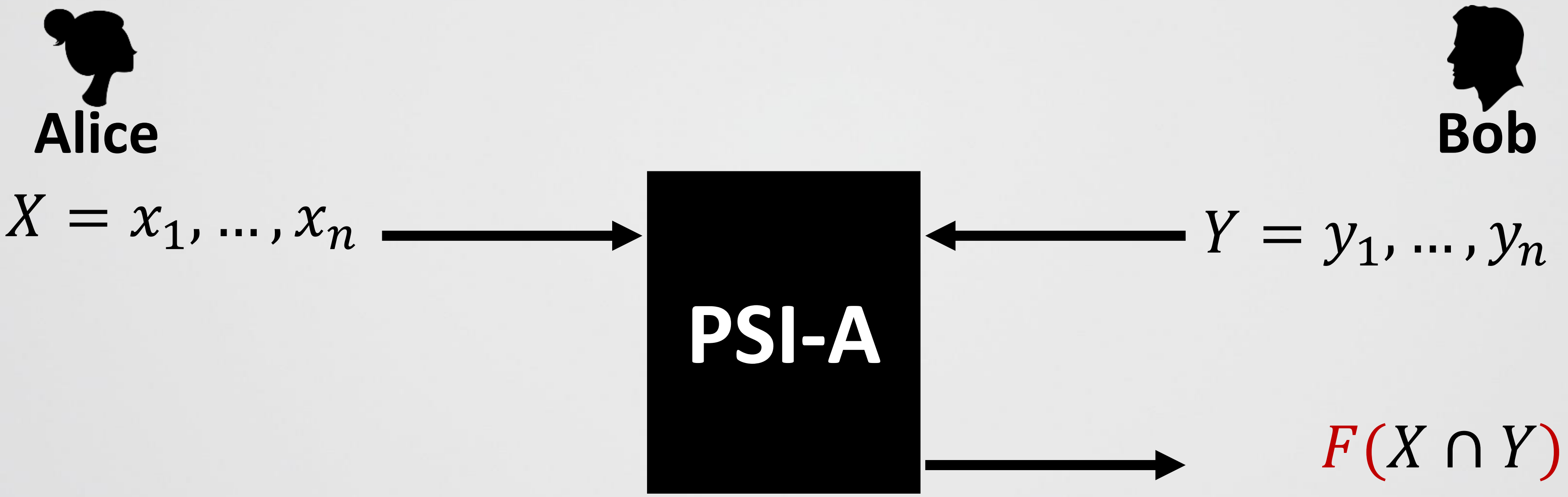
- Alice has a new customer x
- Each other member has a set of known customers and their rating
- Alice learns the rating of x by the other members
- The other members do not learn x

- Or, Alice receives a purchase request to a certain address. She can check if purchases with of this credit card with the other members were sent to the same address.

Private Set Intersection (PSI) + Analytics

Definition

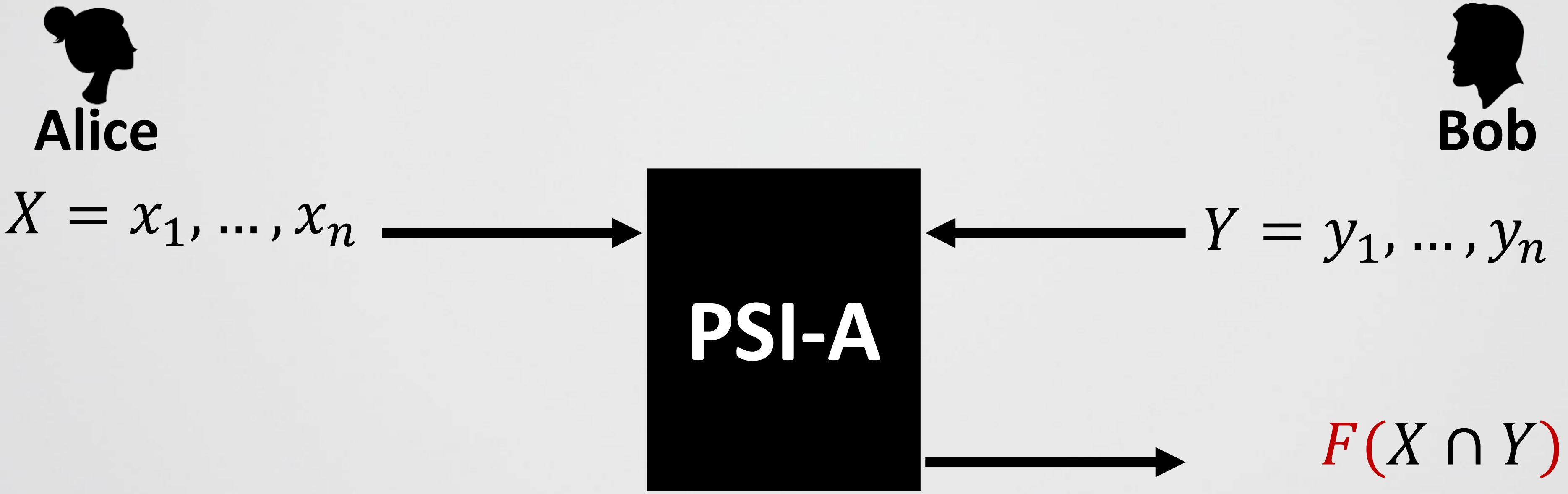
1. Post processing function F.   E.g. $F = |X \cap Y|$



