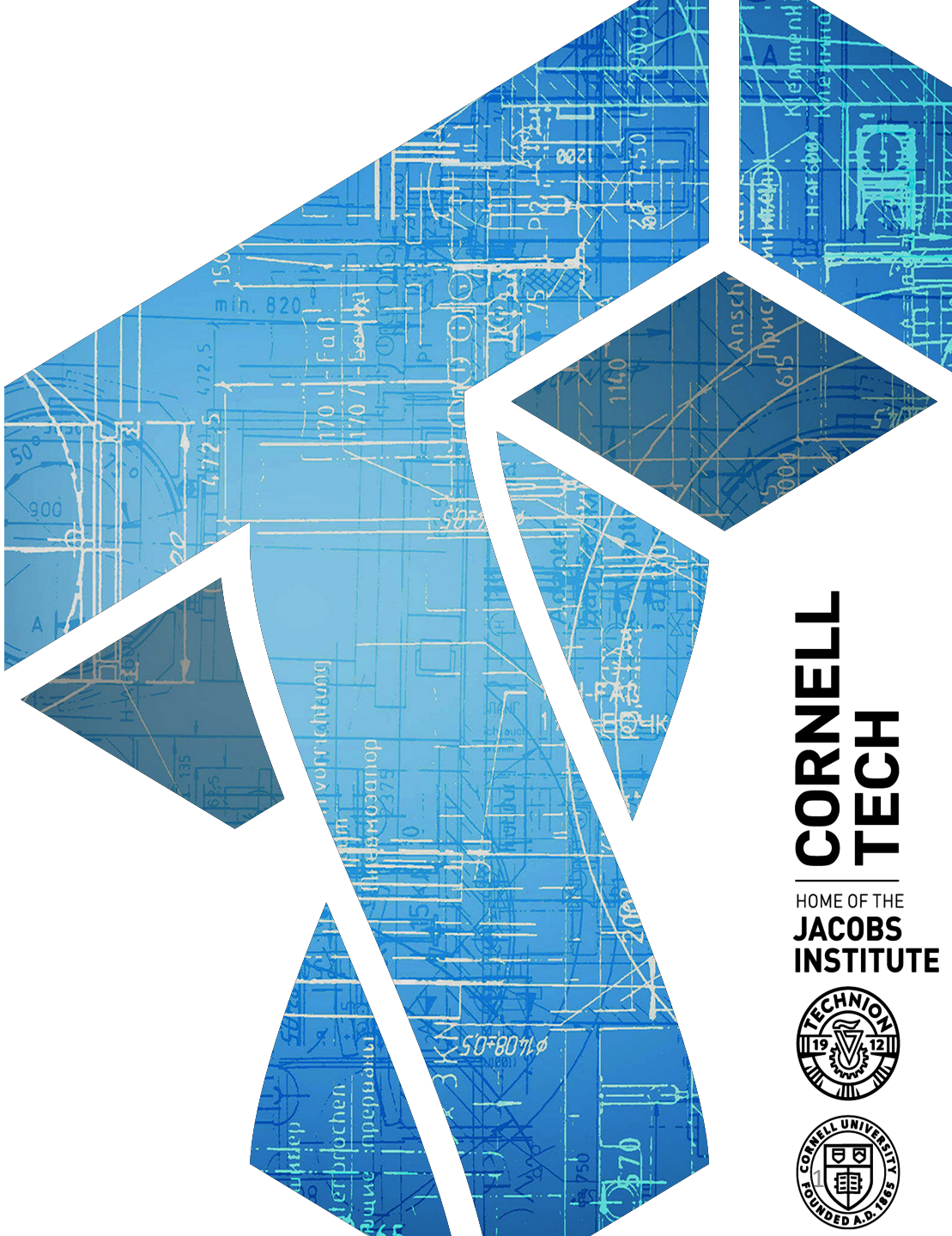
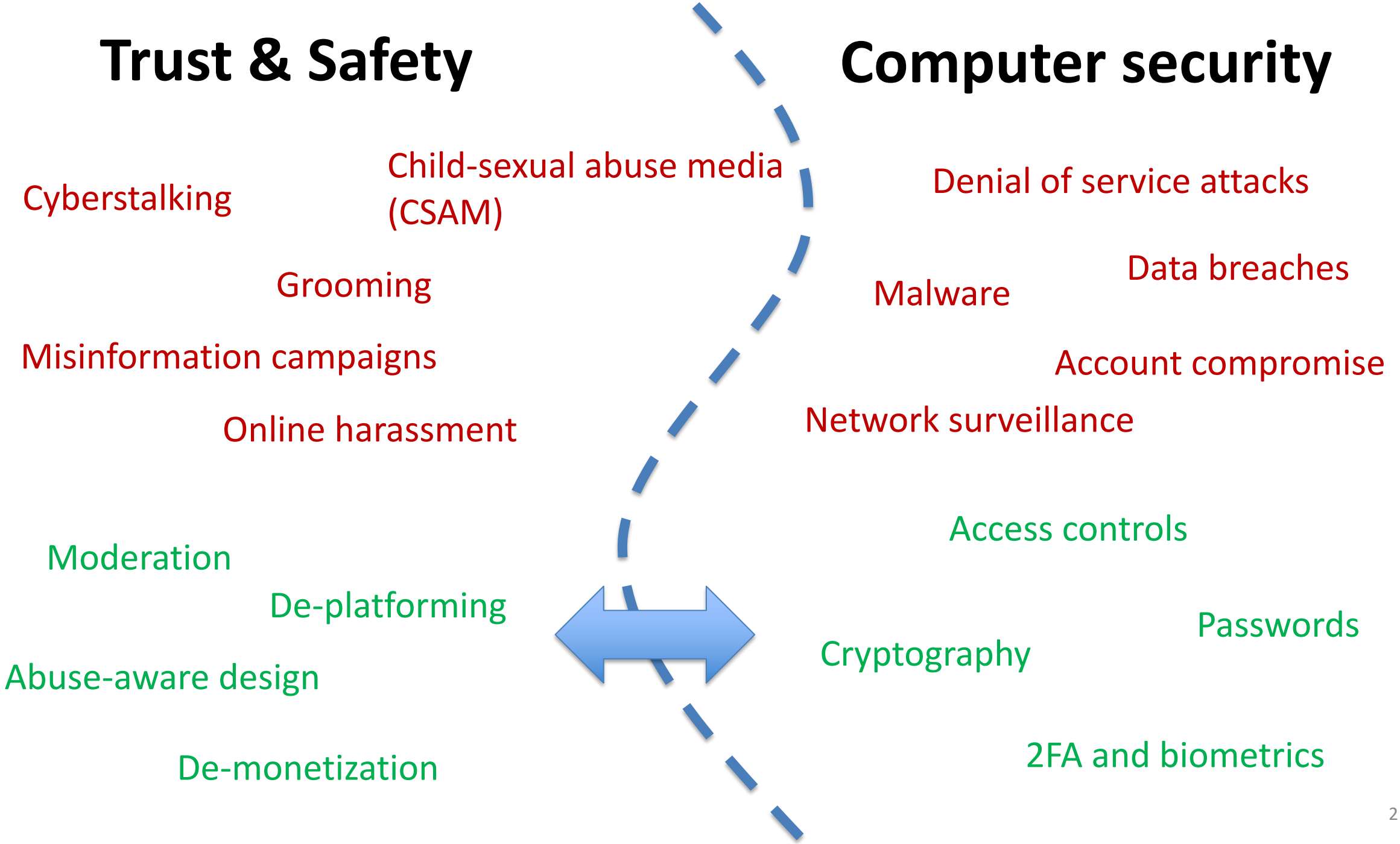
CS 5830
Cryptography

