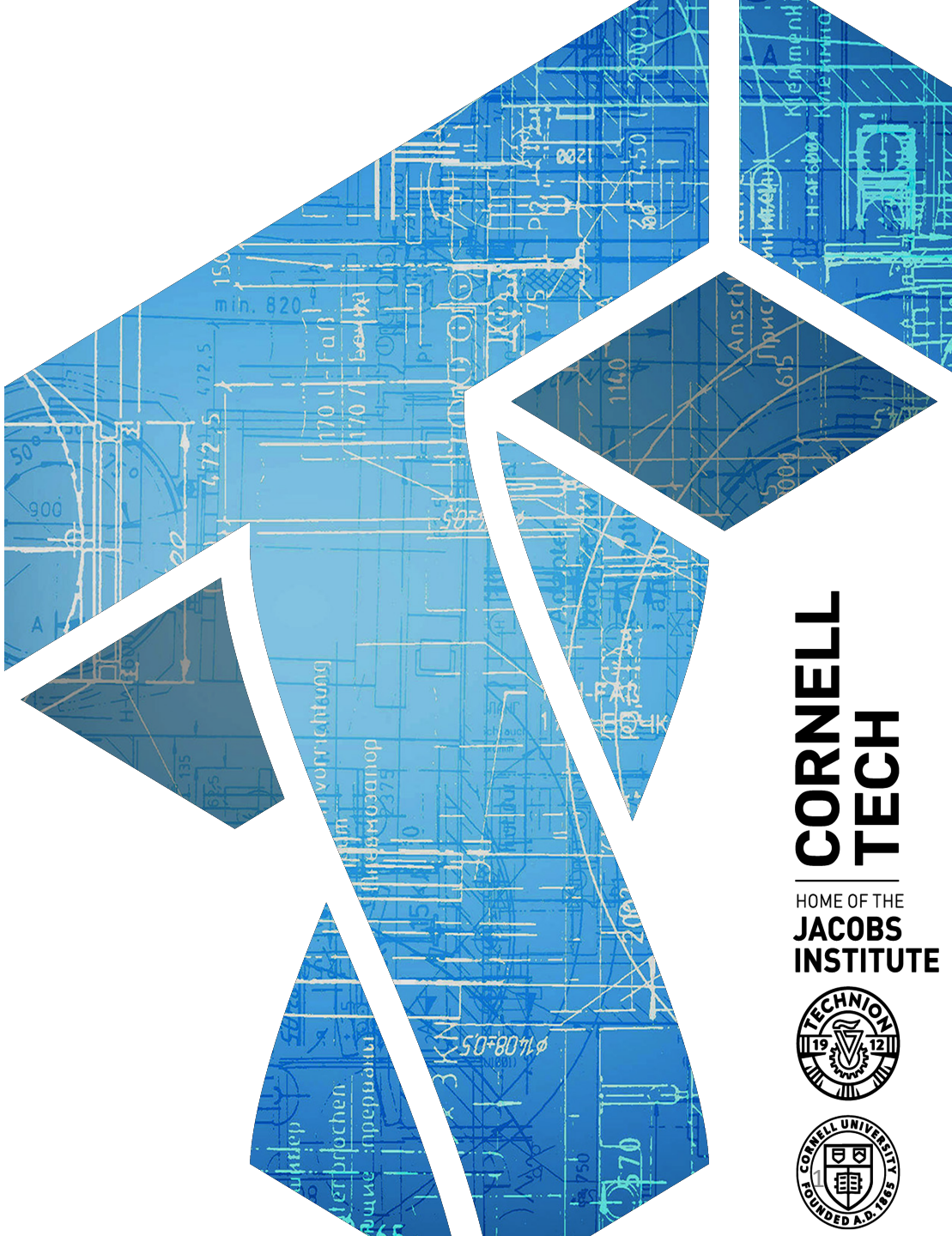
CS 5830
Cryptography

Instructor: Tom Ristenpart
TAs: Yan Ji, Sanketh Menda



CORNELL TECH
HOME OF THE JACOBS INSTITUTE

Emerging topics in applied crypto

- We'll spend a couple lectures covering some cryptographic tools that have started seeing use in practice recently
  - Secure computation
  - Oblivious PRFs
  - Zero-knowledge proofs

