# How to (not) Share a Password:

## Privacy preserving protocols for finding **heavy hitters** with **adversarial** behavior


Moni Naor

Benny Pinkas

**Eyal Ronen**

מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

Tradition of Excellence
Bar-Ilan University

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

# Passwords

- First “modern” use in MIT's CTSS (1961)

# Passwords

- First “modern” use in MIT's CTSS (1961)
- “Passwords are dead”?

# Passwords

- First “modern” use in MIT's CTSS (1961)
- “Passwords are dead”?
- User tend to choose passwords with **low** min–entropy
  - Easy to guess

# Compromise a User, Attack the Eco System

- Bad passwords do not only compromise the users

# Compromise a User, Attack the Eco System

- Bad passwords do not only compromise the users
- Weak and popular passwords can be used for large scale attacks

# Compromise a User, Attack the Eco System

- Bad passwords do not only compromise the users
- Weak and popular passwords can be used for large scale attacks
  - E.g. the Mirai attack
  - Easy to find IoT devices with Shodan like search engines
  - Unique weak passwords might be ok, popular passwords are bad

# Compromise a User, Attack the Eco System

- Bad passwords do not only compromise the users

- Weak and popular passwords can be used for large scale attacks
  - E.g. the Mirai attack
  - Easy to find IoT devices with Shodan like search engines
  - Unique weak passwords might be ok, popular passwords are bad

- Service provider liability?

# California Guideline for IoT

- “Security of Connected Devices” signed in California Law October 2018

As of 2020 manufacturers required to either include :

- "a preprogrammed password **unique** to each device manufactured" or
- "a security feature that requires a user to **generate a new** means of authentication before access is granted to the device for the first time.“

# California Guideline for IoT

- “Security of Connected Devices” signed in California Law October 2018

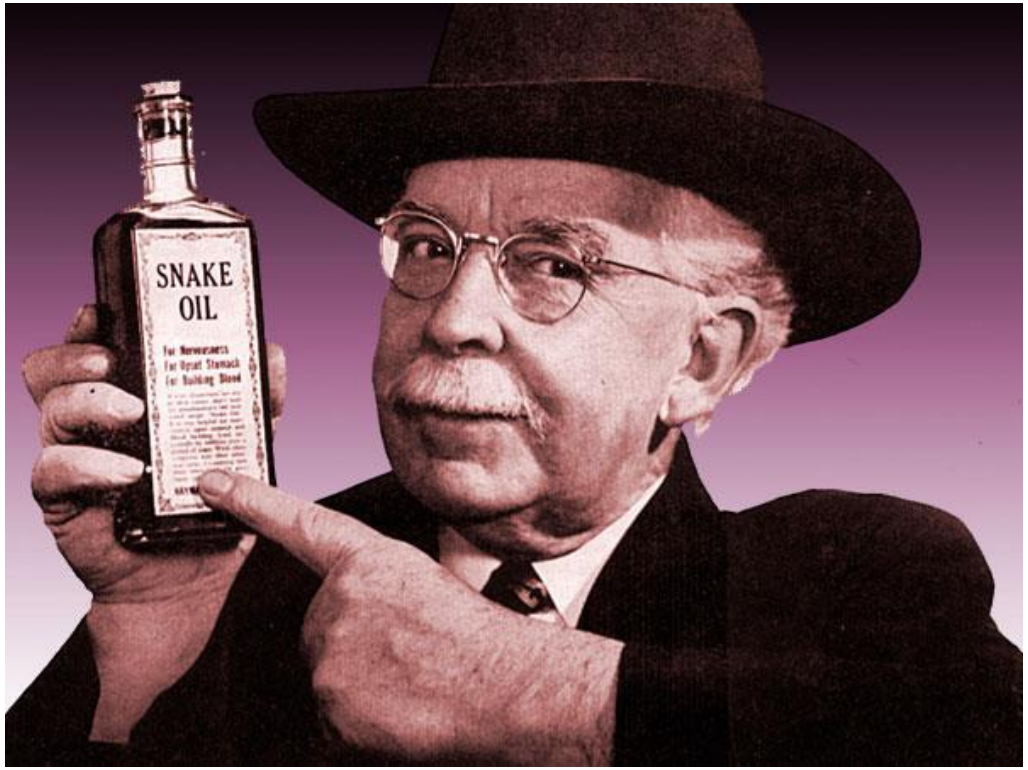
As of 2020 manufacturers required to either include :

- "a preprogrammed password **unique** to each device manufactured" or
- "a security feature that requires a user to **generate a new** means of authentication before access is granted to the device for the first time.“

Force users to change passwords, but to what?

# Possible Solutions

- It is hard to even decide the ideal guidelines for passwords

# Possible Solutions

# Possible Solutions

- Two factor authentication (2FA)

# Possible Solutions

- Two factor authentication (2FA)

BLOG ▸ SECURITY

## The bleak picture of two-factor authentication adoption in the wild

Authors Elie Bursztein  Date December 2018  Reading time 10 min

This post looks at two-factor authentication adoption in the wild, highlights the disparity of support between the various categories of websites, and illuminates how fragmented the two factor ecosystem is in terms of standard adoption.

# Possible Solutions

- Two factor authentication (2FA)
- Server saves a list of **all** users’ passwords and blacklists the popular passwords

# Possible Solutions

- Two factor authentication (2FA)

- Server saves a list of **all** users' passwords and blacklists the popular passwords
  - Put users' passwords at risk: new single point of failure

# Possible Solutions

- Two factor authentication (2FA)

- Server saves a list of **all** users' passwords and blacklists the popular passwords
  - Put users' passwords at risk: new single point of failure

- Blacklisting **known** popular passwords
  - From previous breaches
  - Known lists of popular passwords

# Passwords Distribution Over Time

# Passwords Distribution Over Time

- password -> passw0rd -> p@assw0rd->password

# Passwords Distribution Over Time

- password -> passw0rd -> p@assw0rd->password
- superman -> wonderwoman

# Passwords Distribution Over Time

- password -> passw0rd -> p@assw0rd->password
- superman -> wonderwoman
- Different populations

# Passwords Distribution Over Time

- password -> passw0rd -> p@assw0rd->password

- superman -> wonderwoman

- Different populations

# Primum Non Nocere

First do (almost) no harm

# Primum Non Nocere

## First do (almost) no harm

- Publishing passwords blacklist can also help attackers

# Primum Non Nocere

First do (almost) no harm

- Publishing passwords blacklist can also help attackers
  - Attacker can use **auxiliary data** to guess password distribution

# Primum Non Nocere

## First do (almost) no harm

- Publishing passwords blacklist can also help attackers
  - Attacker can use **auxiliary data** to guess password distribution
  - Publishing the blacklist is like publishing a **code vulnerability**

# Primum Non Nocere

## First do (almost) no harm

- Publishing passwords blacklist can also help attackers
  - Attacker can use **auxiliary data** to guess password distribution
  - Publishing the blacklist is like publishing a **code vulnerability**
- Leaking password information can hurt the user

# Primum Non Nocere

## First do (almost) no harm

- Publishing passwords blacklist can also help attackers
  - Attacker can use **auxiliary data** to guess password distribution
  - Publishing the blacklist is like publishing a **code vulnerability**
- Leaking password information can hurt the user
  - Gathering statistics requires some password information

# Primum Non Nocere

## First do (almost) no harm

- Publishing passwords blacklist can also help attackers
  - Attacker can use **auxiliary data** to guess password distribution
  - Publishing the blacklist is like publishing a **code vulnerability**
- Leaking password information can hurt the user
  - Gathering statistics requires some password information
  - One bit leakage doesn't hurt the user a lot (next slide)

# Primum Non Nocere

## First do (almost) no harm

- Publishing passwords blacklist can also help attackers
  - Attacker can use **auxiliary data** to guess password distribution
  - Publishing the blacklist is like publishing a **code vulnerability**
- Leaking password information can hurt the user
  - Gathering statistics requires some password information
  - One bit leakage doesn't hurt the user a lot (next slide)
  - Differential privacy can also help

# The Password Game

- PGame(L): Attacker A wants to attack device D
  - Publishes a list with L guesses for passwords
  - Wins if the password of D is in the list
- Effect of one bit leakage on password:
  - If A wins PGame(L) w.p at least $\delta$ using a 1 bit leak
  
  **implies**
  - There is A’ that wins PGame(2L) w.p $\delta$ without a leak
- $\epsilon$-Differential Privacy
  - If A wins PGame(L) w.p at least $\delta$ using $\epsilon$-DP information
  
  **then**
  - There is A’ wins PGame(L) w.p at least $\delta \cdot e^{-\epsilon}$ **without a leak**

# How to (not) Share a Password: **Desiderata**

- Identify and **blacklist** popular passwords (heavy hitters)
  - Those chosen by more than a fraction $\tau$ of the users
- Server should not learn ``more than 1 bit" on any user's password
  - This (at most) halves the number of password guesses
- Probability of False Negatives (pFN) must be **negligible**
  - **No popular password is missed**
- Probability of False Positives (pFP) should be small
  - A legitimate password **may** be rejected with **low probability**
    - **Most legit passwords OK**

# Previous Work

- Finding **heavy hitters** in many settings - [DNP+10,DNPR10,CSS11,CLSX12, HKR12,DNRR15]
- Semi-honest version [BS15,BNST17]
- Non colluding mix servers – [MS17]

- DP password list with trusted server – [BDB16]
- Similar motivation, no DP – [SHM10]

Why is this work different from all the other works?