Instructor: Tom Ristenpart
TAs: Yan Ji, Sanketh Menda



CORNELL TECH
HOME OF THE JACOBS INSTITUTE

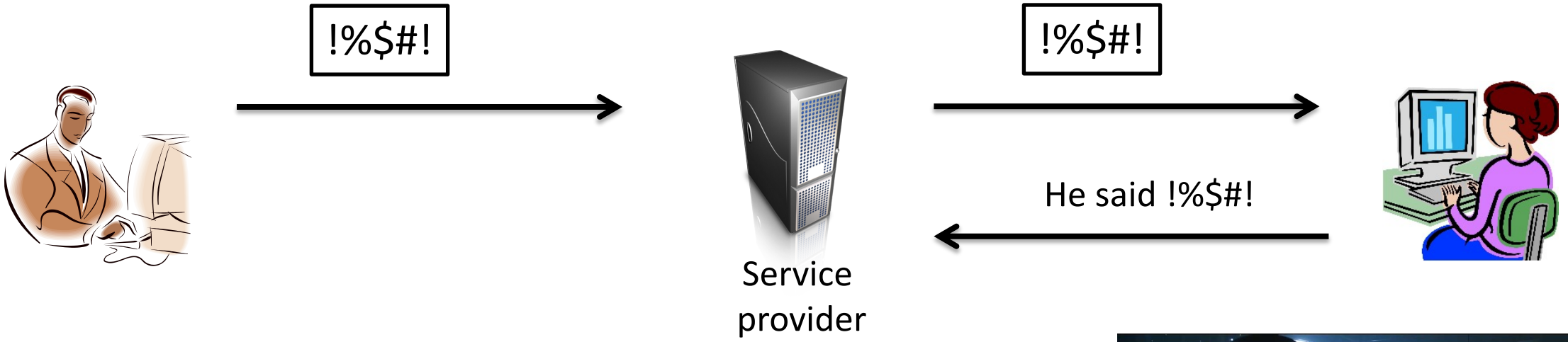

Trust & Safety
Computer security
Cyberstalking
Child-sexual abuse media (CSAM)
Denial of service attacks
Grooming
Malware
Data breaches
Misinformation campaigns
Account compromise
Online harassment
Network surveillance
Moderation
Access controls
De-platforming
Passwords
Cryptography
Abuse-aware design
2FA and biometrics
De-monetization

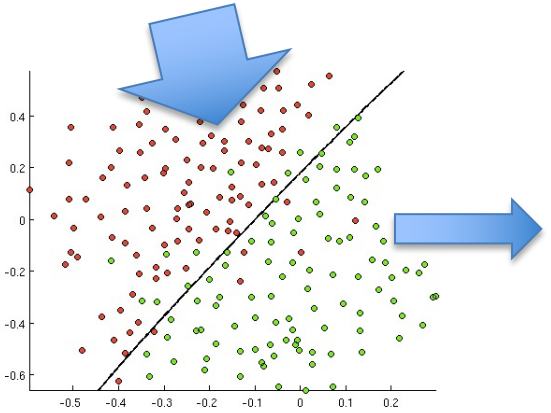
Trust & safety increasingly important

- Qualitatively different types of problems than in traditional computer security & cryptography
- Large companies all have teams focused on these issues

- Key question: How does cryptographic design interplay with trust and safety concerns?
  - Encryption vs. abuse mitigation tensions

