The TypTop System

Personalized Typo-Tolerant Password Checking

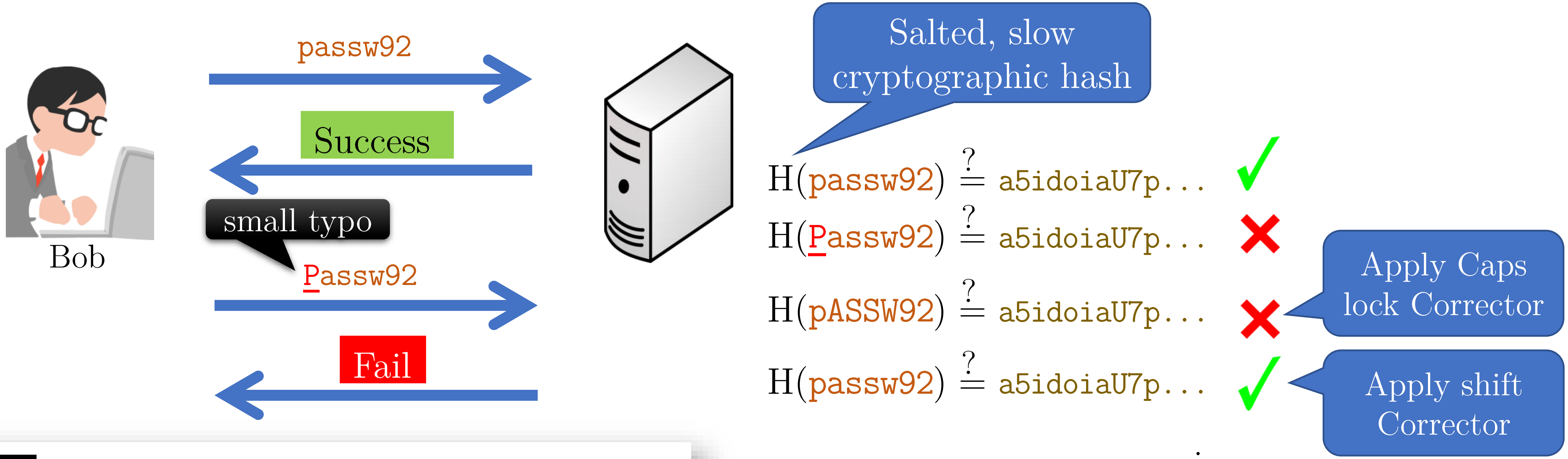
R. Chatterjee, J. Woodage, Y. Pnueli, A. Chowdhury, T. Ristenpart

CORNELL TECH
HOME OF THE JACOBS INSTITUTE

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Technion
Israel Institute of Technology

Password checking systems and typos

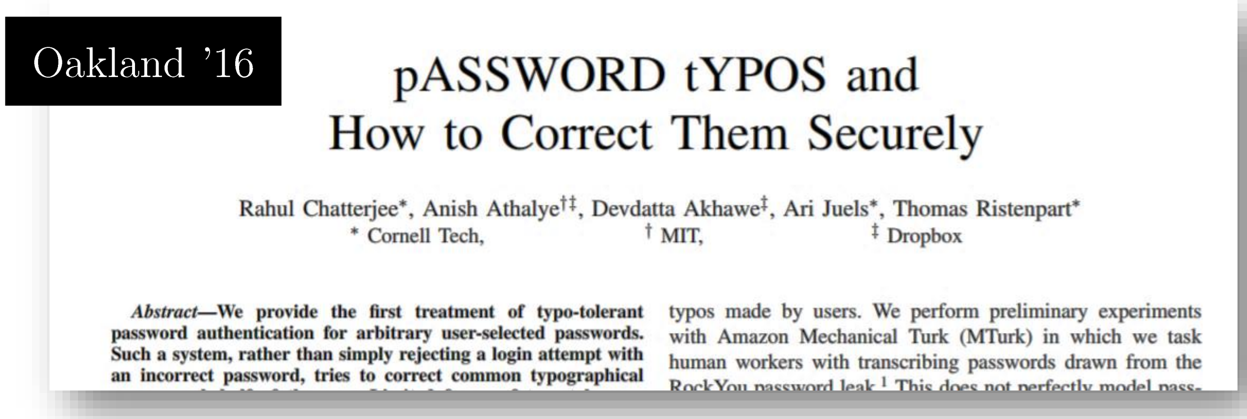


$\mathrm{H}(\texttt{passw92}) \overset{?}{=} \texttt{a5idoiaU7p...}$ ✓

$\mathrm{H}(\texttt{Passw92}) \overset{?}{=} \texttt{a5idoiaU7p...}$ ✗

$\mathrm{H}(\texttt{pASSW92}) \overset{?}{=} \texttt{a5idoiaU7p...}$ ✗

$\mathrm{H}(\texttt{passw92}) \overset{?}{=} \texttt{a5idoiaU7p...}$ ✓



Oakland '16

pASSWORD tYPOS and
How to Correct Them Securely

Rahul Chatterjee*, Anish Athalye†‡, Devdatta Akhawe‡, Ari Juels*, Thomas Ristenpart*
* Cornell Tech, † MIT, ‡ Dropbox