# The Malicious World

- **Both** users and server might be malicious


- A malicious server wants to learn the passwords

- Malicious users want to “hide” popular passwords
  - **Adversary controls a coalition of users**
    - **Want to hide weak passwords of other users**

# MPC Meets **DP** in the **Malicious** World

Security requirements in the protocol are **asymmetric**:

- Relatively easy to protect users' privacy from server
- Harder to protect against **colluding malicious** users
- E.g. how we can prevent **cheating** in randomized response

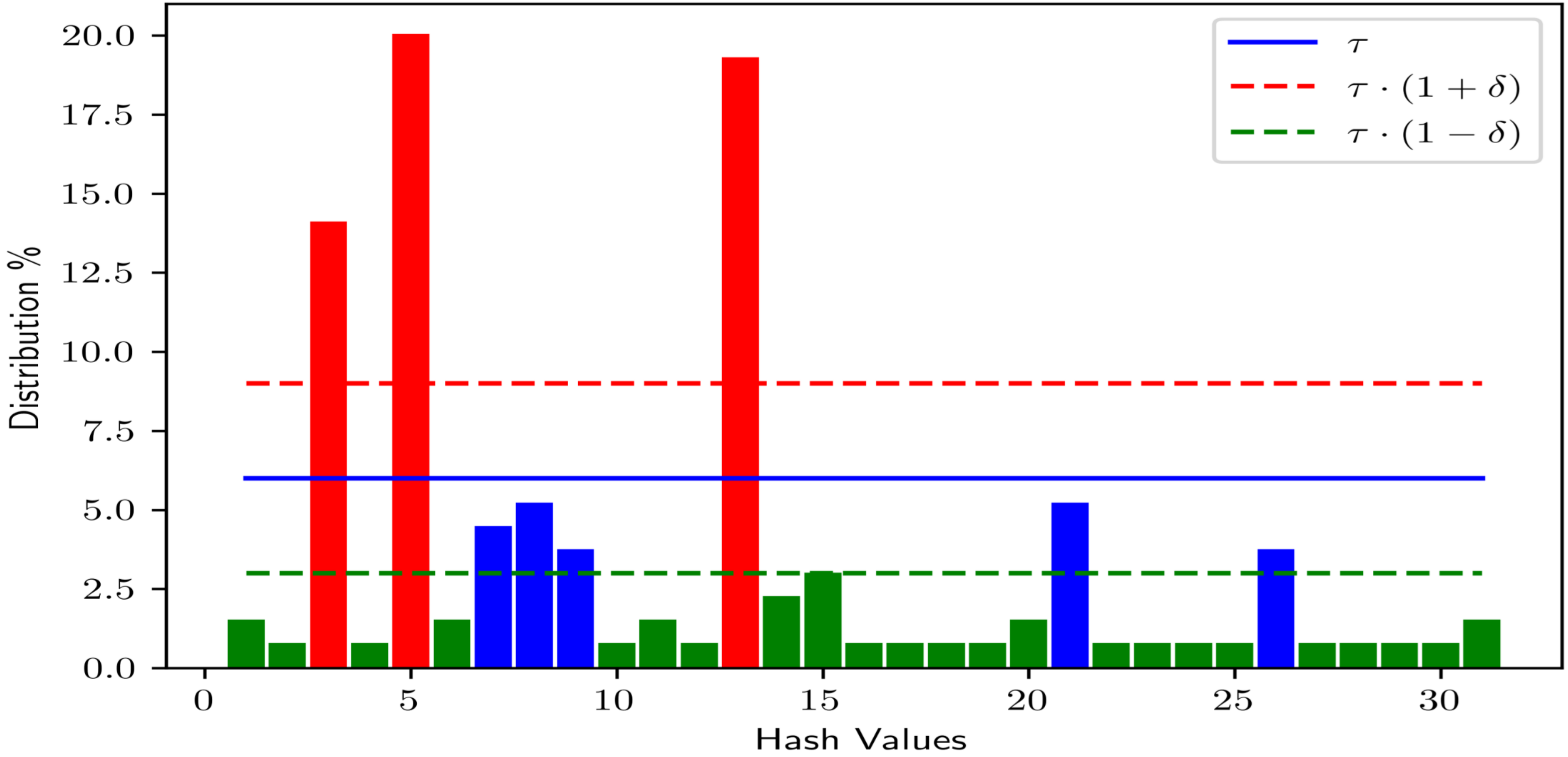
- Use **efficient 2PC protocol tailored to the system's correctness requirements**

# Correctness

- Password used by at least a $(1+\delta)\tau$ fraction of the users: **identified as a heavy hitter** w.p **at least** (1-pFN)
  - Even at the presence of malicious user coalition

- Password used by at most a $(1-\delta)\tau$ fraction of the users: identified as a heavy hitter w.p **at most** pFP

# Creating the Hash Black List



# The Semi Honest Solution

Bassily, Nissim, Stemmer, Thakurta

- Similar to the heavy hitters solution of [BNSTS17]
- Hash the passwords to $\ell$ bits values
  - "Naïve" hash function
  - We identify popular hash values
  - Ban all passwords hashed to these values
  - May have a small chance of collisions
- Server initializes to zero a **counter histogram** T of size $2^{\ell}$

# The Semi-Honest Protocol

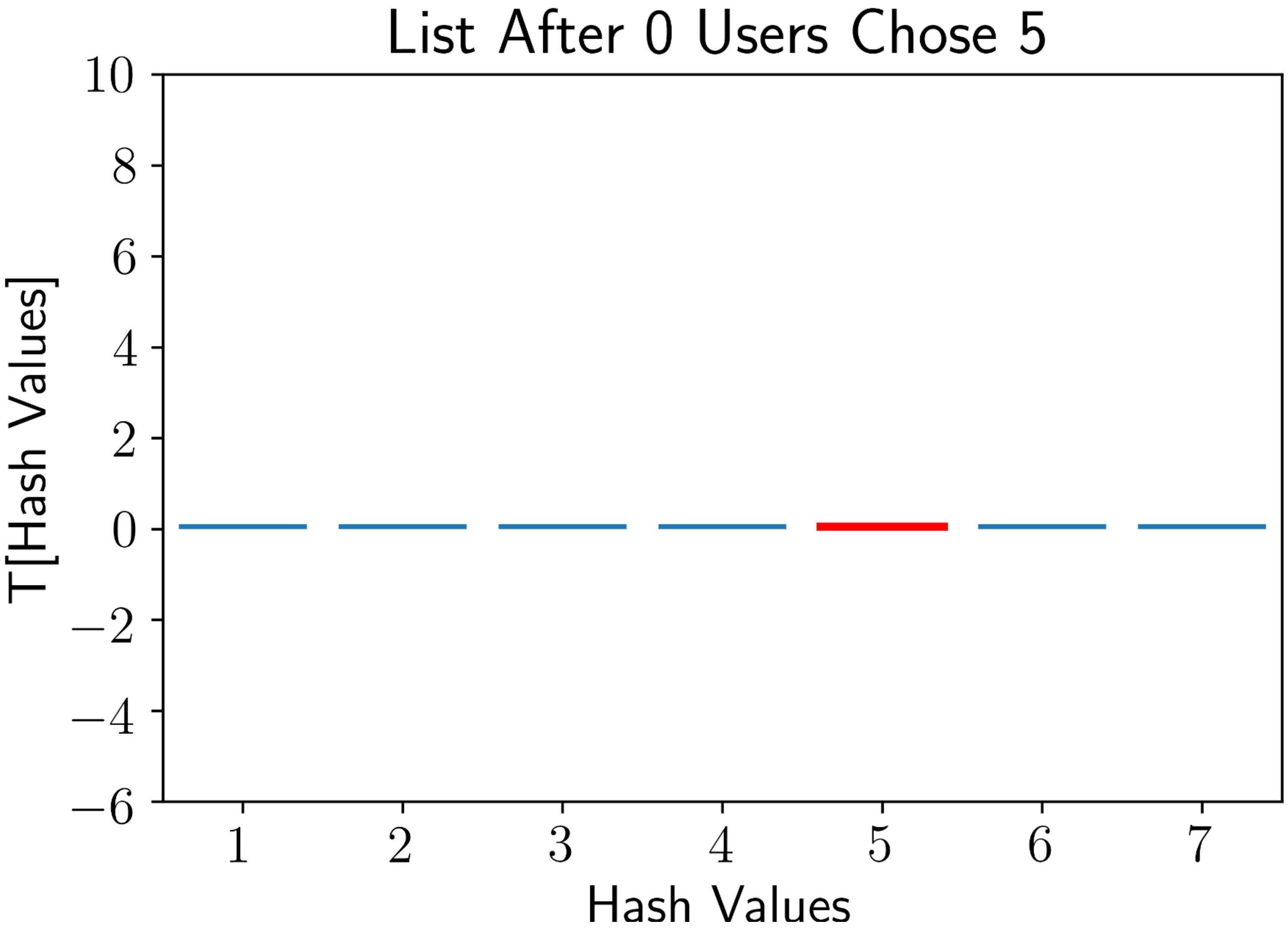
- For every user:

$$\text{Server} \xrightarrow{\text{random } r \in \{0,1\}^\ell} \text{User}$$

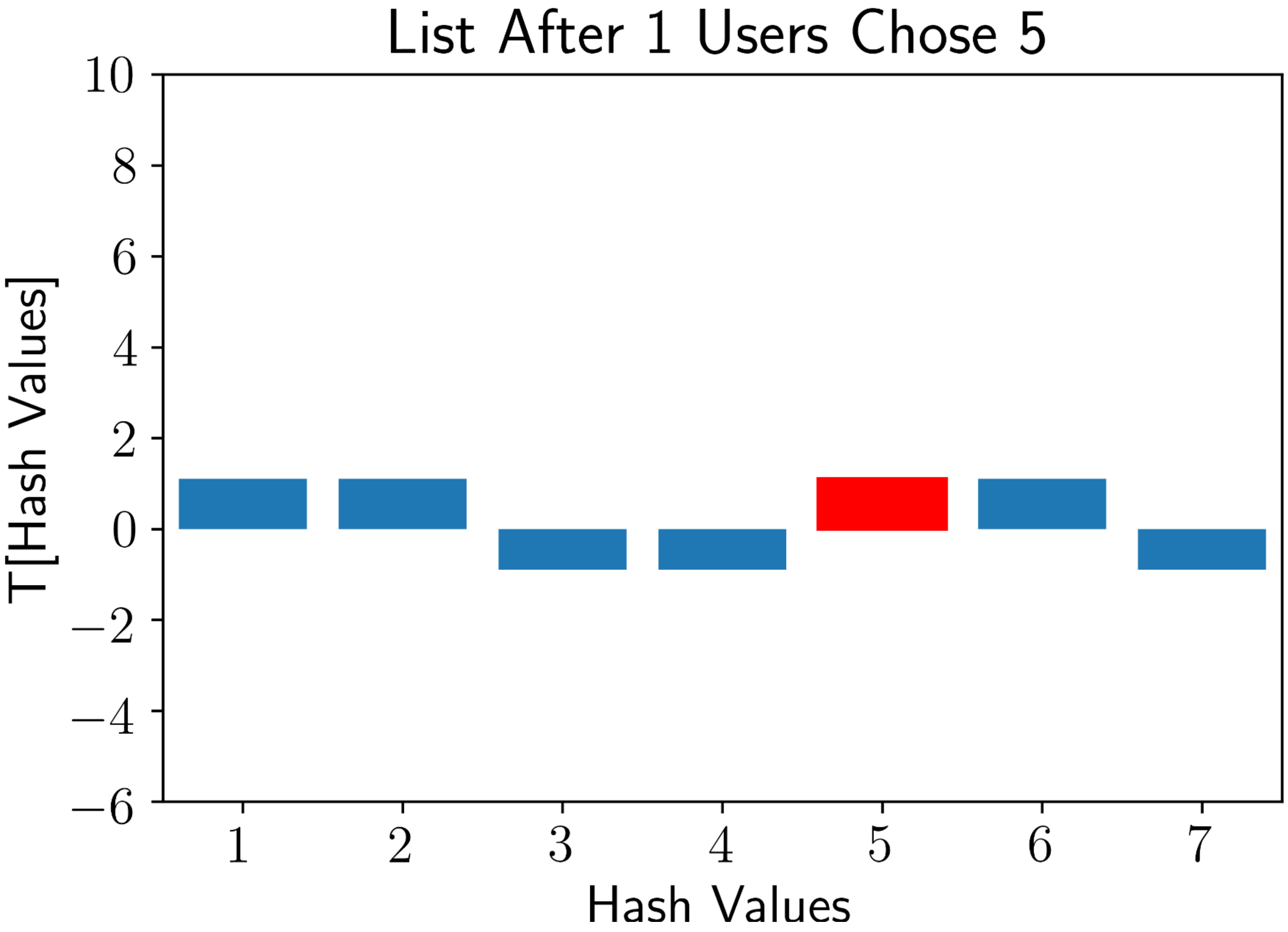
$$\text{Server} \xleftarrow{v = \langle H(pass), r \rangle} \text{User}$$

- Server iterates over all possible value of $x \in \{0,1\}^\ell$
    - If $v = \langle x, r \rangle$: $T[x] \mathrel{+}= 1$
    - Else: $T[x] \mathrel{-}= 1$