Private Set Intersection (PSI) + Payload Analytics

Definition

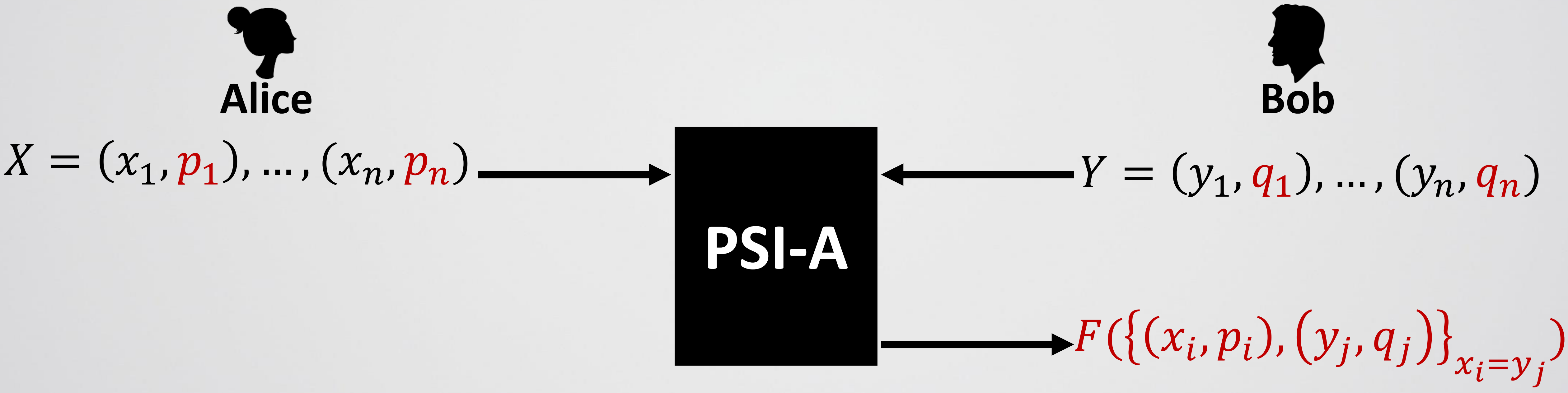
1. Post processing function F. E.g. $F = |X \cap Y|$
2. Items are associated with payloads (aka. PSI with data-transfer)



Private Set Intersection (PSI) + Payload Analytics

Definition

1. Post processing function F. E.g. $F = |X \cap Y|$
2. Items are associated with payloads (aka. PSI with data-transfer)



Private Set Intersection (PSI) - Analytics

Applications

Private Intersection-Sum Protocol with Applications to Attributing Aggregate Ad Conversions

Mihaela Ion[†], Ben Kreuter[†], Erhan Nergiz[†], Sarvar Patel[†], Shobhit Saxena[†], Karn Seth[†], David Shanahan[†] and Moti Yung[‡*]

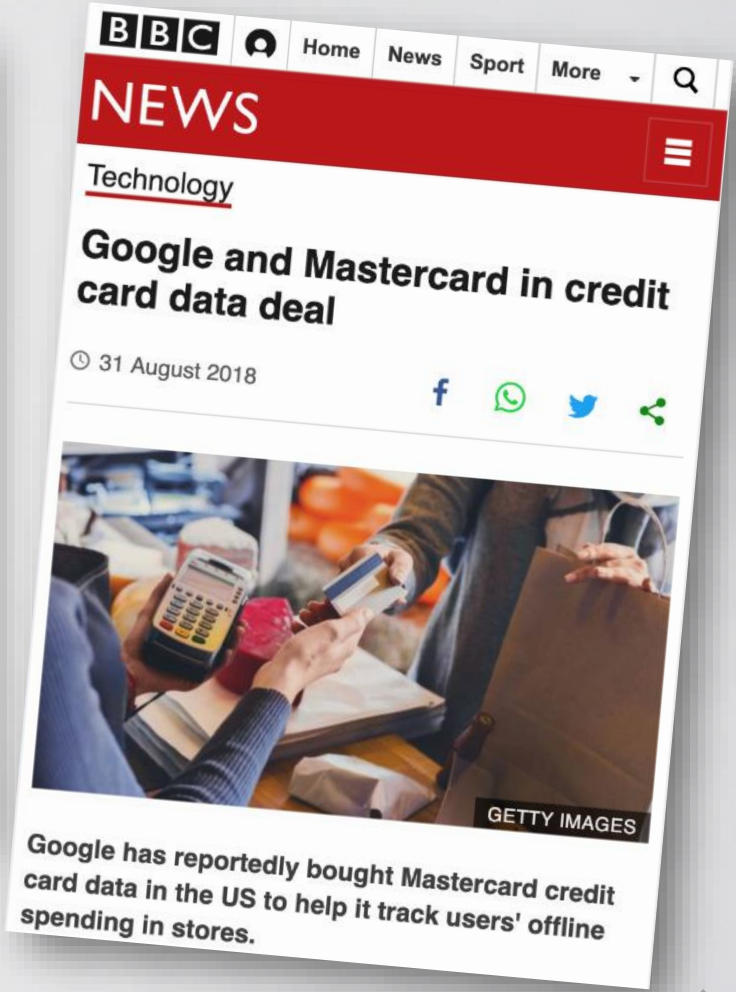
[†]{mion, benkreuter, anergiz, sarvar, shobhitsaxena, karn, dshanahan}@google.com

Google Inc.