End-to-end encrypted messaging and abuse



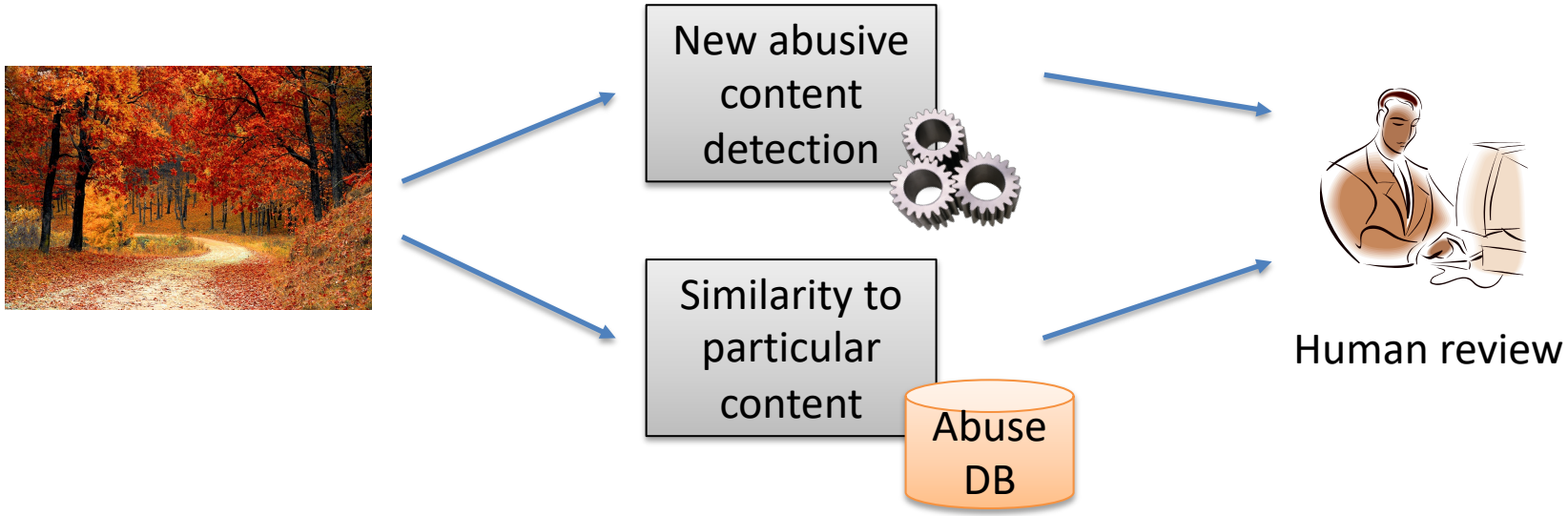


Content moderation:

- Abuse reports from users
- Use machine learning & human review to apply policies

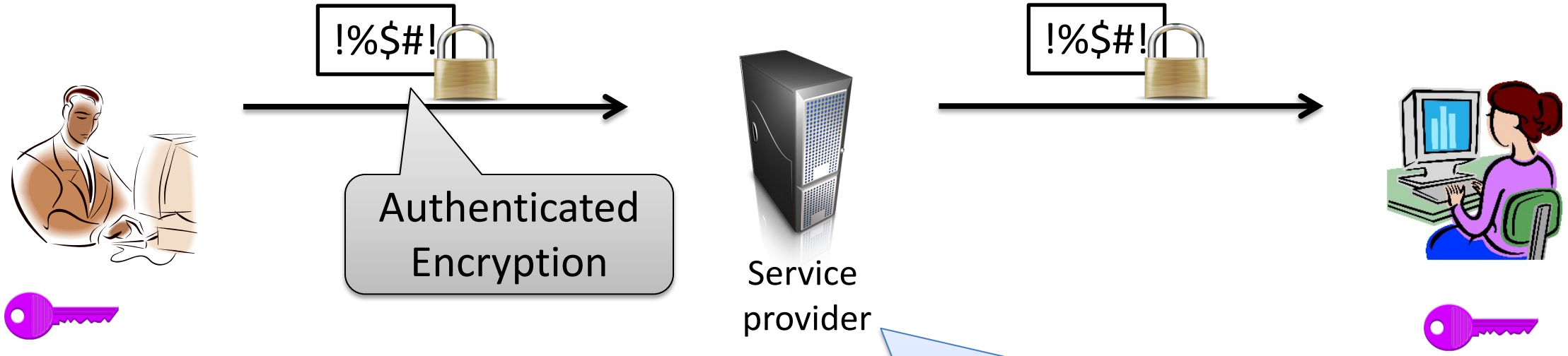
May lead to content or account removal (deplatforming)

Identifying abuse content



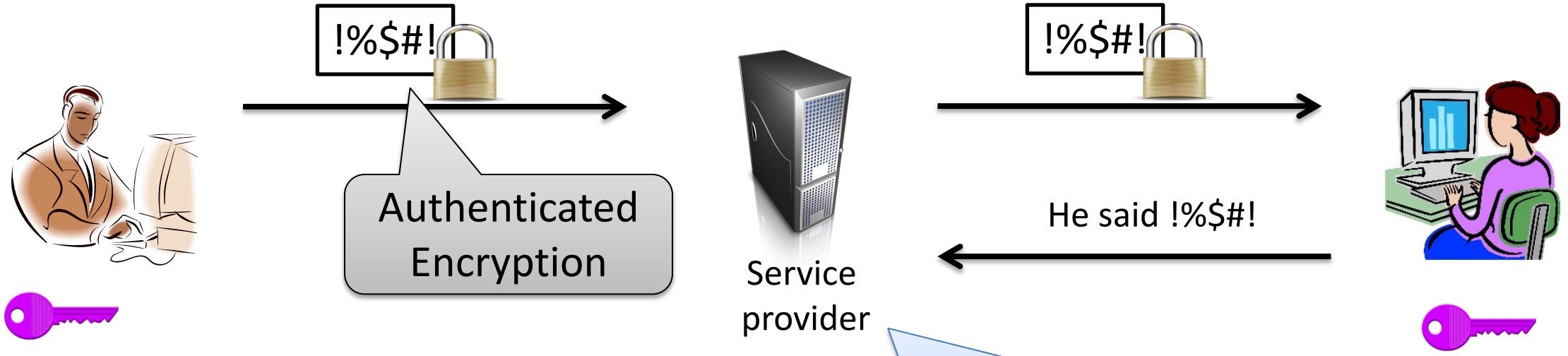
- Use locality-sensitive hashing to check if similar to known bad content
  - H such that |H(image) – H(image')| < threshold if image, image' very similar
- Use machine learning to try to identify patterns indicative of abusive content
  - Is image a picture of a naked child?
- Refer out to human for review (Facebook has 10,000s of moderators)
- Content flagged by users also sent through reviewing pipelines
  - Flagging mechanisms also subject to abuse (illicit takedowns)