- Theory goes back decades, but real-world applications were historically elusive

Context: credential stuffing attacks

tom, password1

Login server

tom	$salt_1$, $H(salt_1$, password1)
alice	$salt_2$, $H(salt_2$, 123456)
bob	$salt_3$, $H(salt_3$, p@ssword!)

✓

tom	password1
sally	hell0w0rlD4
nick	cloudflareRul3z
…	…

Breach alerting service

alert: tom has compromised credential!

';--have i been pwned?

Password Checkup

Offered by: google.com

Protocols for checking if password is in breach with third-party service

[Li et al. 2019] [Thomas et al. 2019]

Google & Li et al. credential checking protocol

Login server

Breach alerting service

tom	password1
sally	hell0w0rlD4
nick	cloudflareRul3z
…	…

tom, password1

Private set membership protocol:

- Client learns if (tom, password1) in breach database
- Server learns nothing about password1

Two-party private set intersection (PSI)

- Party 1 has a set A
- Party 2 has a set B
- One or both parties want to learn $A \cap B$
- Parties should not learn any other values from other party's set

A hash-based solution

Client

Server

$h_1, \ldots, h_{|A|}$

$g_1, \ldots, g_{|B|}$

For $x_i \in A$:
$h_i \leftarrow H(x_i)$

For $1 \le i \le |A|$:
if $h_i = g_j$ for some j then
$C \leftarrow C \cup \{x_i\}$

Return C

For $y_i \in B$:
$g_i \leftarrow H(y_i)$

For $1 \le i \le |B|$:
if $g_i = h_j$ for some j then
$C \leftarrow C \cup \{y_i\}$

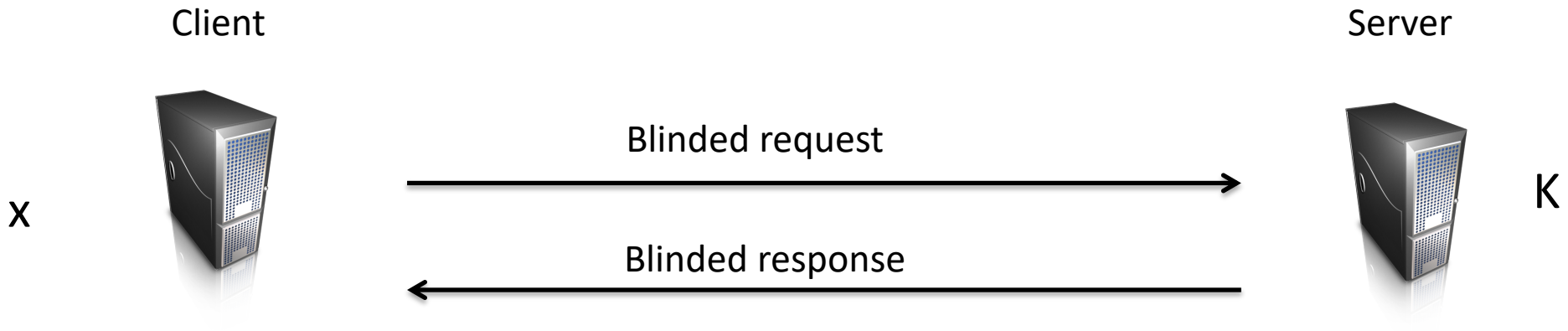
Return C

Security problems?