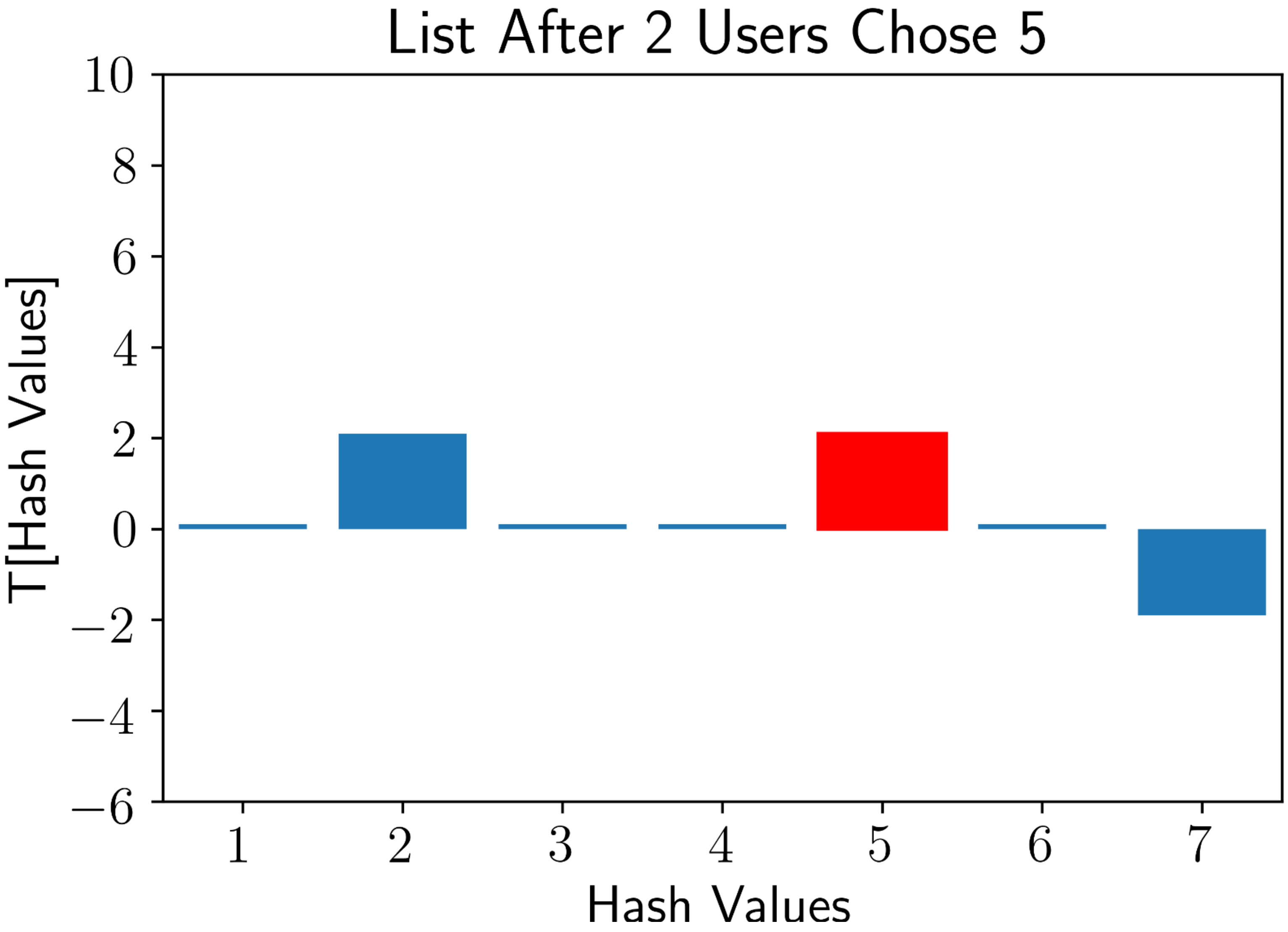
# The Semi-Honest Protocol



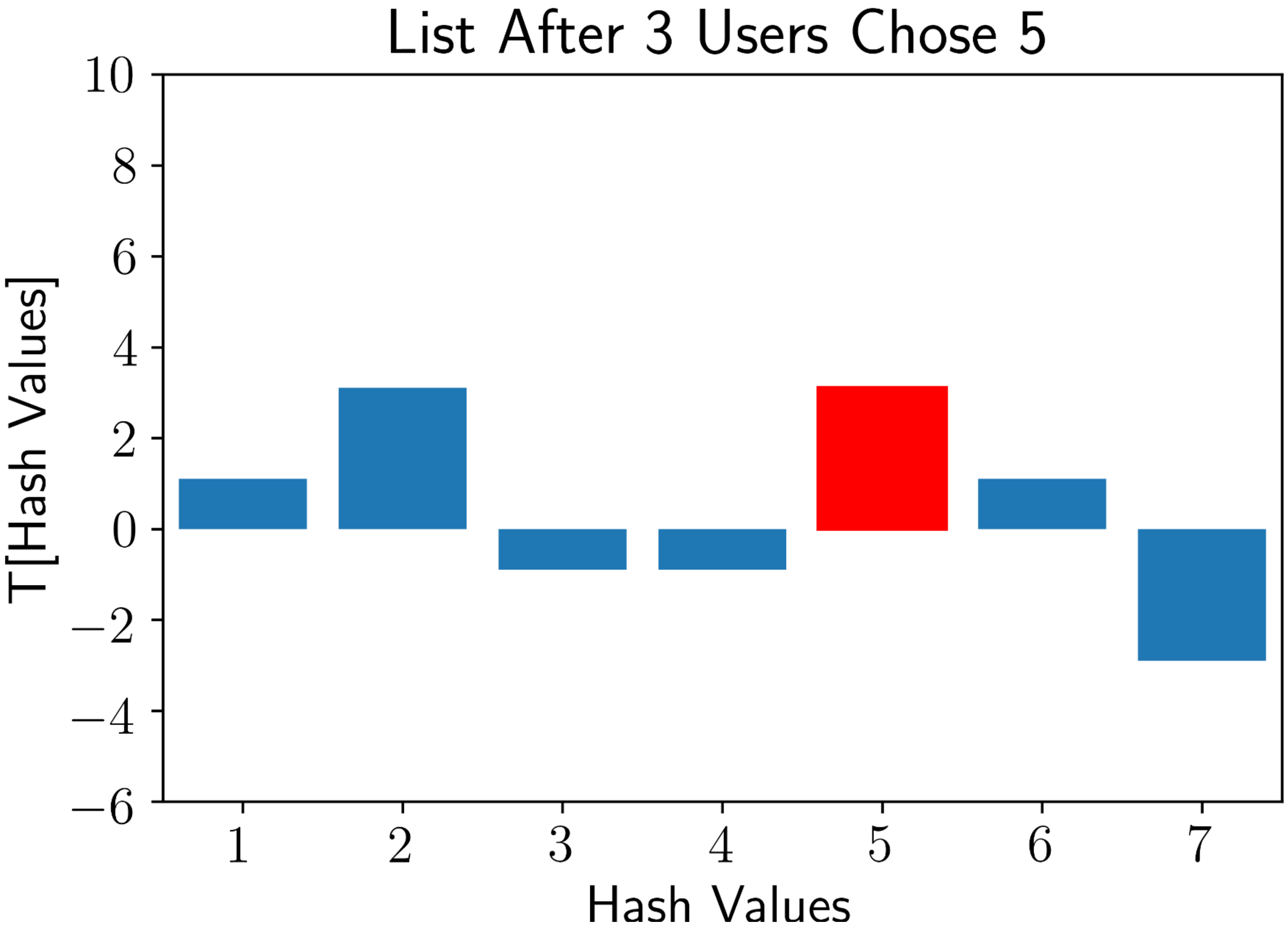
# The Semi-Honest Protocol



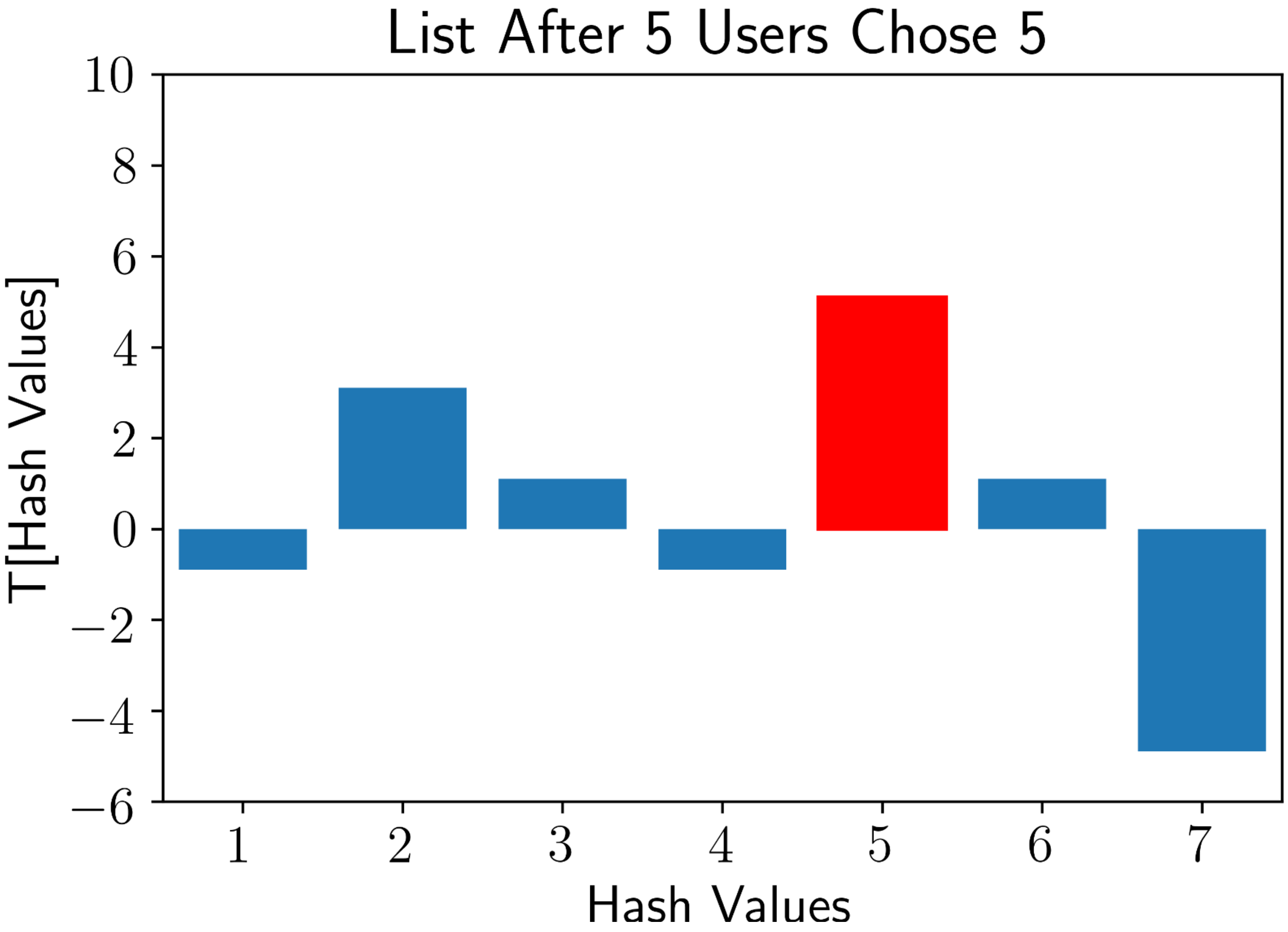
# The Semi-Honest Protocol



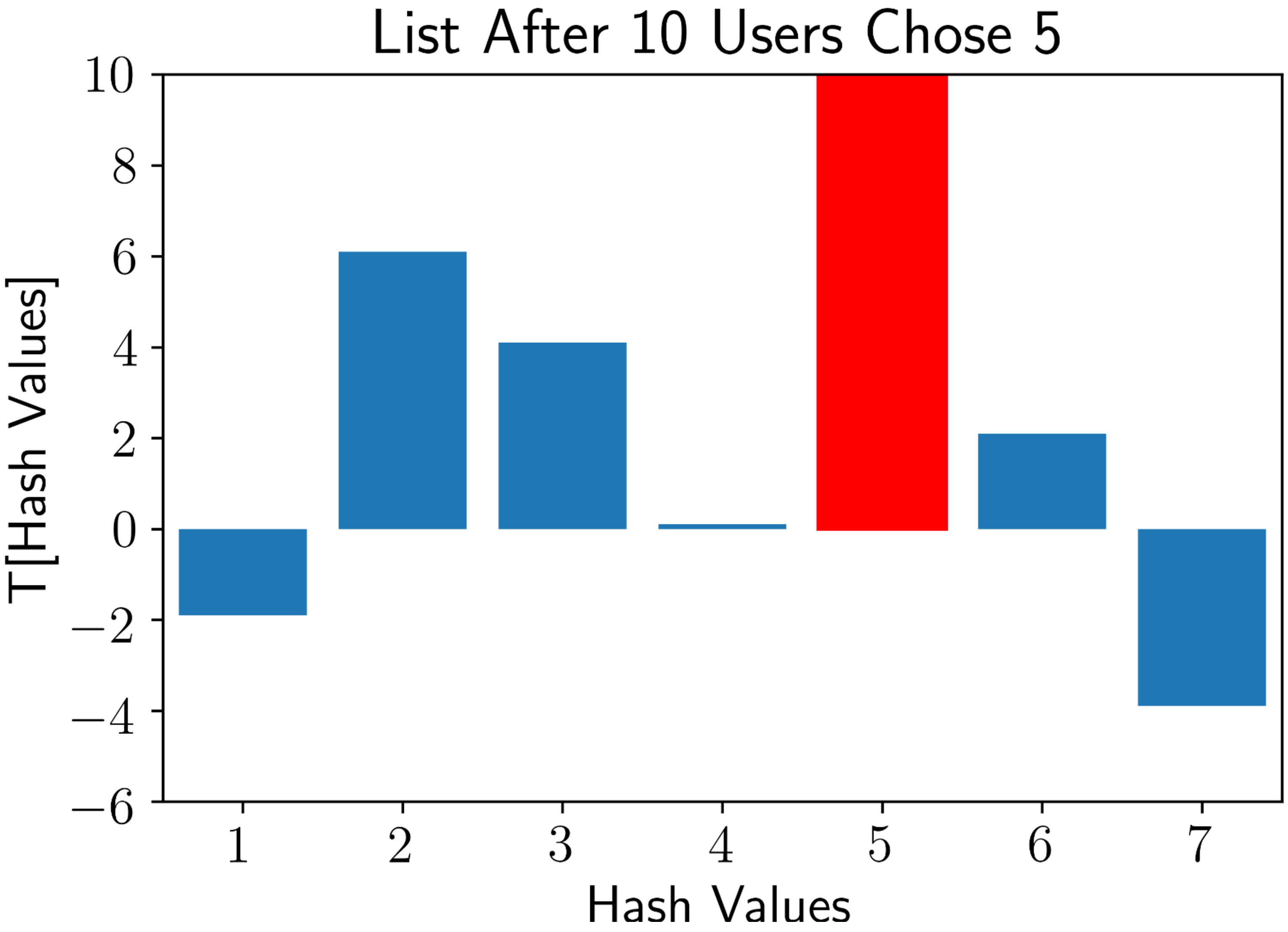
# The Semi-Honest Protocol



# The Semi-Honest Protocol



# The Semi-Honest