End-to-end encrypted messaging and abuse



End-to-end security:
Provider cannot read or modify messages

End-to-end encrypted messaging and abuse

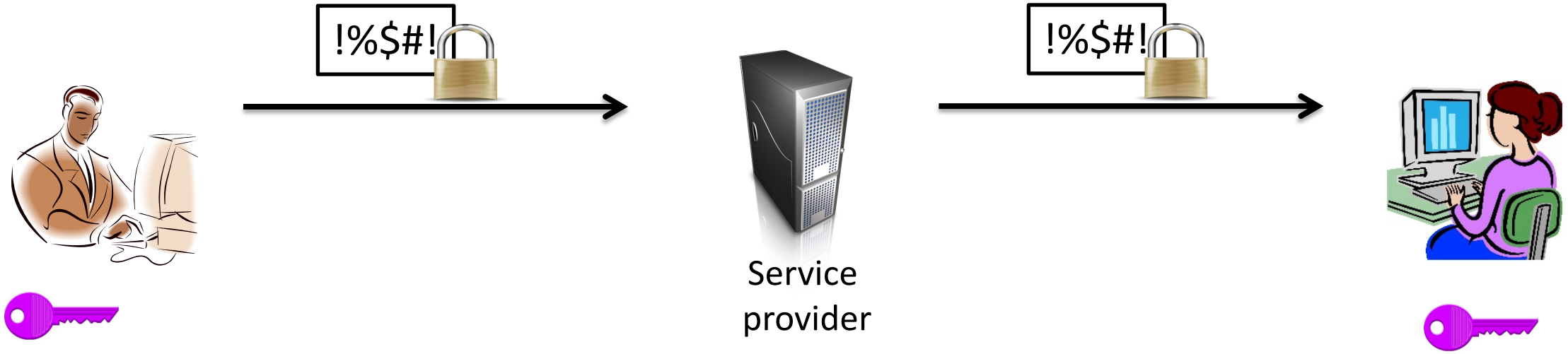


End-to-end security means cannot verify “ !%$#! ” was message sent

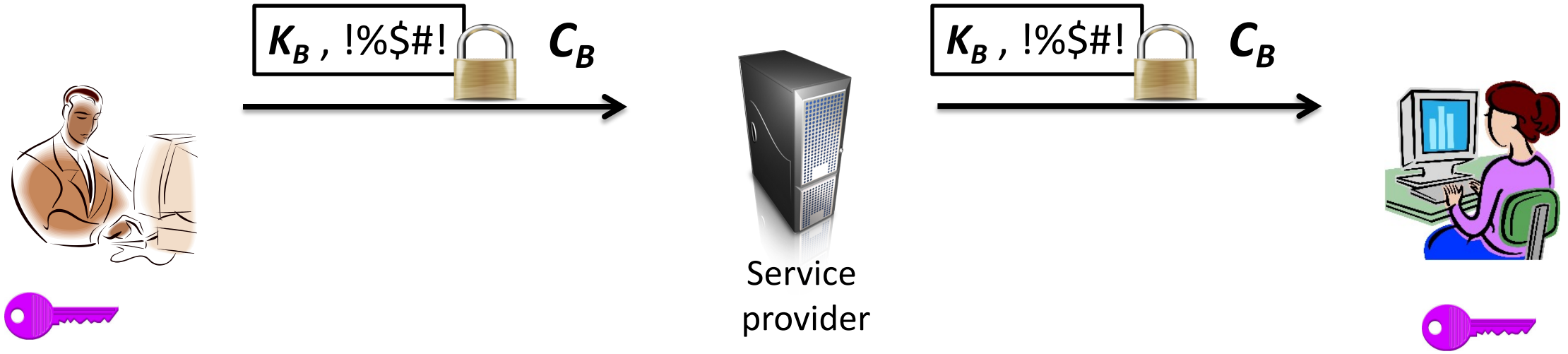
[Facebook 2016]:

- Provide cryptographic proof of message contents when reporting abuse
- Called technique ***message franking***

Facebook's message franking protocol



Facebook's message franking protocol



Sender ***cryptographically commits*** to message: $C_B = HMAC(K_B,$ "!%$#!"$)$

Cryptographic commitments

Classical primitive that allows you to post a “sealed digest” for some message, and to “open” it later

- Commit(M) outputs a pair (C_B, K_B) , the commitment com and the opening K_B
- Open(C_B, K_B,M) outputs 0 (reject) or 1 (accept)

Security properties:

- ***Binding***: can’t open to a different message than what was committed to
- ***Hiding***: C_B leaks nothing about M