[‡]moti@cs.columbia.edu

Columbia University and Snap Inc.

July 31, 2017


BBC Home News Sport More

NEWS

Technology

Google and Mastercard in credit card data deal

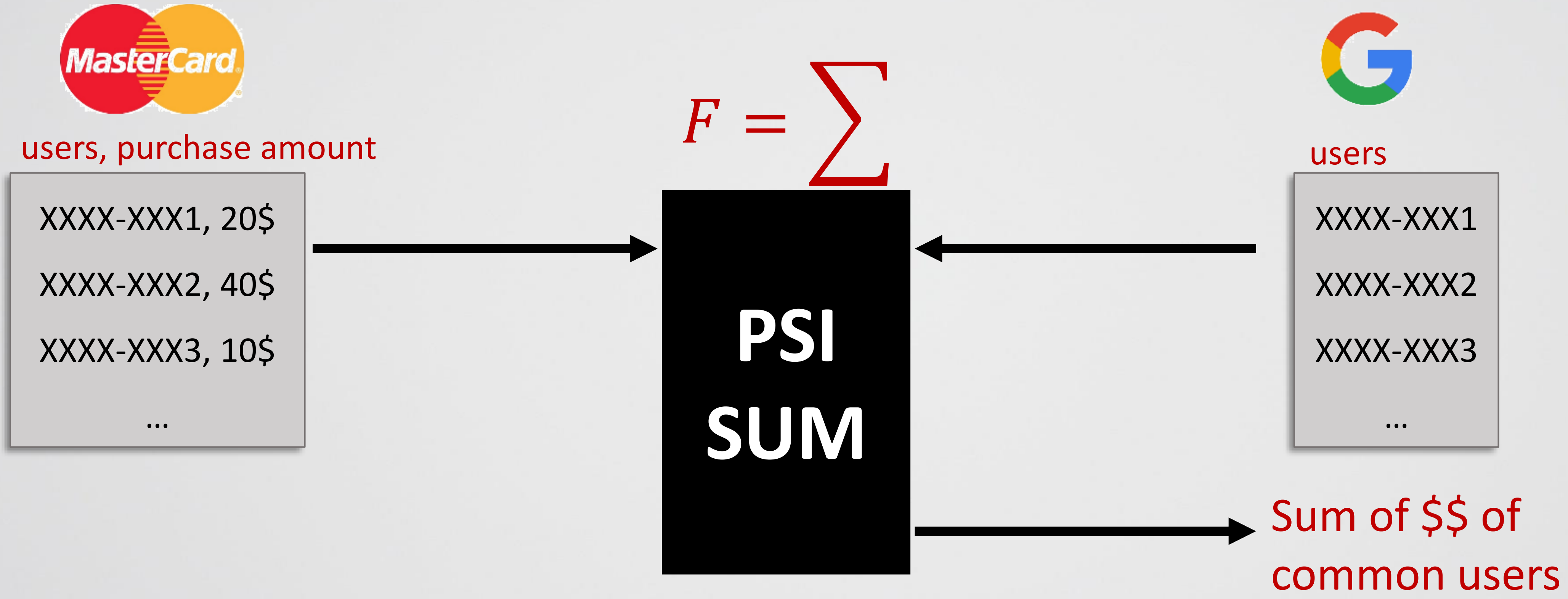
31 August 2018

GETTY IMAGES

Google has reportedly bought Mastercard credit card data in the US to help it track users' offline spending in stores.

Private Set Intersection (PSI) - Analytics

Applications: Online Ads to Offline Purchase Conversion



Why did I investigate PSI?

- Relevant to many applications
- Cannot be efficiently solved using generic MPC (circuits)
- Solutions turned out to be based on interesting techniques

Pseudo-random function (PRF)

- Think of $AES_K(x)$
  - If we don't know the key k, the output looks random

- A PRF is a keyed function $F_k(\ )$
- If k is chosen at random and kept secret, then a polynomial time program cannot distinguish between the output of $F_k(\ )$ and a truly random function. (Even if it knows or chooses the inputs.)