Abstract—We provide the first treatment of typo-tolerant password authentication for arbitrary user-selected passwords. Such a system, rather than simply rejecting a login attempt with an incorrect password, tries to correct common typographical

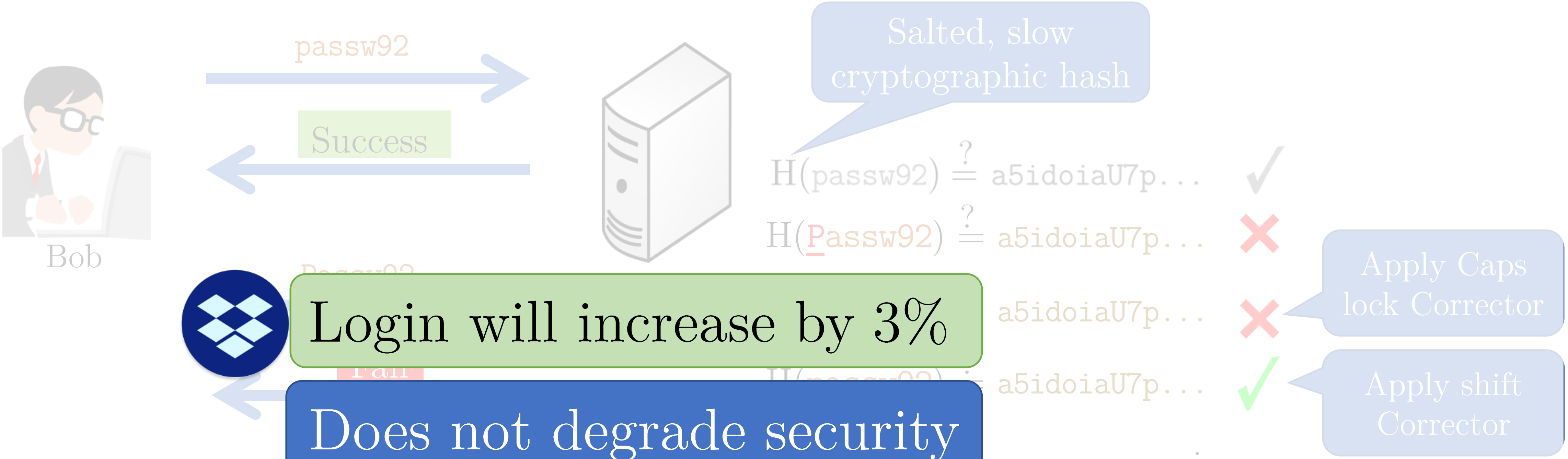
typos made by users. We perform preliminary experiments with Amazon Mechanical Turk (MTurk) in which we task human workers with transcribing passwords drawn from the RockYou password leak.[1] This does not perfectly model pass-