Cryptographic commitments

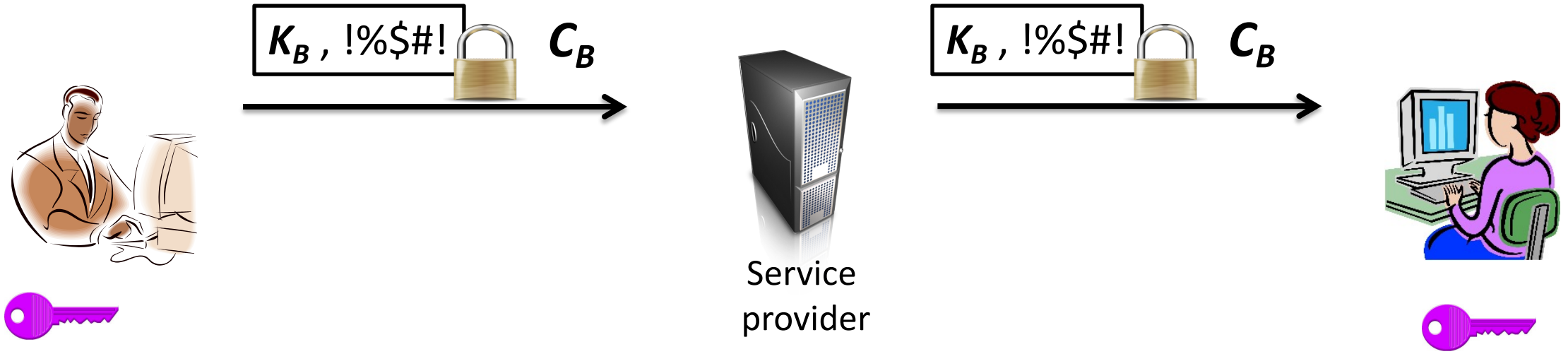
Classic hashing approach:

- Commit(M) chooses key K_B, and outputs (HMAC(K_B,M), K_B)
- Open(C_B,K_B,M) checks that com = HMAC(K_B,M)

Security properties:

- ***Binding***: if HMAC is collision resistant, commitment is binding
- ***Hiding***: C_B leaks nothing about M if HMAC is secure PRF

Facebook's message franking protocol

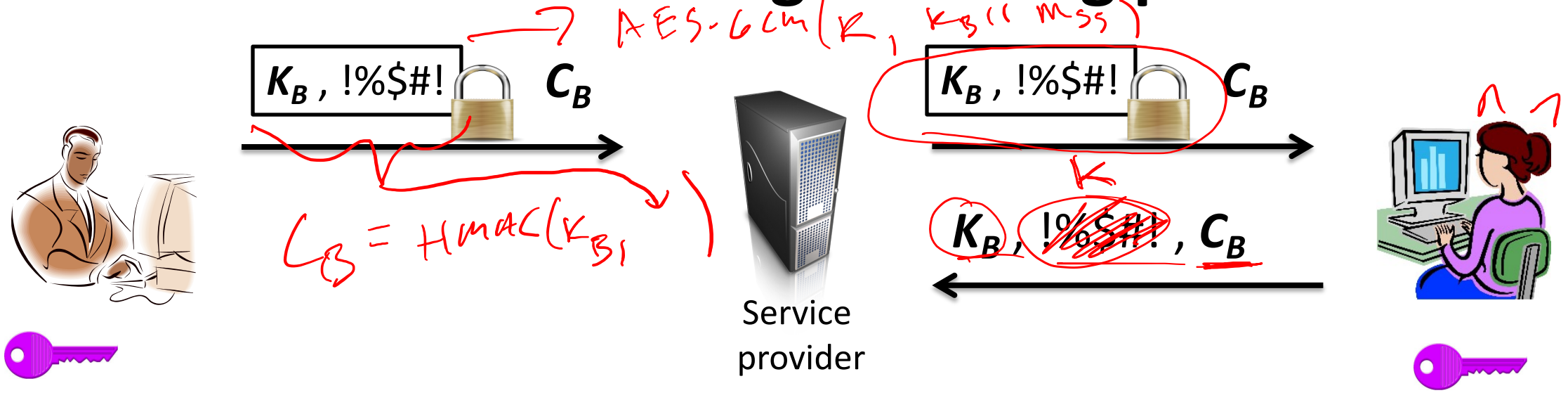


Sender ***cryptographically commits*** to message: $C_B = HMAC(K_B,$ "!%$#!"$)$

Encrypt message along with K_B (called the opening)

Receiver decrypts, retrieves K_B, and verifies C_B

Facebook's message franking protocol

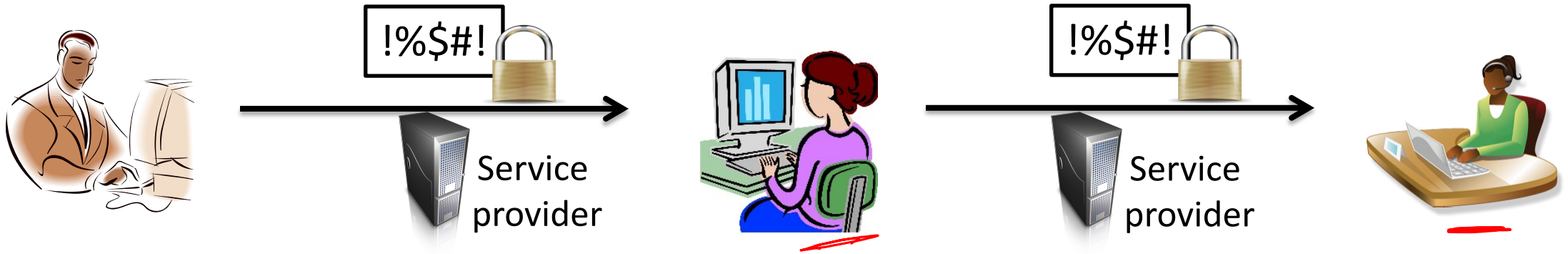


To report abuse, send message as well as K_B , C_B

Provider can verify that C_B was previously sent (and by whom)