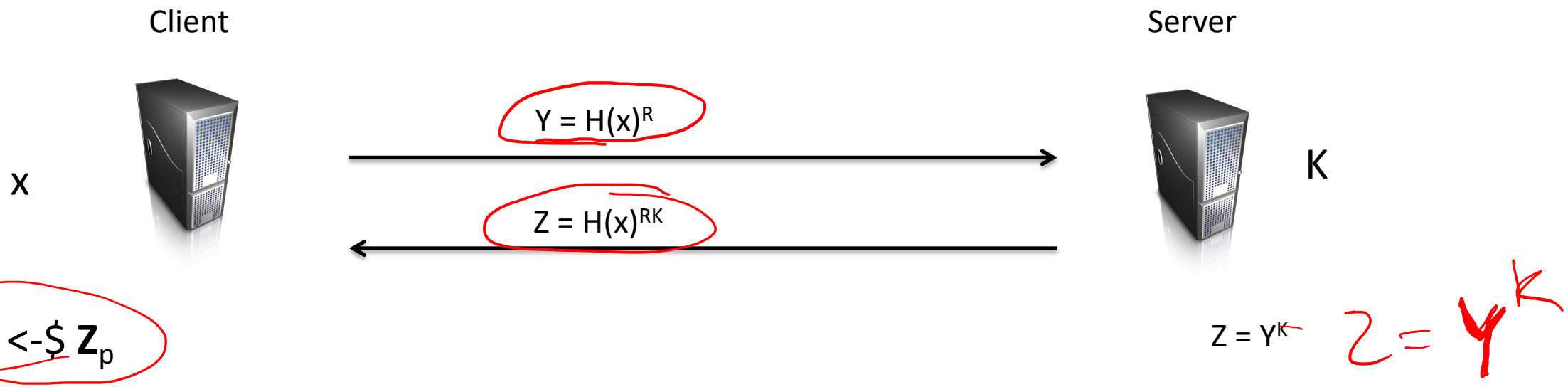
A tool: oblivious PRFs

- PRF is a keyed function $F_K(x)$ which is indistinguishable from random function (when K is secret)
- Oblivious PRFs come with blinded evaluation protocol:
  - Client holds message x
  - Server holds secret key K
  - Client can obtain $F_K(x)$ without learning K, and without revealing x

A tool: oblivious PRFs



A tool: oblivious PRFs



Fix prime-order group G with |G| = p
Let H : {0,1}* -> G be a hash function

$F_K(x) = H(\ H(x)^K\ )$

2HashDH protocol by Jarecki et al.
Now on standards track

One-sided OPRF-based solution to PSI

Client

Server

$Y_1, \dots, Y_{|A|}$ →

← $Z_1, \dots, Z_{|A|}, g_1, \dots, g_{|B|}$

$R \leftarrow\$ \mathbf{Z}_p$
For $x_i \in A$:
  $Y_i \leftarrow H(x_i)^R$

For $1 \le i \le |A|$:
  if $Z_i^{1/R} = g_j$ for some j then
    $C \leftarrow C \cup \{x_i\}$

Return C

For $y_i \in B$:
  $g_i \leftarrow F_K(y_i)$

For $1 \le i \le |A|$:
  $Z_i \leftarrow Y_i^K$

Achieves our goals:
- Client learns intersection, nothing more
- Server learns nothing about A (beyond |A|)

Downsides?

If |A| or |B| is large, prohibitive communication cost

Bucketized PSI

16-bit hash of username

Client

$\text{Trunc}_{16}(username)$, $H(username,pw)^R$

$H(username,pw)^{R*K}$, bucket

C3 service

Set B. $|B| > 10^9$

tom	password1
sally	hell0w0rlD4
nick	cloudflareRul3z
…	…

Holds secret key K for ***oblivious PRF***

Pick R at random
("blind" the hash)