Top-5 correctors correct 20% of all typos

Typo-tolerant password checking

Allow registered password or typos of it

Typo-tolerance improves utility

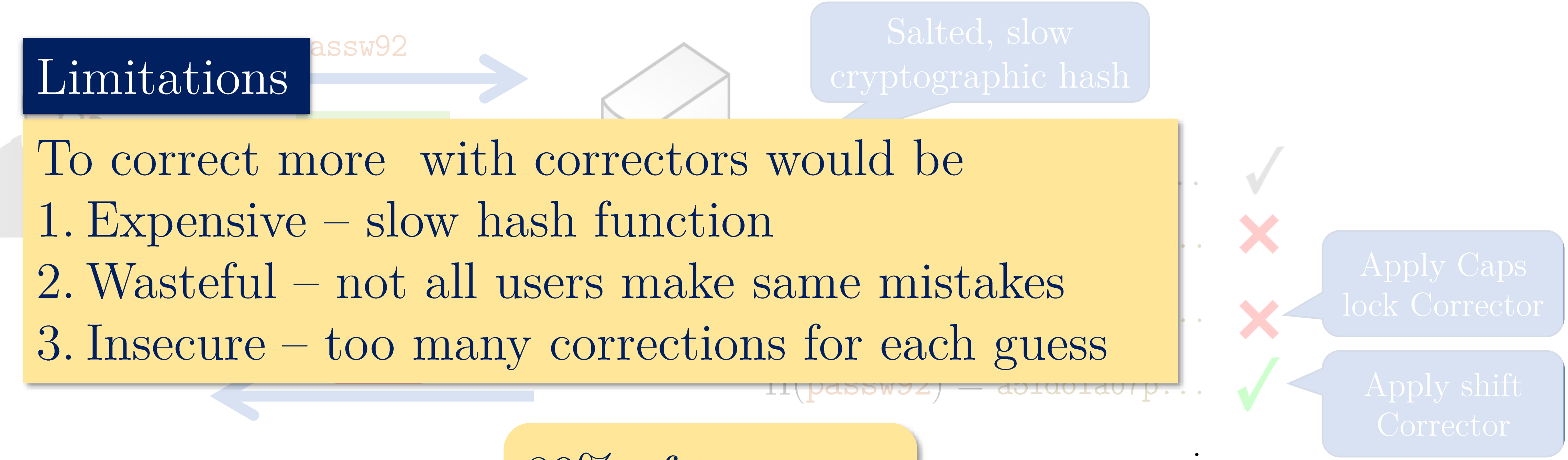

passw92
Success
Bob
Passw92
Fail
Salted, slow cryptographic hash
H(passw92) ≟ a5idoiaU7p...
H(Passw92) ≟ a5idoiaU7p...
a5idoiaU7p...
H(passw92) ≟ a5idoiaU7p...
Apply Caps lock Corrector
Apply shift Corrector

Login will increase by 3%

Does not degrade security

Top-5 correctors correct 20% of all typos

… corrects only the tip of the iceberg


assw92
Salted, slow cryptographic hash
H(passw92) = a5fd0fa07p. . .
Apply Caps lock Corrector
Apply shift Corrector

Limitations

To correct more with correctors would be

1. Expensive – slow hash function
2. Wasteful – not all users make same mistakes
3. Insecure – too many corrections for each guess

80% of typos are left uncorrected

Top-5 correctors correct 20% of all typos

How to correct more typos?

We propose: Personalized typo-tolerance

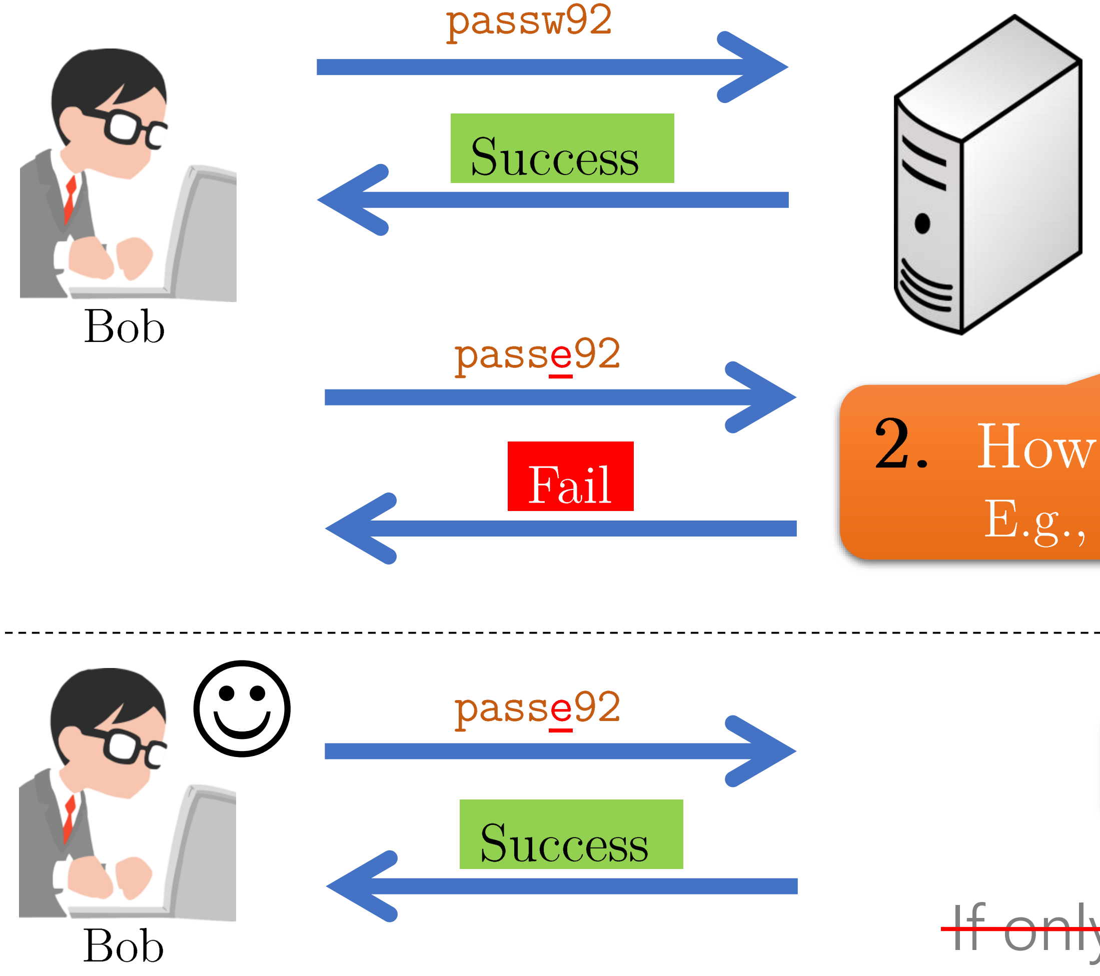
- Introduce personalized typo-tolerant password checking: allow only the typos that a user makes