Recompute $HMAC(K_B,$ "!%$#!"$)$ and check that it equals C_B

Viral misinformation and cryptographic traceback

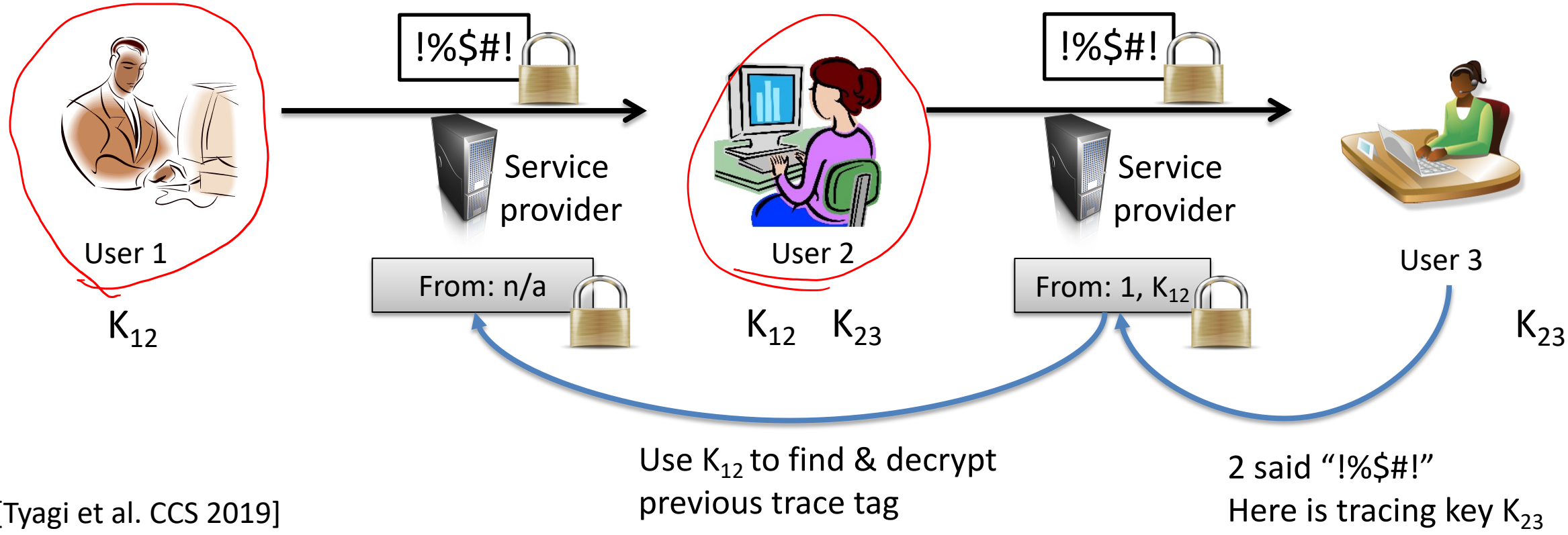


APAC OCTOBER 27, 2020 / 6:06 AM / UPDATED 19 DAYS AGO

Fake news spread on WhatsApp to Indian Americans plays stealth role in U.S. election

By Paresh Dave 5 MIN READ

Viral misinformation and cryptographic traceback



[Tyagi et al. CCS 2019]

Traceback for encrypted messaging:

- Cryptographic abuse report that allows inferring path of message through user network
- Nothing revealed about forwarding behavior & message until report made
- Fast schemes that use only symmetric cryptography
- Lots of interesting open questions, some solved by subsequent work

Can we help identify abuse content?

- Abuse reporting mechanisms rely on recipient to report message
- How do we help users identify abuse content in encrypted messaging?

Client-side scanning

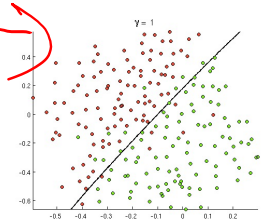
!%$#!

Service provider

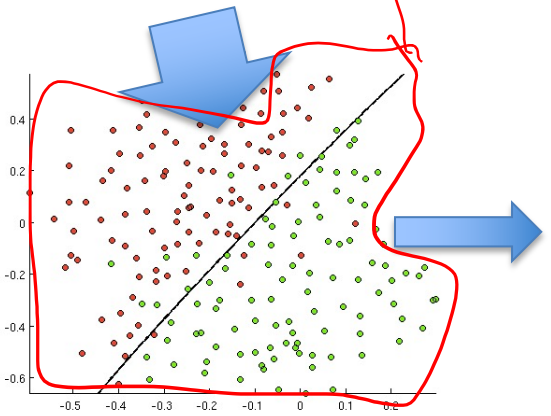
!%$#!

He said !%$#!

!%$#!



Warning!



