

How to (**not**) Share a Password:
Privacy preserving protocols for finding
heavy hitters with adversarial behavior

Moni Naor

Benny Pinkas

Eyal Ronen

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

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 attack

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 attack
 - E.g. the Mirai attack

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 attack
 - E.g. the Mirai attack
 - Easy to find IoT devices with Shodan like search engines

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 attack
 - E.g. the Mirai attack
 - Easy to find IoT devices with Shodan like search engines
- Service provider liability?

Possible solutions

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



Panacea
Greek Goddess of
Universal Remedy
Solution to all
problems; **Cure-all**



Possible solutions

Possible solutions

- Two factor authentication (2FA)

Possible solutions

- Two factor authentication (2FA)



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 over time

Passwords over time

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

Passwords over time

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

Passwords over time

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

Passwords 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
 - Published 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 δ using a 1 bit leak **implies**
 - There is A' wins PGame(2L) w.p δ without a leak
- ϵ -DP
 - If A wins PGame(L) w.p at least δ using ϵ -DP information **then**
 - There is A' wins PGame(L) w.p $> \delta \cdot e^{-\epsilon}$ without a leak

How to (not) share a Password: Desiderata

- Identify and **blacklist** popular passwords (**heavy hitters**)
 - those were chosen by more than a fraction τ of the users

How to (not) share a Password: Desiderata

- Identify and **blacklist** popular passwords (**heavy hitters**)
 - those were chosen by more than a fraction τ of the users
- Server should not learn more than 1 bit on any user's password

How to (not) share a Password: Desiderata

- Identify and **blacklist** popular passwords (**heavy hitters**)
 - those were chosen by more than a fraction τ of the users
- Server should not learn more than 1 bit on any user's password
 - At most halves the number of password guesses

How to (not) share a Password: Desiderata

- Identify and **blacklist** popular passwords (**heavy hitters**)
 - those were chosen by more than a fraction τ of the users
- Server should not learn more than 1 bit on any user's password
 - At most halves the number of password guesses
- Probability of False Negative (pFN) must be **negligible**

How to (not) share a Password: Desiderata

- Identify and **blacklist** popular passwords (**heavy hitters**)
 - those were chosen by more than a fraction τ of the users
- Server should not learn more than 1 bit on any user's password
 - At most halves the number of password guesses
- Probability of False Negative (pFN) must be **negligible**
 - No popular password is missed

How to (not) share a Password: Desiderata

- Identify and **blacklist** popular passwords (**heavy hitters**)
 - those were chosen by more than a fraction τ of the users
- Server should not learn more than 1 bit on any user's password
 - At most halves the number of password guesses
- Probability of False Negative (pFN) must be **negligible**
 - No popular password is missed
- Probability of False Positive (pFP) may be a small value

How to (not) share a Password: Desiderata

- Identify and **blacklist** popular passwords (**heavy hitters**)
 - those were chosen by more than a fraction τ of the users
- Server should not learn more than 1 bit on any user's password
 - At most halves the number of password guesses
- Probability of False Negative (pFN) must be **negligible**
 - No popular password is missed
- Probability of False Positive (pFP) may be a small value
 - A legitimate password can be rejected with low probability

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]

The Malicious World

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

The Malicious World

- Both **users** and **server** might be malicious
- A malicious **server** wants to learn the passwords

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**

MPC meets DP in the Malicious World

- Asymmetric security requirements from the parties in the protocol
 - Relatively easy to protect users' privacy from server
 - Harder to protect against **colluding malicious** users
- Use efficient 2PC protocol tailored to the system's correctness requirements

Correctness

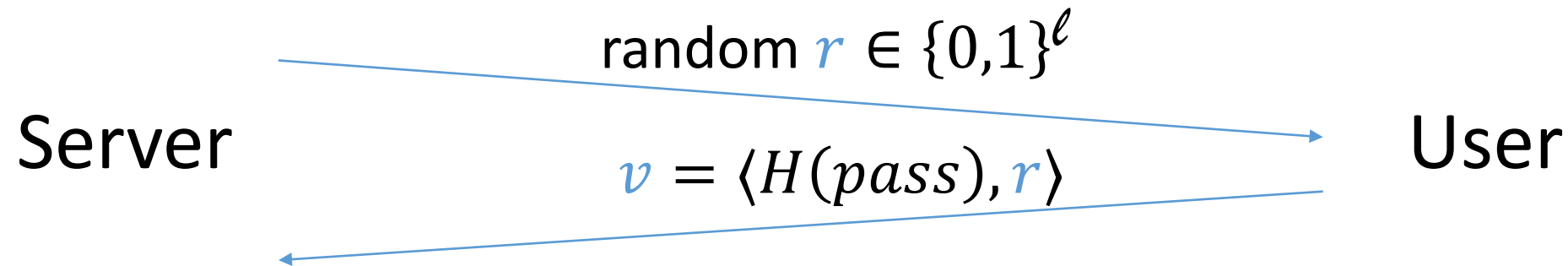
- Password used by at least a $(1 + \delta)\tau$ fraction of the users:
identified as a heavy hitter w.p at least $(1 - p_{FN})$
 - Even at the presence of malicious user coalition
- Password used by at most a $(1 - \delta)\tau$ fraction of the users:
identified as a heavy hitter w.p at most p_{FP}

The semi honest solution

- Similar to the heavy hitters solution of [BNSTS17]
- We hash the passwords to ℓ bits values
 - “Naïve” hash function
 - We assume collisions
 - OK if $1/(1 - \delta)\tau \ll 2^\ell$
- Server initializes to zero a **counter histogram** T of size 2^ℓ