Protocol



# The Semi-Honest Solution

- $T[x] = N * Prob(x) + Noise$
  - $Noise \sim Bin(N * (1 - Prob(x)), 0.5)$
- $E[T[x]] = N * Prob(x)$

- **Blacklist** the hash value if $T[x] > \tau N$

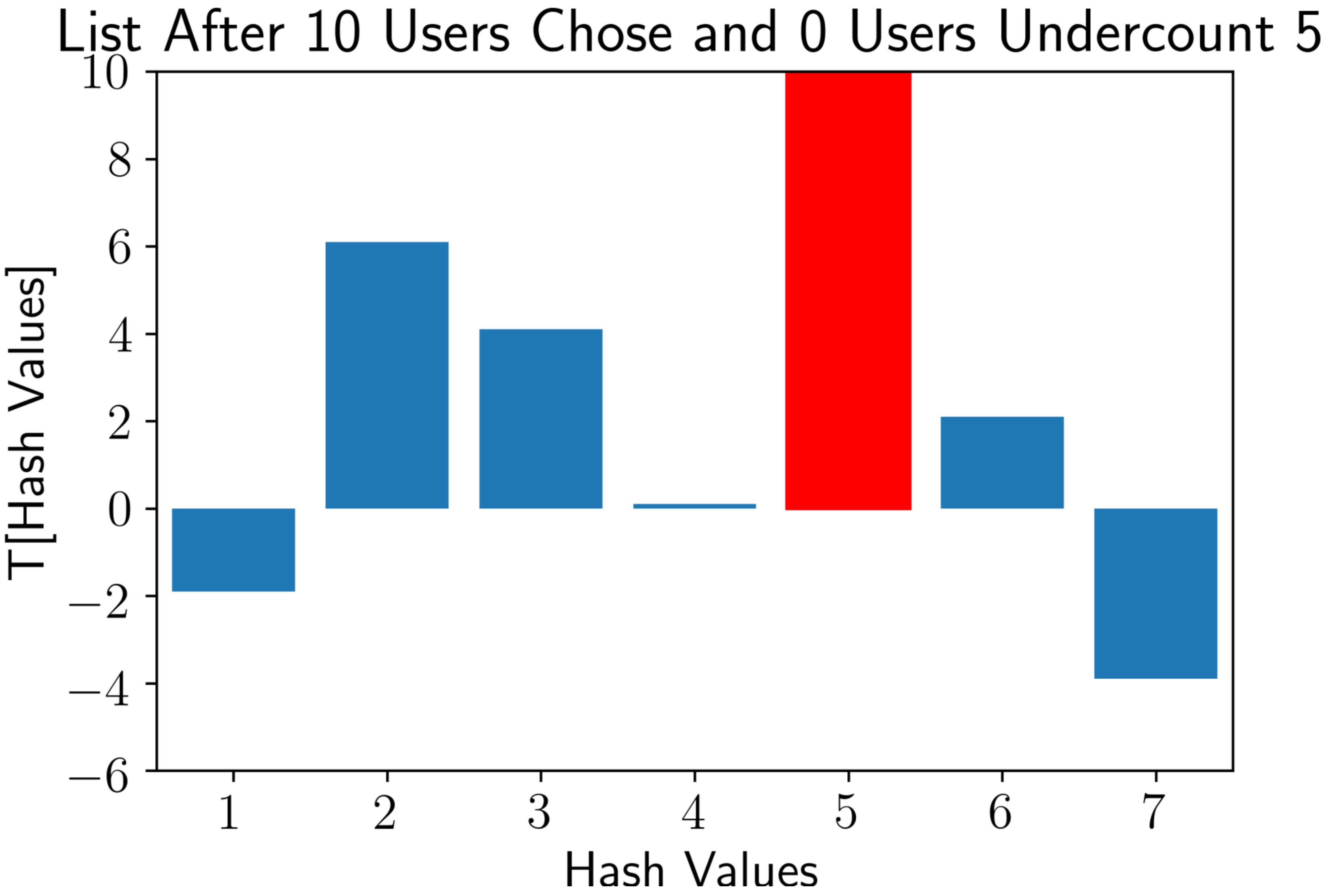
- Define $\tau$ as a function of N and $\delta$ such that:
$$Prob[|Noise| > \tau N\delta] < pFN$$