Oblivious Pseudo-Random Function (OPRF)

A protocol for securely computing a PRF

Oblivious Pseudo-Random Function (OPRF)

Another OPRF version

Useful when there is no need for the server to choose the key K (as in PSI)

A pseudo-random function based on DDH

- Let G be a group in which the DDH assumption holds
- Let H() be a publicly known random function mapping into G
  - H() is modeled as a random oracle
- K is a key

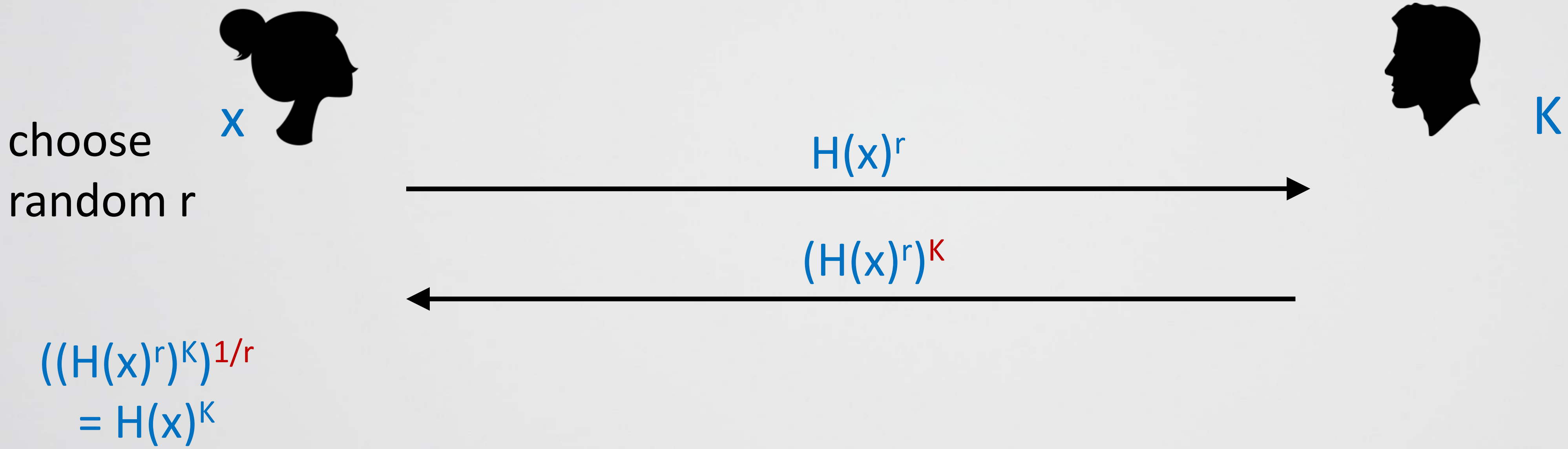
Typically implemented in ECC

DDH assumption: for generator g and random *a,b,c* cannot distinguish (g^a, g^b, g^{ab}) from (g^a, g^b, g^c)

The function $F_K(x) = (H(x))^K$ is pseudo-random

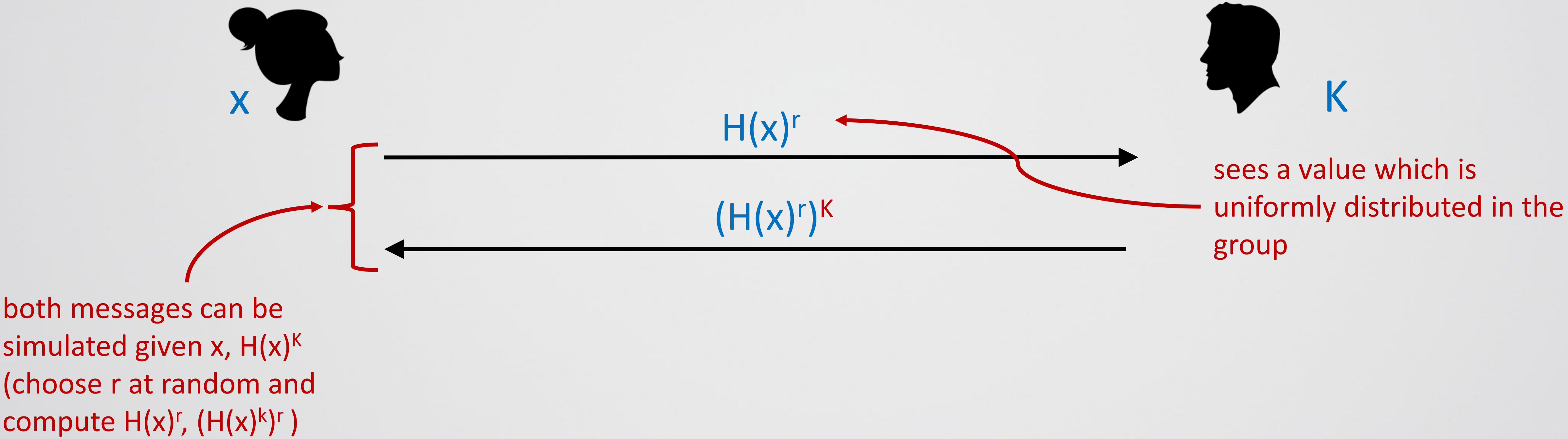
An OPRF construction based on DDH

A protocol for securely computing the PRF $F_K(x) = (H(x))^K$



An OPRF construction based on DDH

Security intuition:



An OPRF construction based on DDH

- Becoming an IETF standard
  - `Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups draft-irtf-cfrg-voprf-01`
  - Implemented in elliptic curve groups
- Also a "verifiable OPRF" version
  - The server can prove that in all invocations it used the same key
    - The server publishes g^k
    - With each answer it proves that the discrete log of $(H(x)^r)^k$ to the base $(H(x)^r)$, is the same as the discrete log of g^k to the base g.