The semi honest Protocol

- For every user:



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

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

The undercount attack

- A user wants to “hide” a popular password *pass*
- The user simply sends: $1 - \langle H(\textit{pass}), r \rangle$

The required functionality

- Input
 - The server sends to the Trusted Third Party (TTP) an ℓ bit input r
 - The user sends to the TTP an ℓ bit input v
- Output
 - The TTP sends to the server $\langle v, r \rangle$
 - The user gets no output
- Two approaches:
 - QR based
 - Yao's garbled circuit based

A naïve QR based solution

- Based on the intractability of the quadratic residuosity (QR) assumption
- We encrypt the r vector as in the Goldwasser-Micali public encryption scheme
- The server generates an RSA modulus $N=pq$, p and q primes
- We encode the bits of r into r^p , 0 as QR and 1 as nQR

$$e = d^2 \cdot \prod_{i=1}^{\ell} (r_i^p)^{V_i^s} \quad \text{where} \quad d \stackrel{R}{\leftarrow} \mathbb{Z}_N$$

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 $\frac{1}{2} + \epsilon$?

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

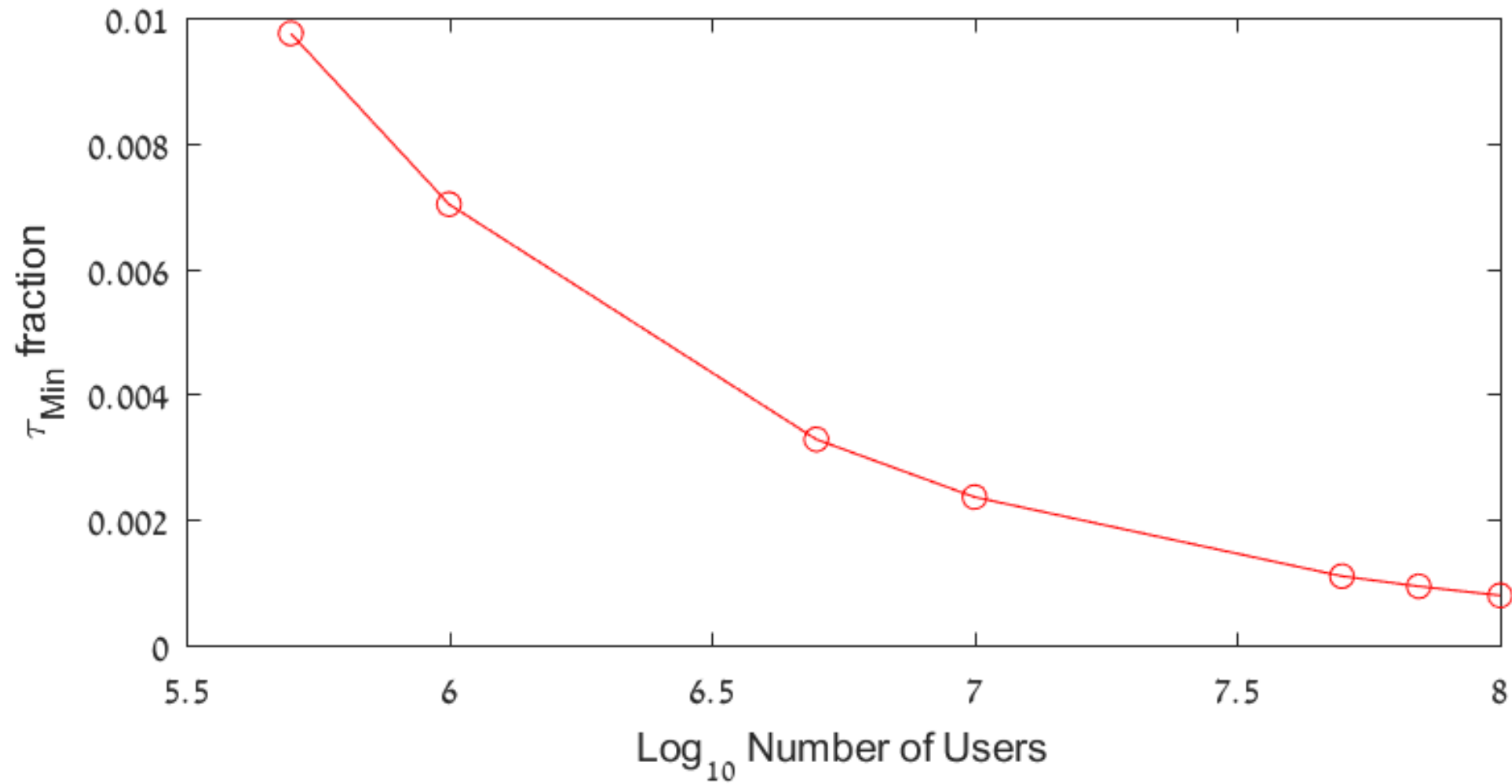
Reduction to nQR generation assumption

- The honest algorithm $A(v, r^p)$ return $e = \langle v, r \rangle$
- The adversary algorithm $A'(v, r^p)$ return $e = 1 - \langle v, r \rangle$
w.p $\frac{1}{2} + \delta$
- Generate r^p by random sample form \mathbb{Z}_N with Jacobi symbol 1
- Generate random v
- Return $A(v, r^p) \cdot A'(v, r^p)$, result nQR w.p $\frac{1}{2} + \delta$

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
- Another garbled circuit solution

Malicious bounds on τ



Implementation and other usages

- We implemented the full malicious QR protocol on a RPi
 - Non interactive version runs in about 15 seconds, 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!
- Can the attacker really use the leaked information from the blacklist publications?