# The Undercount Attack

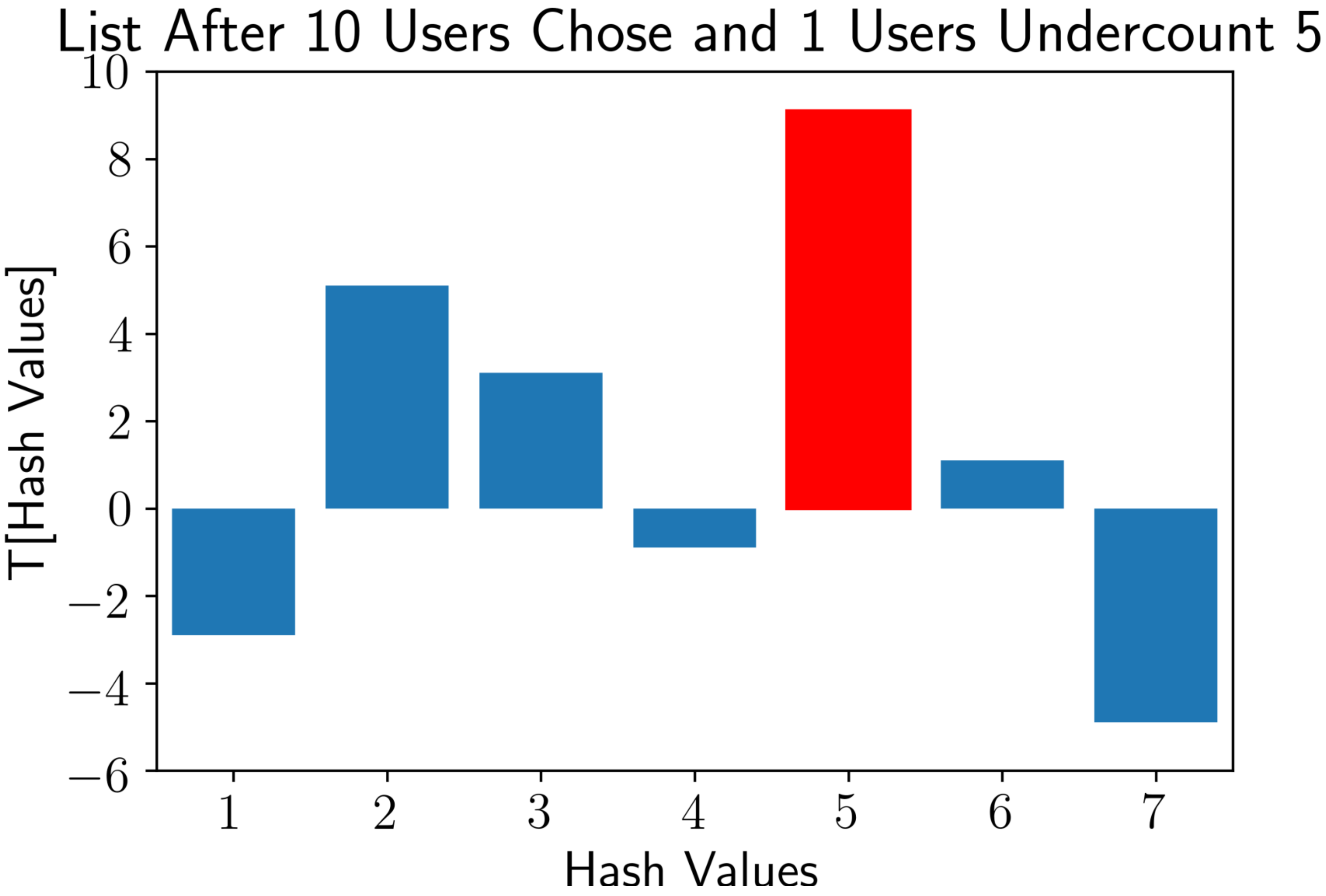
- An attacker-user wants to “hide” a popular password pass
- All users controlled by the attacker simply send:

$$1 - \langle H(pass), r_i \rangle$$

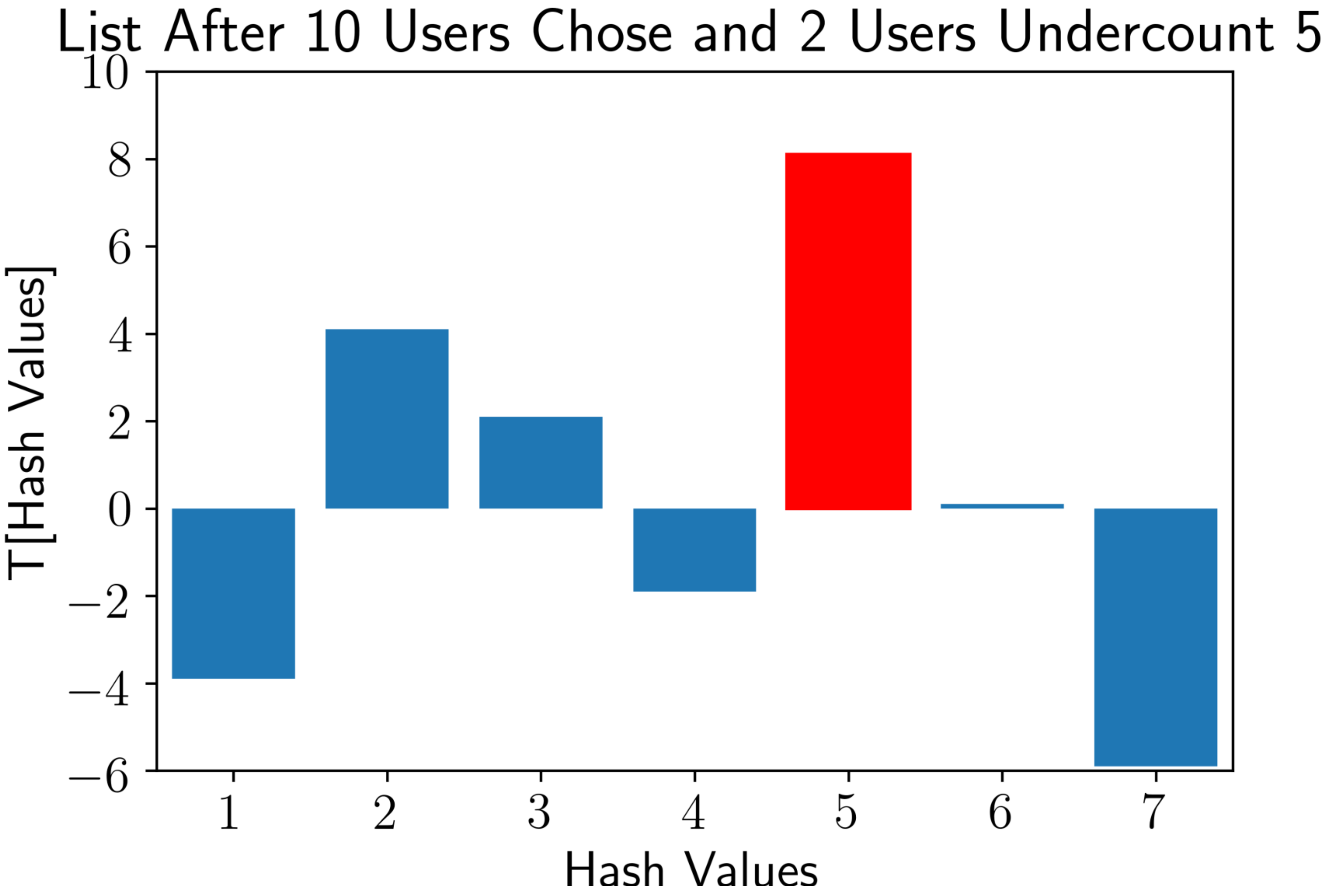
# The Undercount Attack



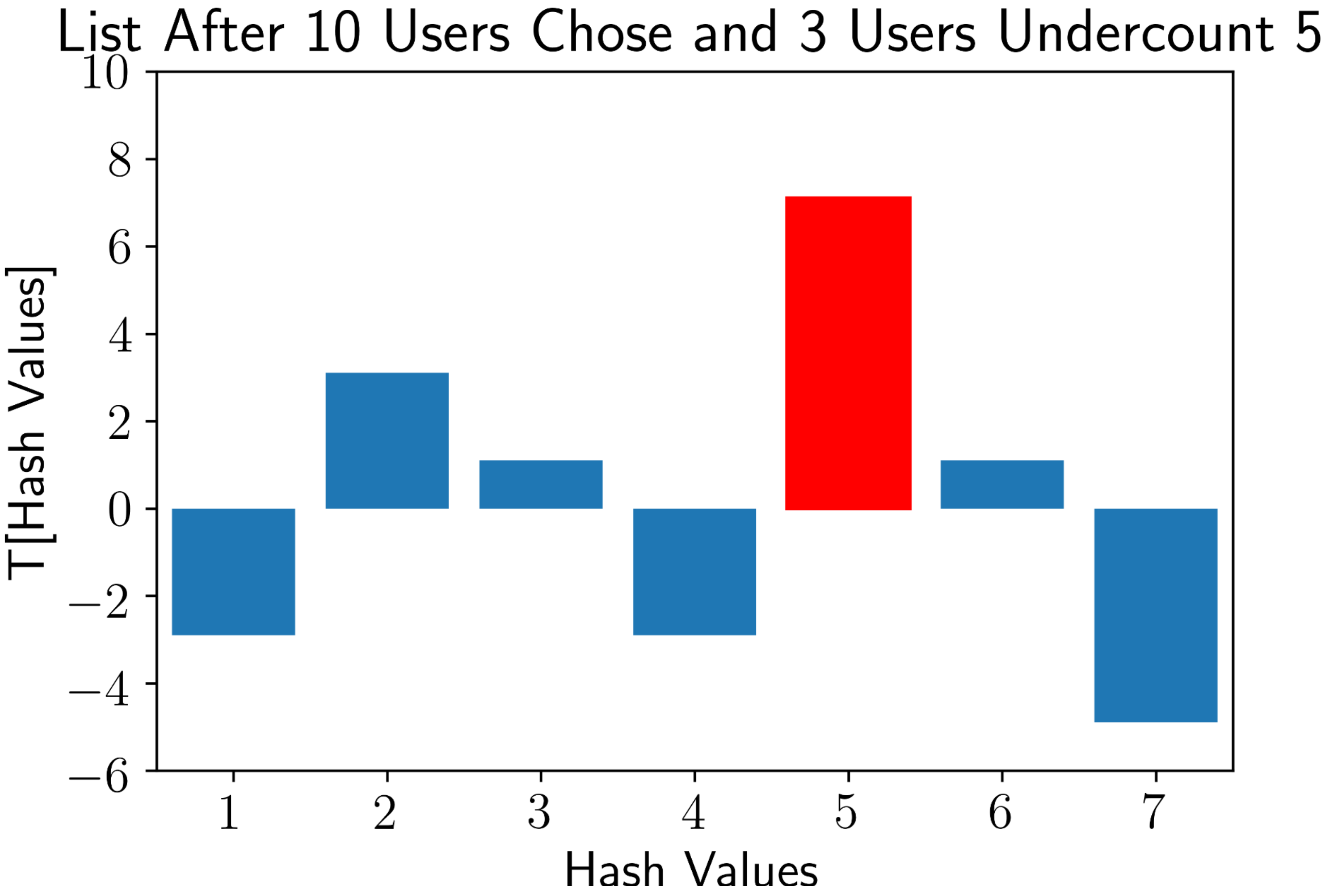
# The Undercount Attack



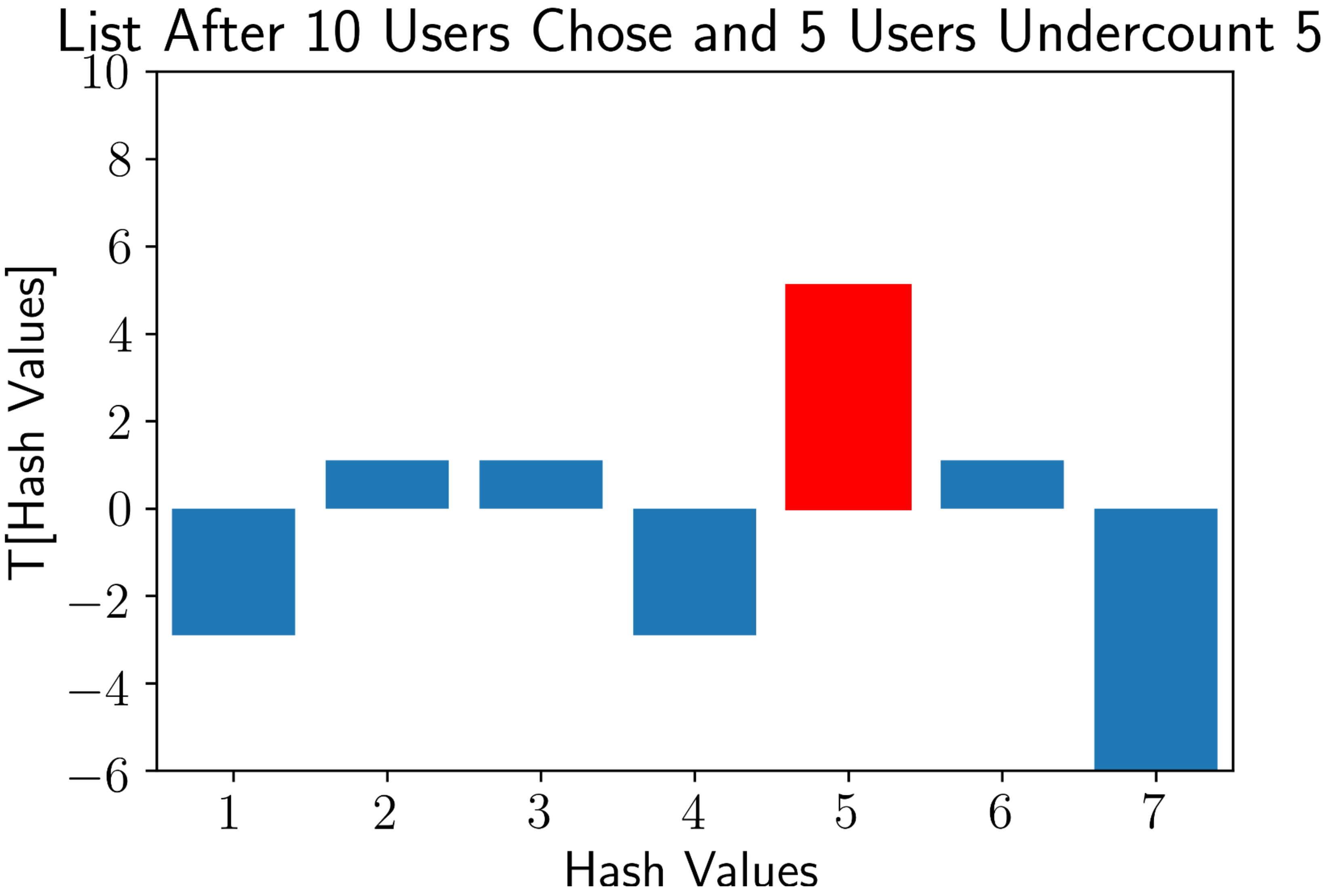
# The Undercount Attack



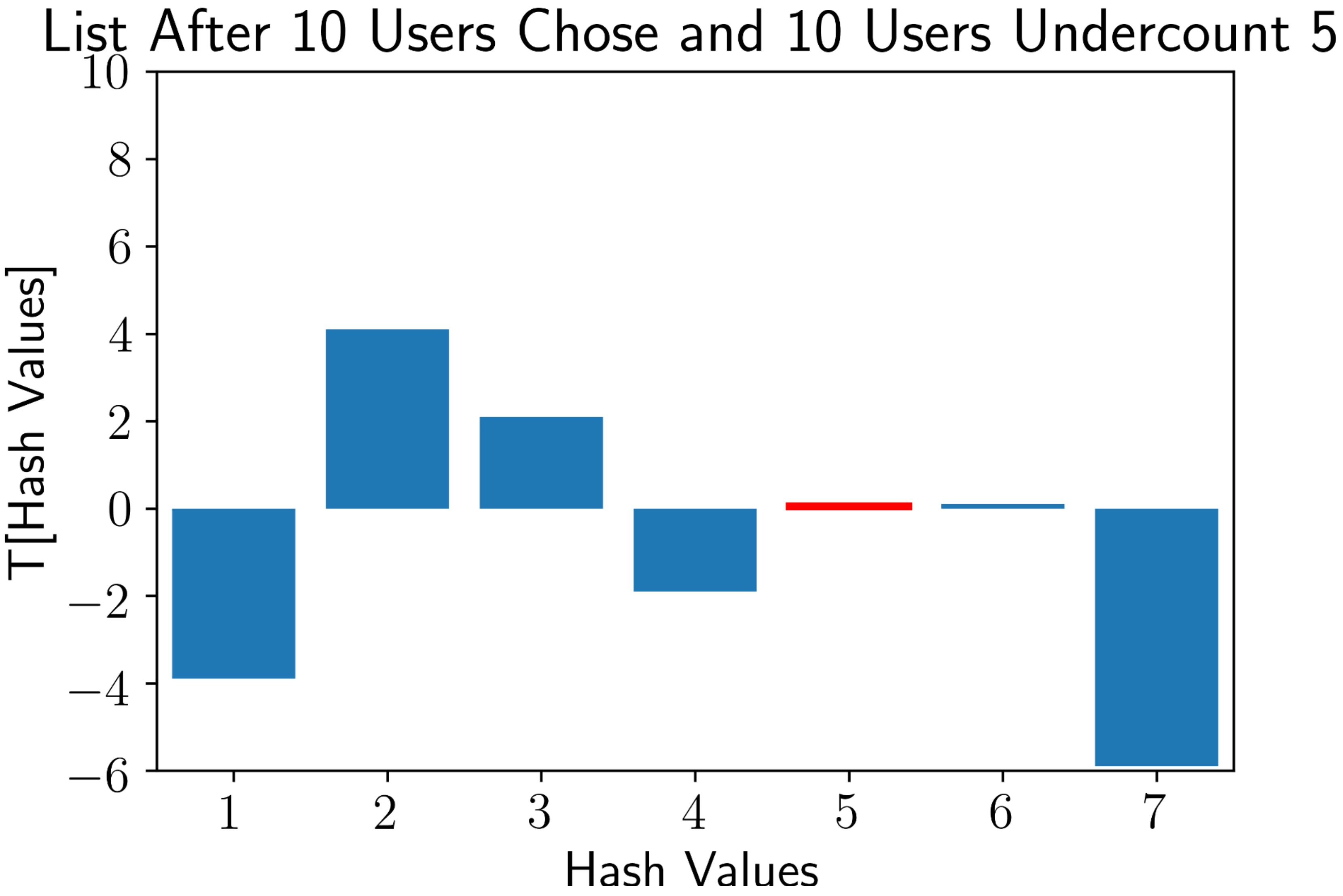
# The Undercount Attack



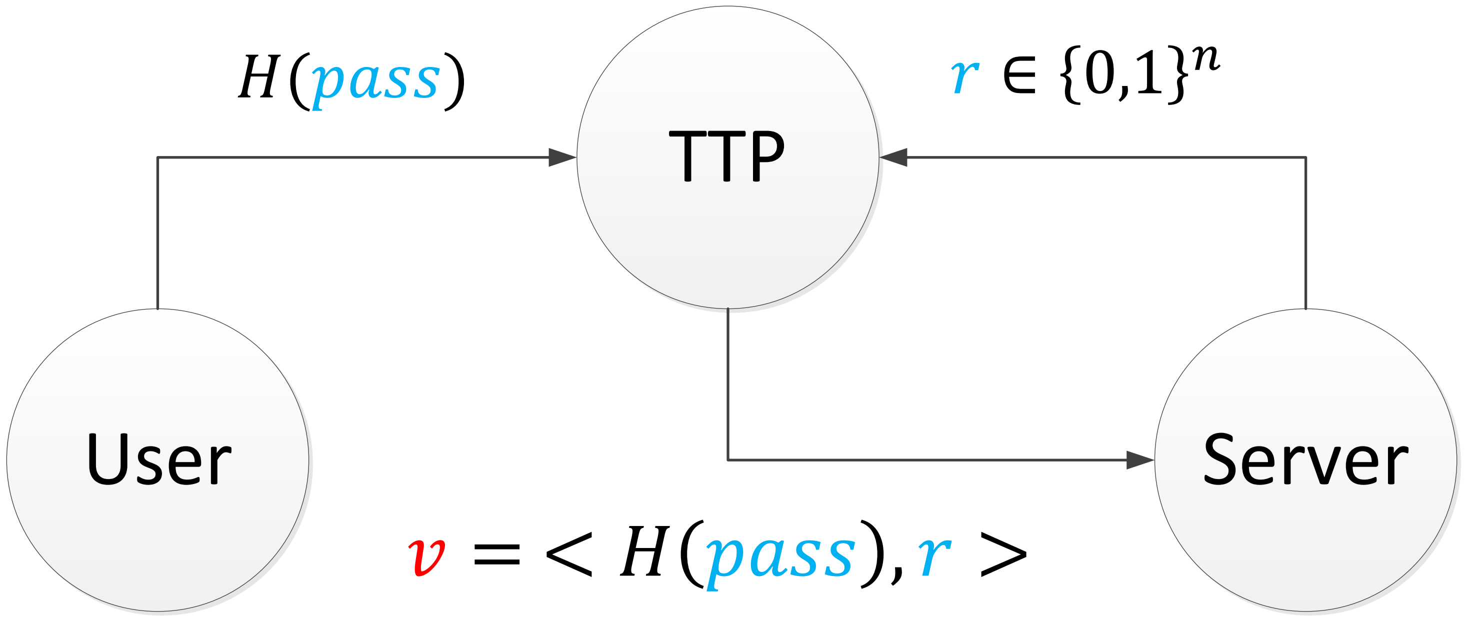
# The Undercount Attack



# The Undercount Attack



# The Required Functionally




- Two approaches:
  - QR based
  - Yao's **garbled circuit based**

# A Naïve QR Based Solution

For **N=pq** hard to distinguish squares (QR) from non-squares (nQR) among those with Jacobi Symbol 1

- Based on the **intractability** of the quadratic residuosity (QR) **assumption**

0 encrypted as a QR and 1 as nQR

- Encrypt the r vector **as in the Goldwasser-Micali public encryption** scheme
  - The server generates an RSA modulus N=pq, p and q primes
  - Encrypt the bits of r=$r_1, r_2 \ldots, r_\ell$ into $c_1, c_2, \ldots, c_\ell$

$$e = d^2 \cdot \prod_{i=1}^{\ell} (c_i)^{v_i} \quad \text{where} \quad d \xleftarrow{R} \mathbb{Z}_N$$

Is it secure?

**Not if adversary knows an nQR**

# The nQR Generation Assumption

- Is it hard to generate a nQR number w.h.p?
  - With probability better than $\frac{1}{2} + negl$?

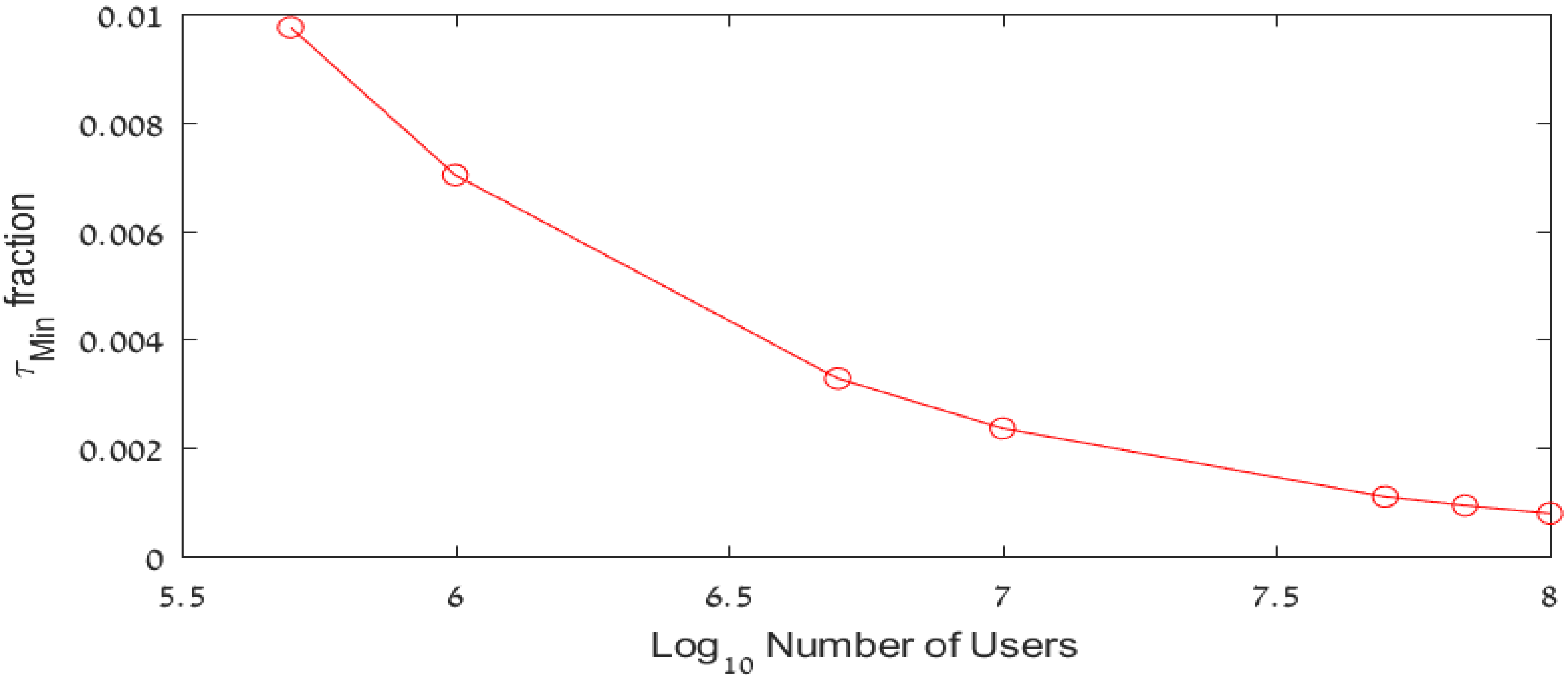