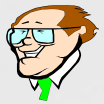
PSI based on DDH (described using OPRF)

α is a randomization used by Alice
β is a OPRF key known to Bob
PRF is $H(x)^{\beta}$

Typically implemented in ECC

α 

$x_1,\ldots,x_n$

 β

$y_1,\ldots,y_n$

$(H(x_1))^{\alpha},\ldots,(H(x_n))^{\alpha}$ $\longrightarrow$

$\longleftarrow$ $\boxed{(H(y_1))^{\beta},\ldots,(H(y_n))^{\beta}}$ PRF of Bob's inputs

$\longleftarrow$ $((H(x_1))^{\alpha})^{\beta},\ldots,((H(x_n))^{\alpha})^{\beta}$ OPRF protocol

$\boxed{(((H(x_1))^{\alpha})^{\beta})^{1/\alpha},\ldots,(((H(x_n))^{\alpha})^{\beta})^{1/\alpha}}$

Compares the two lists

Security intuition

- Recall a trivial **insecure** PSI protocol (known hash)

$x_1,\ldots,x_n$

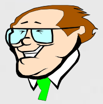
$\mathrm{SHA}(x_1),\ldots,\ \mathrm{SHA}(x_n) \longrightarrow$

- This protocol is insecure against a dictionary attack
- Whereas in the OPRF-PSI protocol we described,
  - Bob sees the output of a hash that can only be computed by Alice $(H(x))^{\alpha}$
  - Alice sees the output of a hash that can only be computed by Bob $(H(y))^{\beta}$

Bob simulating his view

α

$x_1,\ldots,x_n$



β

$y_1,\ldots,y_n$

$(H(x_1))^\alpha,\ldots,(H(x_n))^\alpha \longrightarrow$

Pseudorandom

$\longleftarrow (H(y_1))^\beta,\ldots,(H(y_n))^\beta$ PRF of Bob's inputs

$\longleftarrow ((H(x_1))^\alpha)^\beta,\ldots,((H(x_n))^\alpha)^\beta$ OPRF protocol

$(((H(x_1))^\alpha)^\beta)^{1/\alpha},\ldots,(((H(x_n))^\alpha)^\beta)^{1/\alpha}$

Compares the two lists

Alice simulating her view

α   β

$x_1,\ldots,x_n$ $y_1,\ldots,y_n$

$(H(x_1))^\alpha,\ldots,(H(x_n))^\alpha \longrightarrow$

$\longleftarrow (H(y_1))^\beta,\ldots,(H(y_n))^\beta$ **Pseudorandom**

$\longleftarrow ((H(x_1))^\alpha)^\beta,\ldots,((H(x_n))^\alpha)^\beta$ **Equal to $(((H(y_j))^\beta)^\alpha$ for matching elements, pseudorandom otherwise**

$(((H(x_1))^\alpha)^\beta)^{1/\alpha},\ldots,(((H(x_n))^\alpha)^\beta)^{1/\alpha}$

Compares the two lists

Template for PSI based on OPRF (basic version)