Compute $g = H(H(username,pw)^K)$
by removing blinding

If $g \in$ bucket then
(*username*,*pw*) is in a breach

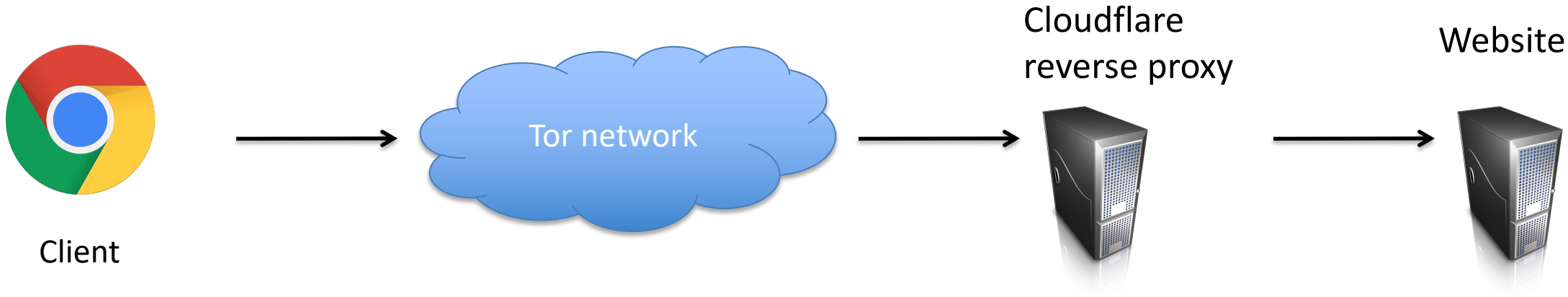
$\text{bucket} = \{ H(username',pw')^K \mid \text{Trunc}_{16}(username') = \text{Trunc}_{16}(username) \}$

- Bucketization leaks some info but only about username
  - Reduces |bucket| (bandwidth cost)
- OPRF requires just one client-side exponentiation (ECC multiplication)
  - Google uses slow hash for abuse resistance

Other applications of PSI

- Signal private contact discovery (used SGX instead)
  - https://signal.org/blog/private-contact-discovery/
- Meta's work on recovering insight into user behavior
  - https://engineering.fb.com/2020/07/10/open-source/private-matching/
  - Microsoft, Google have similar efforts
- Others?
  - Old example: casino badlists