*Remarks about Theorem* 2. When the factorization of $n$ is secret, no efficient algorithm for selecting a quadratic nonresidue mod $n$ is known. Thus it may be that revealing, say, the smallest quadratic nonresidue in $Z_n^1$ may endanger the secrecy of the factorization of $n$ or make deciding quadratic residuosity modulo $n$ easy.

- Simple reduction from protocol security
  - Assuming Unique N for each device

# Solution based only on QR assumption

- Adding an Interactive zero knowledge proof that the inner product was computed correctly
- Non interactive version based on Fiat-Shamir
- Requires proof that N=pq where p and q are primes
- We have another solution based on garbled circuit

# Malicious Bounds On $\tau$




Blacklist top 3 passwords of the Yahoo! Leak, and top 5 passwords of the RockYou leak

# Implementation and Other Usages

- Implemented the full malicious QR protocol on a Raspberry Pi
  - Non interactive version runs in about 15 seconds
    - **Only calculated when the user changes his password**
    - Can run in background
  - Server computer can verify in about 0.5 seconds

- Same solution can be used in any heavy hitters problem with possible malicious setting
  - **TOR network statistics**
  - **Device PIN/Pattern**
  - **Large service providers dynamic passwords statistics**

# Open Questions

- Do we need Crypto?
  - For non-malicious users – no (computational based) crypto needed!
- Research on the nQR assumption
- Can we improve our bounds on $\tau$?
- Real world implementation with real passwords
- Questions?

**eprint.iacr.org/2018/003**
**eyalro.net**