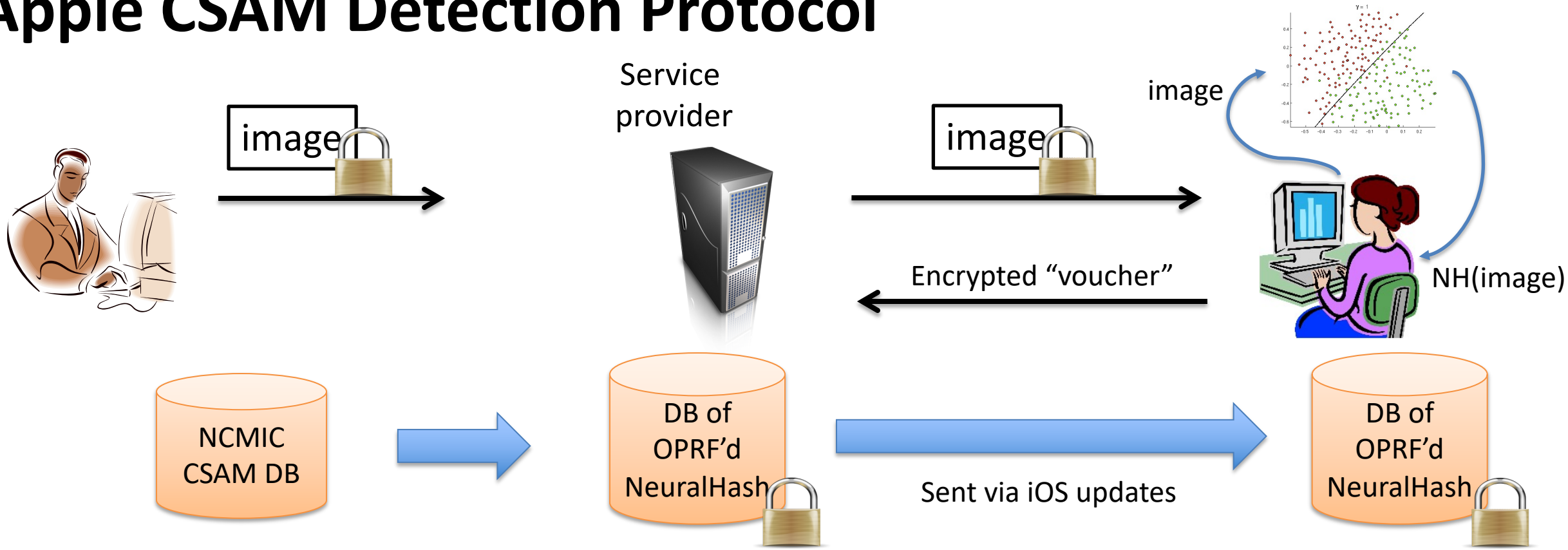
Two options:

1. Client automatically reports flagged content to provider
2. Client warns user, let's them to decide whether to report

Apple CSAM detection proposal

- Use locality-sensitive hashing combined with threshold private-set intersection (PSI) protocol
- Encrypted database of known CSAM hashes on devices
  - Partnership with NCMEC (National Center for Missing & Exploited Children) to gather examples of bad images
- Client-side computation of hashes of user images
- Protocol to notify Apple when sufficiently many images match

Apple CSAM Detection Protocol



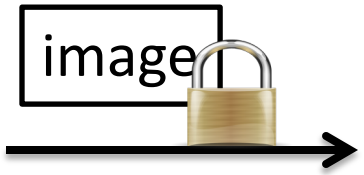
NeuralHash:

Deep learning based mechanism to build hash function NH such that:

NH(image) = NH(image’)

if image and image’ sufficiently similar. Harder to build than similarity hashes

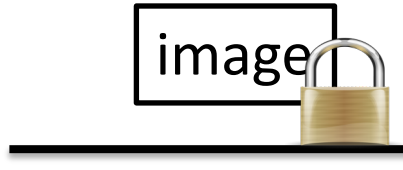
Straw proposal: simple OPRF-style protocol



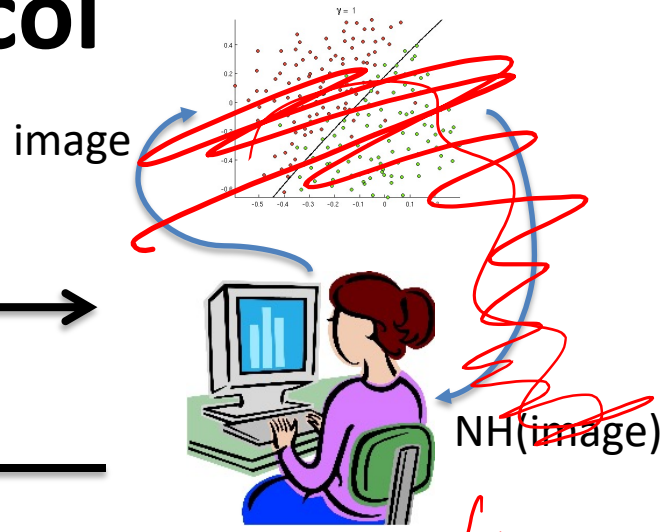
Service provider

Z' <- Q^K

If H(Z') = chk then match



Q, chk



Given NCMEC DB = {$image_1$,…,$image_m$}

PRF key K <-$ $\mathbf{Z}_p$ using group G w/ |G| = p

$pk = g^K$

Build hash table of size n'

$T[NH(image_j)] = NH(image_j)$

For i = 1 to n':