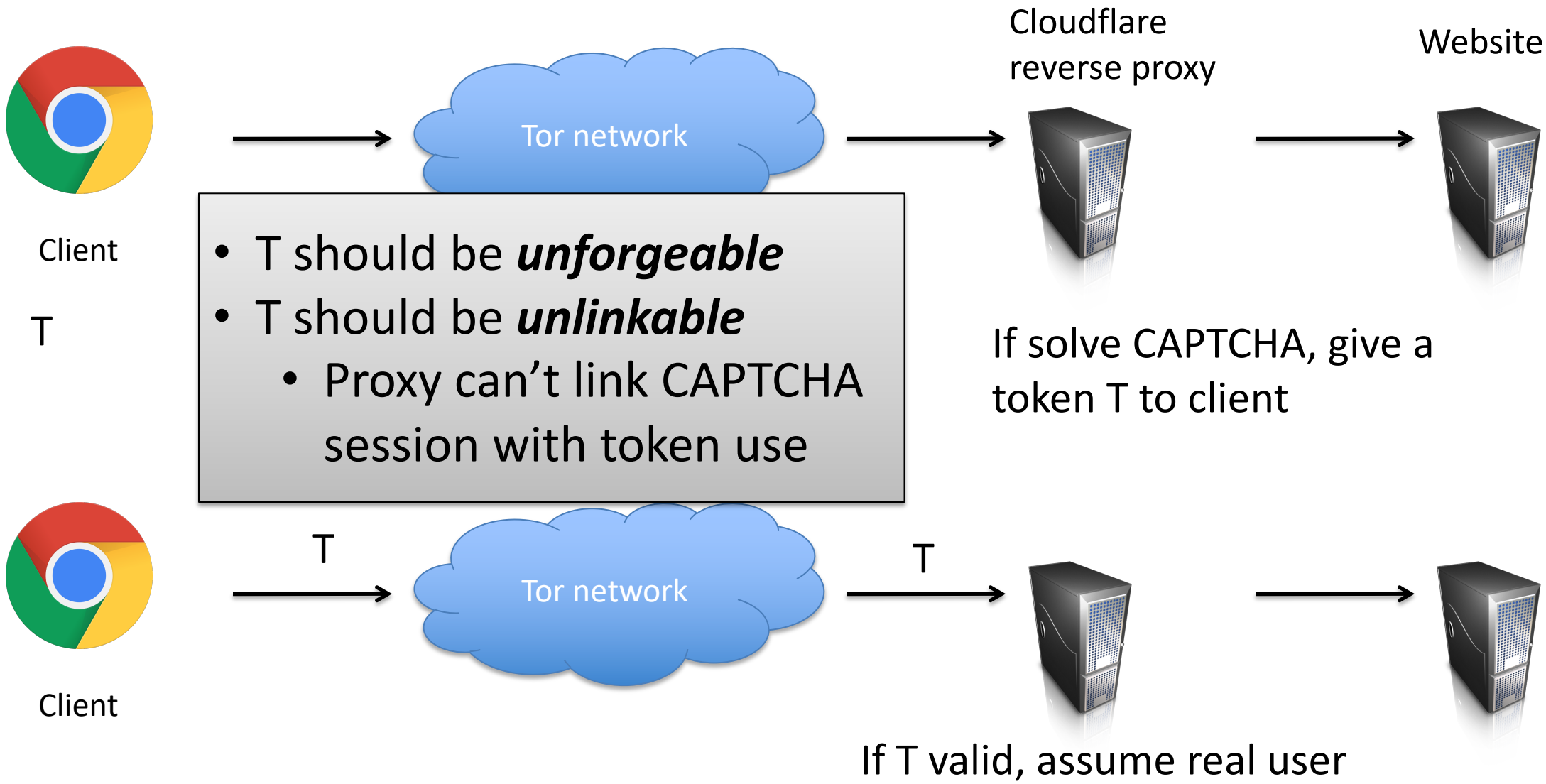
OPRFs used for anonymous tokens



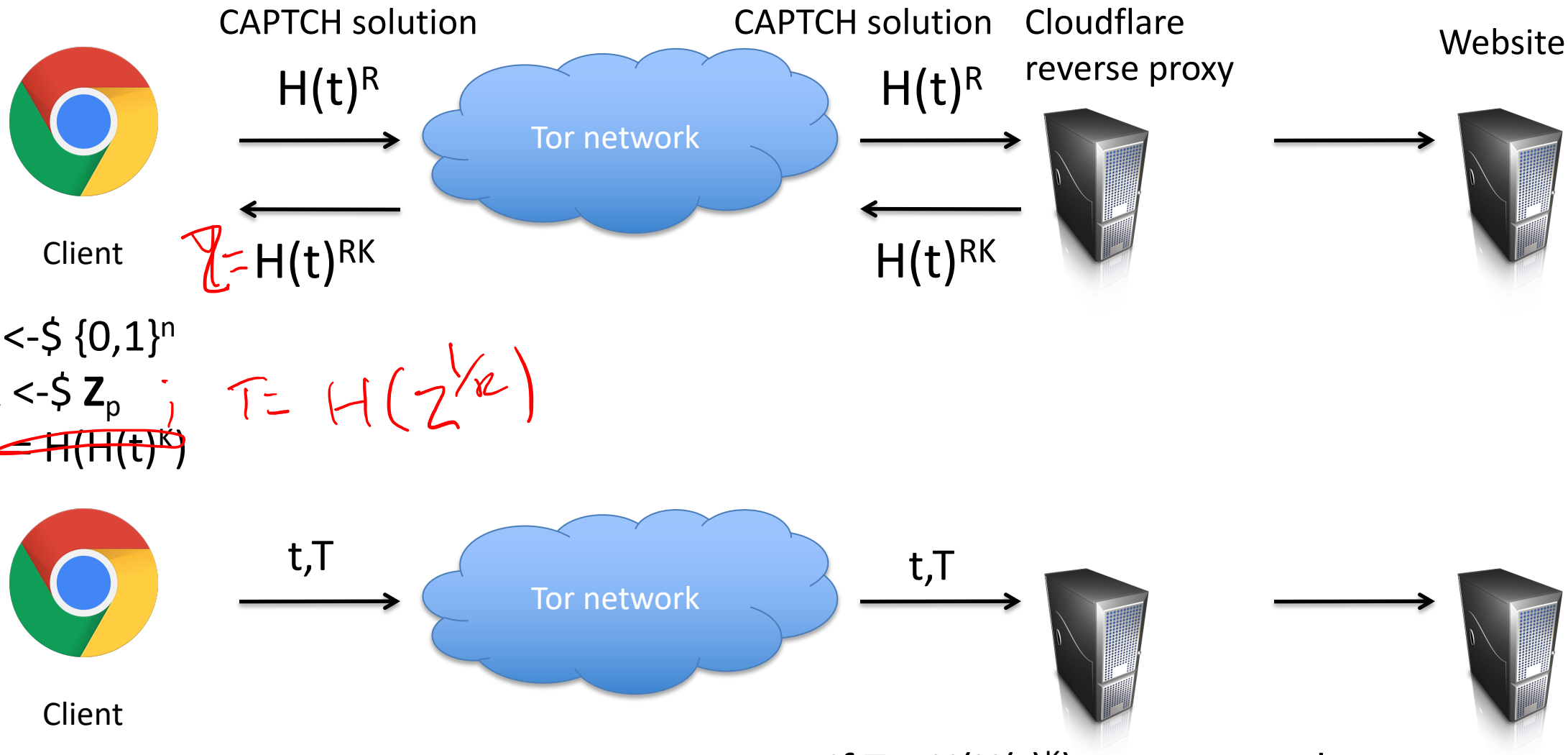
Problem:
Using Tor makes “client IP” seen by proxy change frequently

Anti-fraud protections include forcing users in some cases to solve CAPTCHA.
Only solve once per time period for each client IP