$x_1,\ldots,x_n$ 

 $y_1,\ldots,y_n$

K is a OPRF key known to Bob

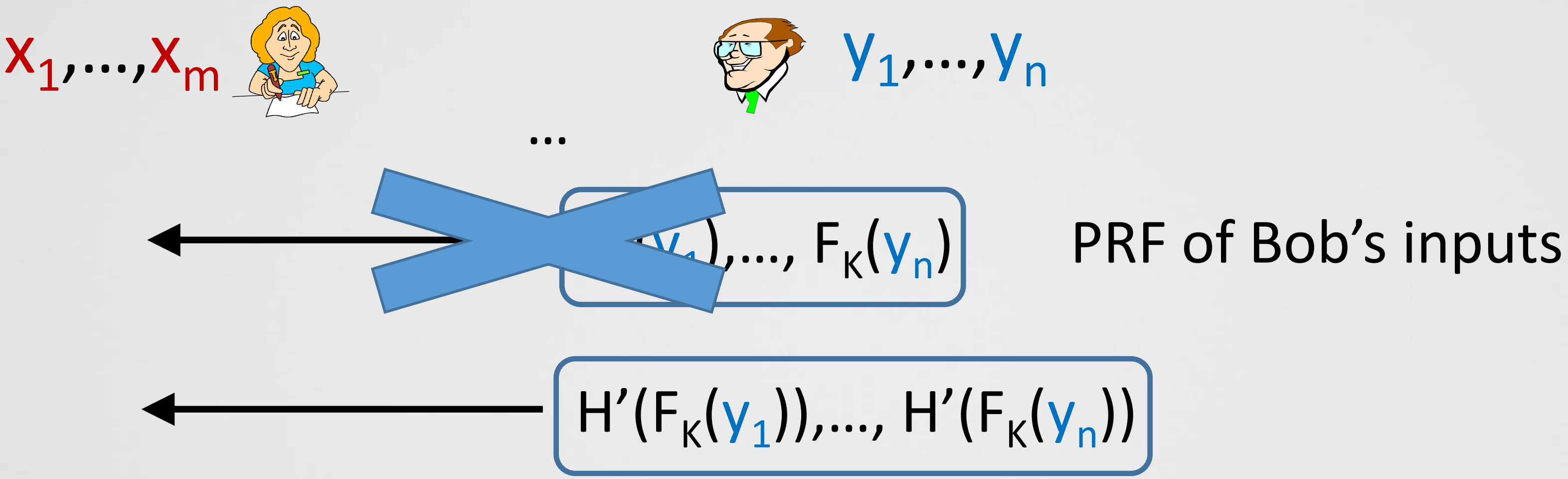
$x_1,\ldots,x_n$ → **OPRF** ← K

$F_K(x_1),\ldots, F_K(x_n)$

$F_K(y_1),\ldots, F_K(y_n)$ PRF of Bob's inputs

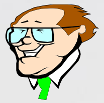
Compares the two lists

Sending the PRF values



- Instead, apply a hash function with a **shorter output** and send the results
- The output length of H'() depends on desired false positive probability, set sizes, and the data structure.
- Can also encode the results in an efficient data structure supporting searches (dictionary), such as a Bloom filter or a cuckoo filter

PSI of sets of unequal sizes

 $y_1,\ldots,y_n$

K is a OPRF key known to Bob

$\longleftarrow D(F_K(y_1),\ldots, F_K(y_n))$

Dictionary encoding of the PRF outputs of Bob's inputs

$x_1,\ldots,x_m$

$\forall\, x_i$ {

$x_i \longrightarrow$ **OPRF** $\longleftarrow K$

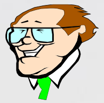
$F_K(x_i) \longleftarrow$ **OPRF**

checks if $F_K(x_i)$ is in D

}

Center for Research in Applied Cryptography and Cyber Security

PSI of sets of unequal sizes



$y_1,\ldots,y_n$

K is a OPRF key known to Bob

(before Alice learns her inputs)

$\longleftarrow$ $D(F_K(y_1),\ldots, F_K(y_n))$

Dictionary encoding of the PRF outputs of Bob's inputs

$x_1,\ldots,x_m$

$\forall\, x_i$

$x_i \longrightarrow$ **OPRF** $\longleftarrow K$

$F_K(x_i) \longleftarrow$ **OPRF**

checks if $F_K(x_i)$ is in D

checks for a match whenever Alice has a new x_i

Implementations of OPRFs

Some applications need
- Verifiable OPRF
- Malicious security

- Based on DH and random oracle H() (described earlier)
- Based on DH alone (Naor-Reingold)
- Based on RSA
- Based on 2-party MPC of AES or other ciphers (communication depends on circuit size)
- Post quantum secure:
  - Based on the ring learning-with-errors problem and the short-integer-solution problem in one dimension
  - Based on isogenies of supersingular elliptic curves

Apple’s CSAM Detection System
(my interpretation of it)

"May your research area be relevant to current affairs" is the "May you live in interesting times" curse-disguised-as-a-blessing for the academic set” - Riana Pfefferkorn

Background

- CSAM = Child Sexual Abuse Material
- All companies, including Apple, allow law enforcement to access cloud data, if they are provided the right warrant
  - But this is different than scanning **all** data
- Apple wanted / was required to scan uploaded photos (not on phones or in messaging)

The Internet Is Overrun With Images of Child Sexual Abuse. What Went Wrong?

Online predators create and share the illegal material, which is increasingly cloaked by technology. Tech companies, the government and the authorities are no match.

Last year, tech companies reported over 45 million online photos and videos of children being sexually abused — more than double what they found the previous year.

Current industry practices for CSAM detection

Scan data (photos) in the clear

NCMEC

Facebook, Google, etc.
(but not Apple)

Only looking for matches with **known** content. **Not analyzing** your photos.