If T[i] empty then P[i] <-$ G

Else $P[i] <- H(T[i])^K$

Let X = P[NH(image)]

y <-$ $\mathbf{Z}_p$

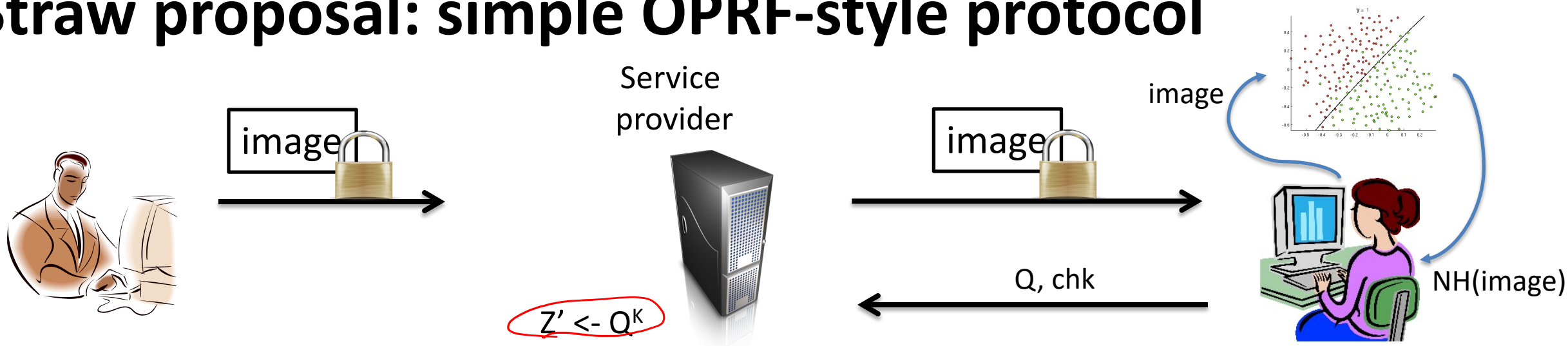
$Q <- H(NH(image))^y \cdot g^y$

$Z <- X^y \cdot (pk)^y$

chk <- H(Z)

h
$H(NH(image_1))^K$
…
h'
$H(NH(image_5))^K$
…

Straw proposal: simple OPRF-style protocol



Match case:

$X = H(NH(image))^K$

$Z = X^y * (pk)^y = H(NH(image))^{Ky}\, g^{Ky}$

No match case:

X = random group element

$Z = X^y * (pk)^y = X^y\, g^{Ky}$

Let X = P[NH(image)]

$y \leftarrow\$ \mathbf{Z}_p$

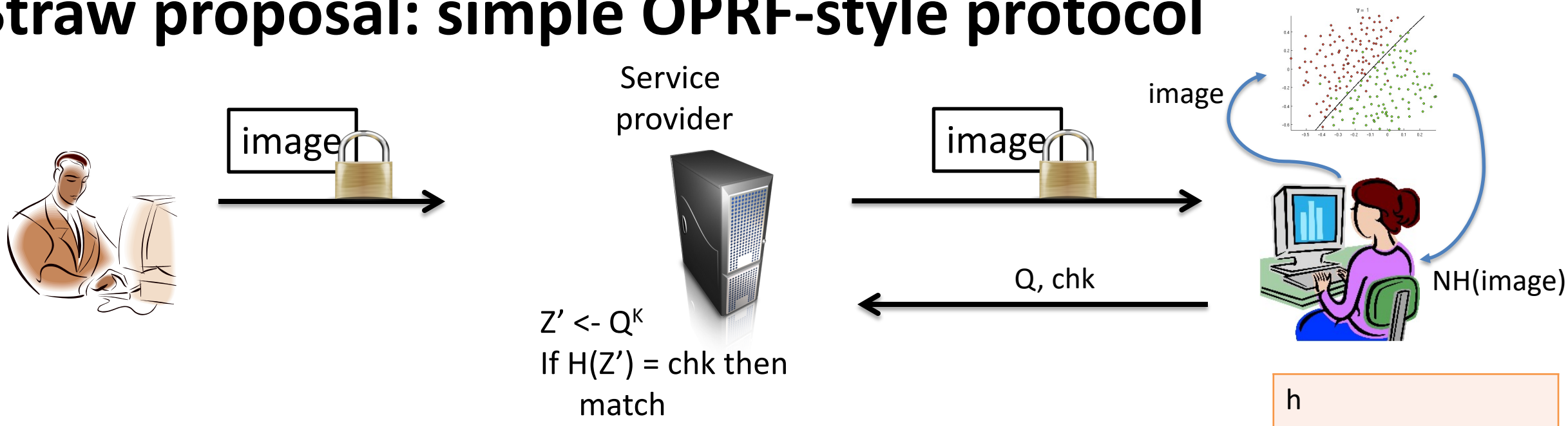
$Q \leftarrow H(NH(image))^y \cdot g^y$

$Z \leftarrow X^y \cdot (pk)^y$

chk <- H(Z)

h
$H(NH(image_1))^K$
…
h'
$H(NH(image_5))^K$
…

Straw proposal: simple OPRF-style protocol



h
$H(NH(image_1))^K$
…
h'
$H(NH(image_5))^K$
…

This is far simpler protocol, only provides private set intersection

Apple protocol uses:

- Cuckoo hash tables to build P,
- Thresholding to reveal number of matches only after certain number
- Synthetic vouchers

Let X = P[NH(image)]
y <-$ $\mathbf{Z}_p$
Q <- $H(NH(image))^y \cdot g^y$
Z <- $X^y \cdot (pk)^y$
chk <- H(Z)

Client-side scanning debated topic

- Negative reaction to proposed Apple protocol
- “Bugs in our Pockets: The Risks of Client-Side Scanning”
  - https://arxiv.org/abs/2110.07450
  - This is explicitly about option (1), where scanning triggering automated content reporting, as in Apple case
- Also proposed in same press release nudity detection feature for children (grooming countermeasure)
  - Warning to parent’s Apple account

Very negative public reaction

- Use locality-sensitive hashing combined with threshold private-set intersection (PSI) protocol
- Encrypted database of known CSAM hashes on devices
- Client-side computation of hashes of user images
- Protocol to notify Apple when sufficiently many images match

Take-aways

- Trust and safety important topic
- Deeply intertwined with cryptography
  - Encryption raises challenges
  - Cryptographic protocols provide potential for privacy-preserving solutions