OPRFs used for anonymous tokens



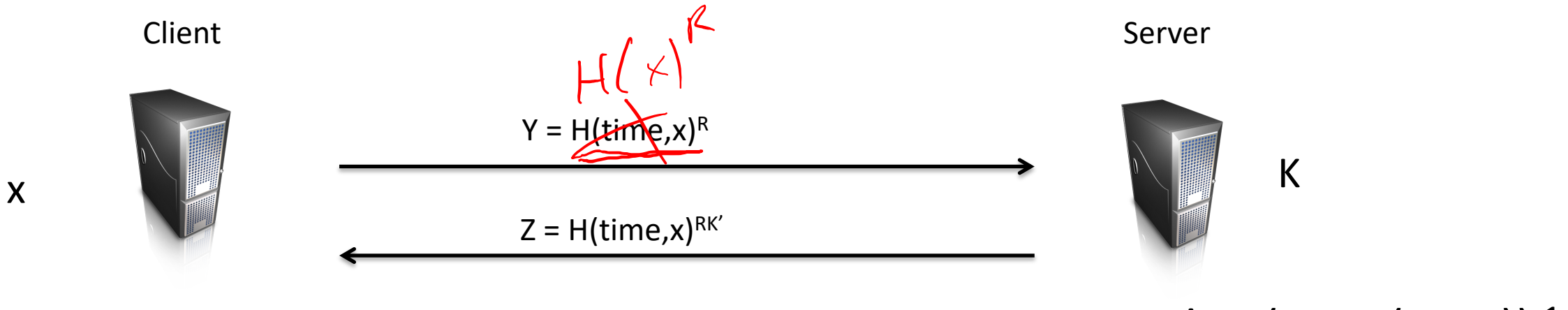
OPRFs used for anonymous tokens



Privacy Pass

- OPRF-based solution written up in Privacy Pass paper
- Subsequently standardized and deployed
- One concern is ***hoarding attacks***
  - Attacker harvests a bunch of tokens
  - Attacker spends all tokens at once to perform DDoS attack
  - Unlinkability prevents detection
  - Solution: partially oblivious PRFs
    - F_K(time,x) where time is public input, x is private input
    - Problem: no fast PO-PRFs were known

3HashSDHI partially oblivious PRF



Fix prime-order group G with |G| = p
Let H : {0,1}* -> G be a hash function

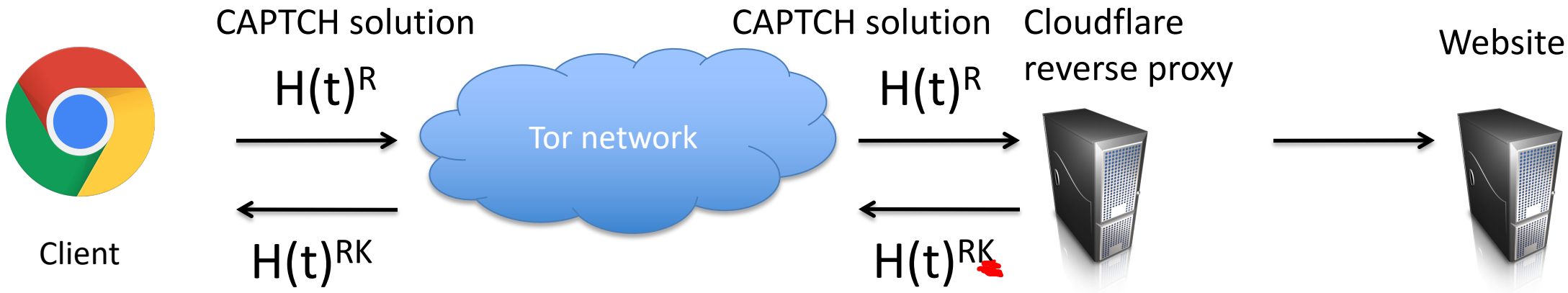
$$F_K(time,x) = H(\ H(time, x)^{1 / (K + H(time))}\ )$$

Now on standards track.
Also useful in breach alerting: bind PRF output to buckets

Privacy Pass

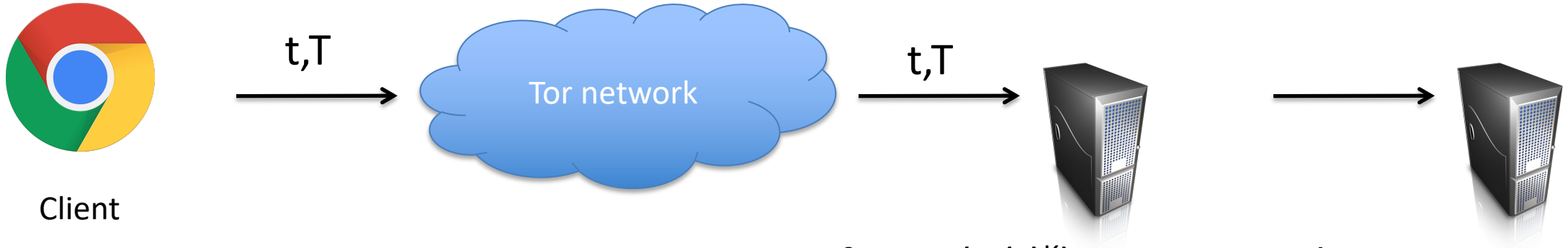
- OPRF-based solution written up in Privacy Pass paper
- Subsequently standardized and deployed
- One concern is ***hoarding attacks***
  - Attacker harvests a bunch of tokens
  - Attacker spends all tokens at once to perform DDoS attack
  - Unlinkability prevents detection
  - Solution: partially oblivious PRFs
    - F_K(time,x) where time is public input, x is private input
    - Problem: no fast PO-PRFs were known
- Another concern is ***tagging attacks by proxy***