Apple's design objectives

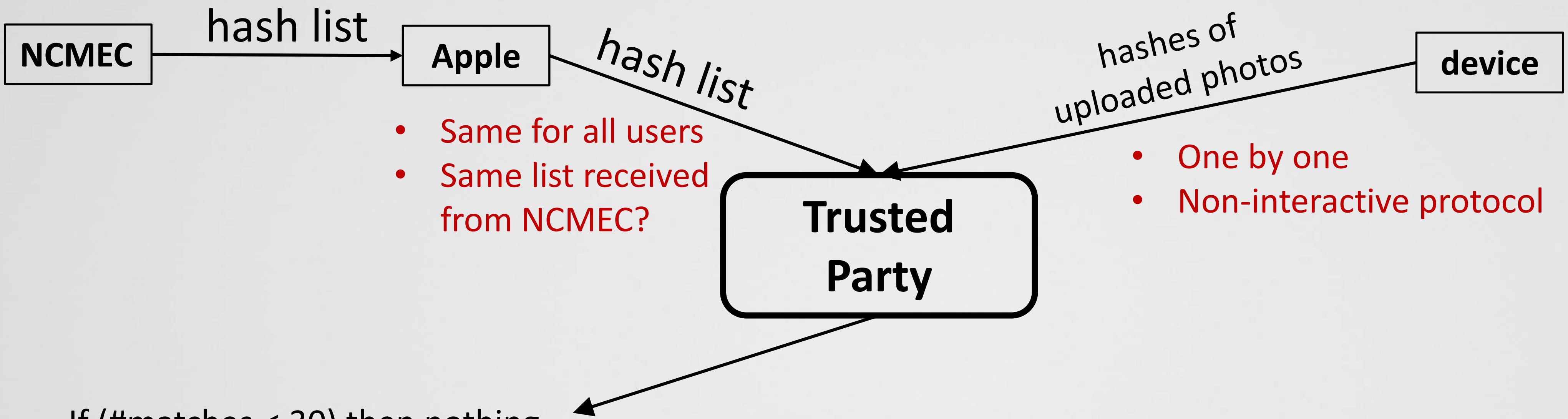
- Apple learns nothing about images that do not match **known** CSAM
- Apple can't access any data of matched CSAM images until a **threshold** of 30 matches is exceeded for an iCloud account
- The risk of the system incorrectly flagging an account is extremely low
- In addition, Apple **manually** reviews all reports made to NCMEC to ensure reporting accuracy

Protecting users

- Users can't access or view the database of known CSAM images/hashes
- Users can't identify which images were flagged as CSAM by the system

Protecting society

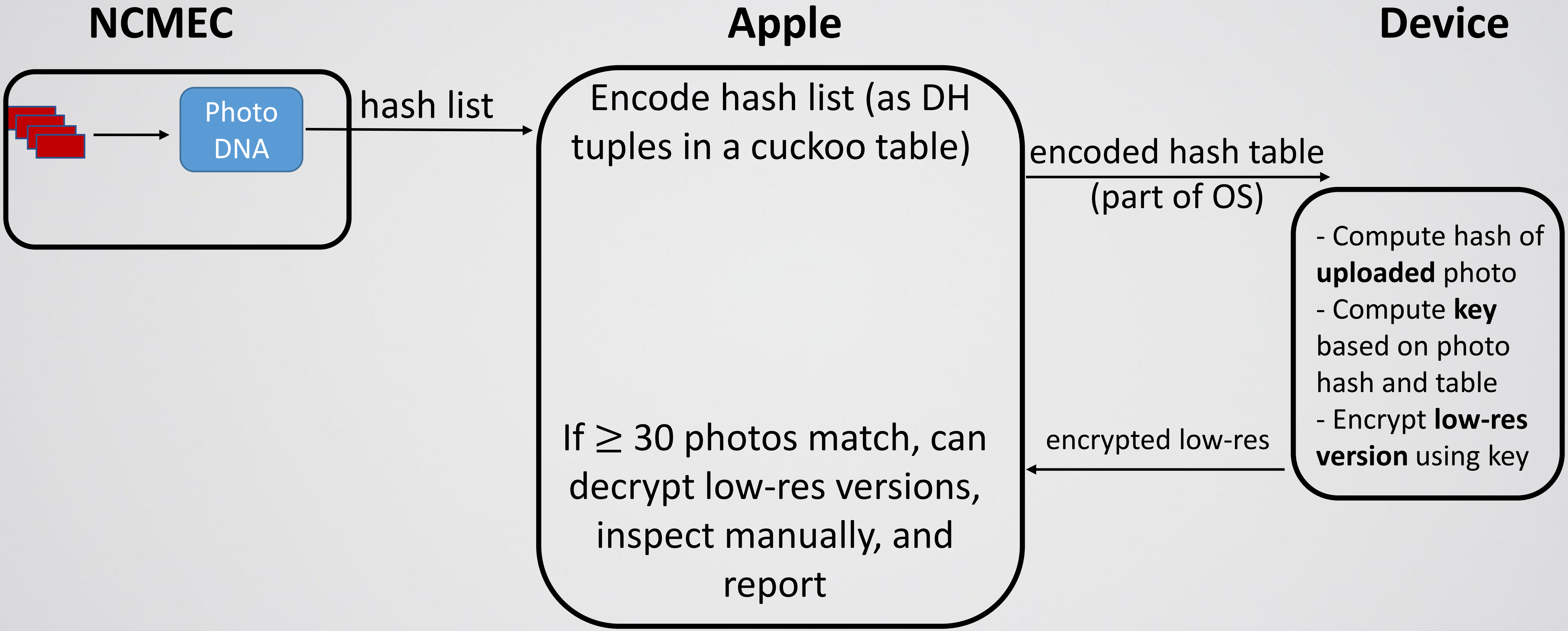
Ideal world view



If (#matches < 30) then nothing
Otherwise, identities of matches

Needs additional inspection by humans

Apple's CSAM scanning system



Apple’s CSAM scanning system

NCMEC

Photo DNA

hash list

Apple

Encode hash list (as DH tuples in a cuckoo table)