Privacy Pass



$t \leftarrow\$ \{0,1\}^n$
$R \leftarrow\$ \mathbf{Z}_p$
$T = H(H(t)^K)$

What if server uses new K for each client?
Can trivially link users!



If $T = H(H(t)^K)$, assume real user

Resisting active tagging attacks

- Privacy Pass: proxy needs to prove that they sign each request with the same secret key K
  - Can publish a public key $pk = g^K$
  - Prove that Y^K and pk have same discrete log, without revealing K
- Zero-knowledge proofs (ZKPs) can handle this

Zero-knowledge proofs

- Prove knowledge of secret x without revealing any information about x
- Introduced by Goldwasser, Micali, Rackoff
  - "The Knowledge Complexity of Interactive Proof-Systems", 1989

(Variant of) Schnorr signatures

Let G be group of prime order q. Let g be a generator

sk = x chosen randomly from $\mathbf{Z}_q$ pk = X = g^x

Sign(x, M)

r <-$ $\mathbf{Z}_q$

R = g^r ; c = H(M || R) ; z = r + cx mod q

Return (R,z)

Ver(X, M, (R,z))

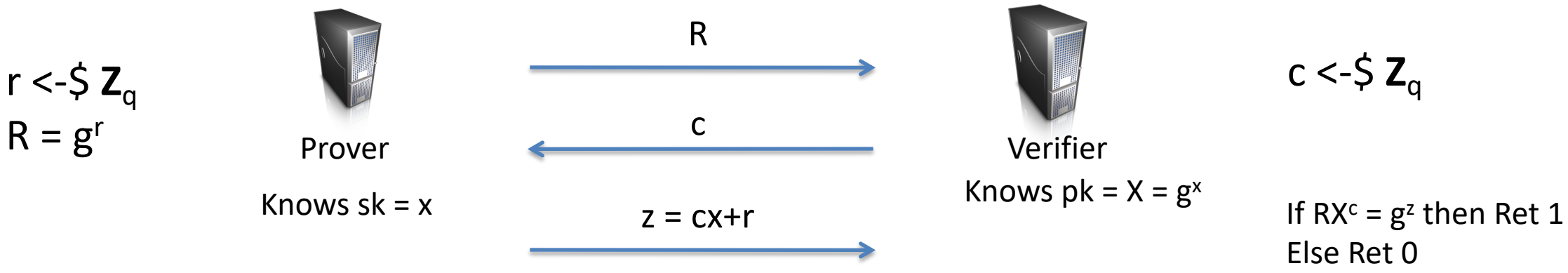
c = H(M || R)

If $g^z = RX^c$ then Return 1

Return 0

Correctness? $g^z = g^{r + cx} = g^r g^{xc} = RX^c$

Schnorr Sigma protocol



Prover wants to prove that they have sk = x associated to public key $X = g^x$

Sigma protocols are class of 3-round protocols:

- Prover sends ***commitment***
- Verifier replies with ***challenge***
- Prover sends ***response***

Schnorr just one example, many others:

- Okamoto's, Chaum-Pederson, Guillou-Quisquater (RSA), …

Zero-knowledge properties

- ***Completeness***: can prove a true statement
- ***Soundness***: can't prove a false statement
- ***Zero-knowledge***: proof reveals nothing (except validity)

Next time: ZKPs

- We’ll pick up here and look at ZKPs