If $\geq$ **30** photos match, can decrypt low-res versions, **inspect manually**, and report

encoded hash table (**part of OS**)

encrypted low-res

Device

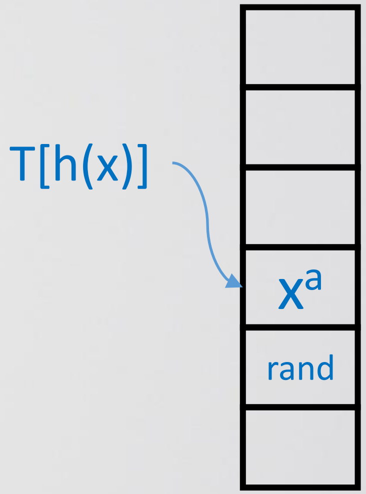
- Compute hash of **uploaded** photo
- Compute **key** based on photo hash and table
- Encrypt **low-res version** using key

- Device does not learn hash list
- Device does not learn if an uploaded photo is a match

- Apple does not learn anything about non-matched photos

Server preprocessing

- Choose a random secret key (exponent) a
- Construct a (cuckoo hash) table storing the CSAM hash list
- For every hash x in the CSAM list, store x^a in location $T[h(x)]$
  - (This is actually a cuckoo table with two hash functions)
  - Elements that do not find a place are dropped
- Put random values in empty entries
  - Since $F(x)=x^a$ is a prf (x is already the output of a hash function), this is **indistinguishable** without knowledge of the key a
- Send **table** and g^a to client (as part of the operating system)



DH self reduction [Naor-Reingold]

- $(g,\ L=g^a,\ T=g^b,\ P=g^{ab})$ is a DH tuple
- Pick random c,d
- $Q = g^d T^c, \quad S = L^d P^c$
- If (g,L,T,P) is a DH tuple then, since $L=g^a$ and $P=T^a$, we get that $S=Q^a$
- If (g,L,T,P) is **not** a DH tuple then (Q,S) is a uniformly random pair

Client side

- Client has photo it wants to upload (**x** = hash, **id**, **data** = associated data)
  - Picks random **c,d**
  - Computes $\mathbf{Q = x^c g^d}$ and $\mathbf{S = T[h(x)]^c (g^a)^d}$
  - If $\mathbf{T[h(x)] = x^a}$ then $\mathbf{S=Q^a}$. Otherwise, based on DH self reducibility, **Q** and **S** are a uniformly random pair of values.
  - Idea: Send **Q** and use **S** as a **key**. If this is a DH tuple then the server, which knows **a**, can recover key **S** as $\mathbf{Q^a}$. Otherwise, **S** is uniformly random.

- Security:
  - Client sees the table, which looks pseudo-random without knowledge of key a
  - If x is **not** in table, client encrypts with a key **S** which is random.

Threshold PSI

- Client Picks a single random master key **adkey** for all items.
- Client has input item (**x** = hash, **id**, **data** = associated data)
  - Computes **Enc(adkey, data)**
  - Generates a Shamir **share** of **adkey** with threshold **t = 30**
  - Computes **Q,S** as described before
  - Sends **Q** and authenticated encryption with key **S** of **(share, Enc(adkey, data))**

- If x is in hash table, server knows **S** and can decrypt **(share, Enc(adkey, data))**
- After 30 such shares, server can recover **adkey** and decrypt data
  - But we do not want to the server to identify which users have 29 shares, etc.

Threshold PSI

- For users who have < 30 matches, need to hide #matches from server
- Solution: users send at random dummy matches – messages for which server can decrypt **(dummy-share, dummy-Enc)**
- The server will have some real shares and some dummy shares. It must be able to decrypt whenever #real shares ≥ 30.

Threshold PSI

- The server will now have some real shares and some dummy shares. It must be able to decrypt whenever #real shares $\geq$ 30.

- A solution based on a decoding algorithm of Coppersmith-Sudan
  - Given t correct "Shamir+" shares and s random shares, can decode secret
  - Caveat: The size of each share is larger by a factor of s+1
  - E.g., for t=30 and s=100, each share contains 101 field elements…
  - Doesn't matter for this application, since the share size is "negligible" compared to the size of the uploaded photo

Associated data

- The associated data includes a low resolution “visual derivative” of the image.
- If 30 matches are found, human operators examine these visual derivatives, and if they look like CSAM then the user is reported.

- (associated data size >> data sent in PSI protocol)

Potential problems beyond the technical solution

- False positives
- Apple is scanning on my device
- Who controls which items are searched for? Governments coerce Apple into searching for specific photos
- “A first crack in the dam”

Relevant papers

- https://www.apple.com/child-safety/
- In particular
  - Technical description https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf
  - Another technical description https://decentralizedthoughts.github.io/2021-08-29-the-private-set-intersection-psi-protocol-of-the-apple-csam-detection-system/
  - Threat model https://www.apple.com/child-safety/pdf/Security_Threat_Model_Review_of_Apple_Child_Safety_Features.pdf