- Design TypTop, a password checker that learns user's frequent typos and allows login with them. Rigorously analyze TypTop's security.

- Build a prototype for rendering computer logins typo-tolerant

  https://typtop.info

Adaptive typo-tolerance



Salted, slow cryptographic hash

$H(\texttt{passw92}) \overset{?}{=} \texttt{a5idoiaU7p...}$ ✓

$H(\texttt{pass\underline{e}92}) \overset{?}{=} \boxed{\texttt{a5idoiaU7p...}}$ ✗

2. How to figure out this is a legitimate typo?
   E.g., within edit-distance 1 from real password

Allow previously seen typos

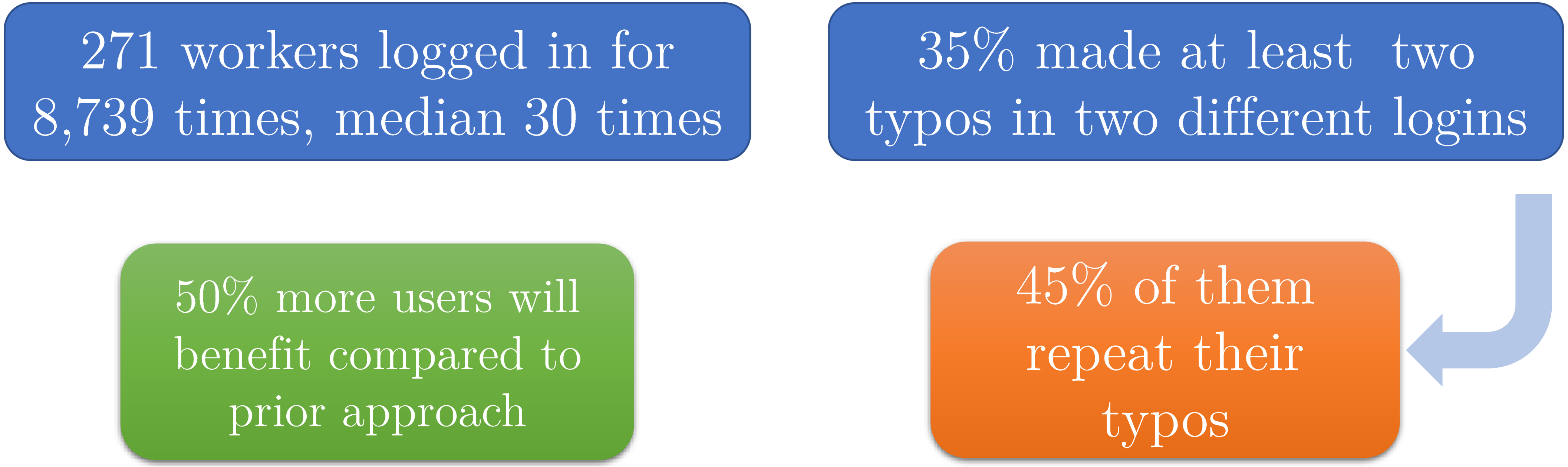
1. Do users repeat their typos?

~~If only we could store passwords in plaintext…~~

Do users repeat their typos?

Simulate password typing behavior at amazon mechanicalturk

- Asked workers
  - to register a password for an imaginary email service
  - and then, login by typing the password over multiple days



271 workers logged in for 8,739 times, median 30 times

35% made at least two typos in two different logins

45% of them repeat their typos

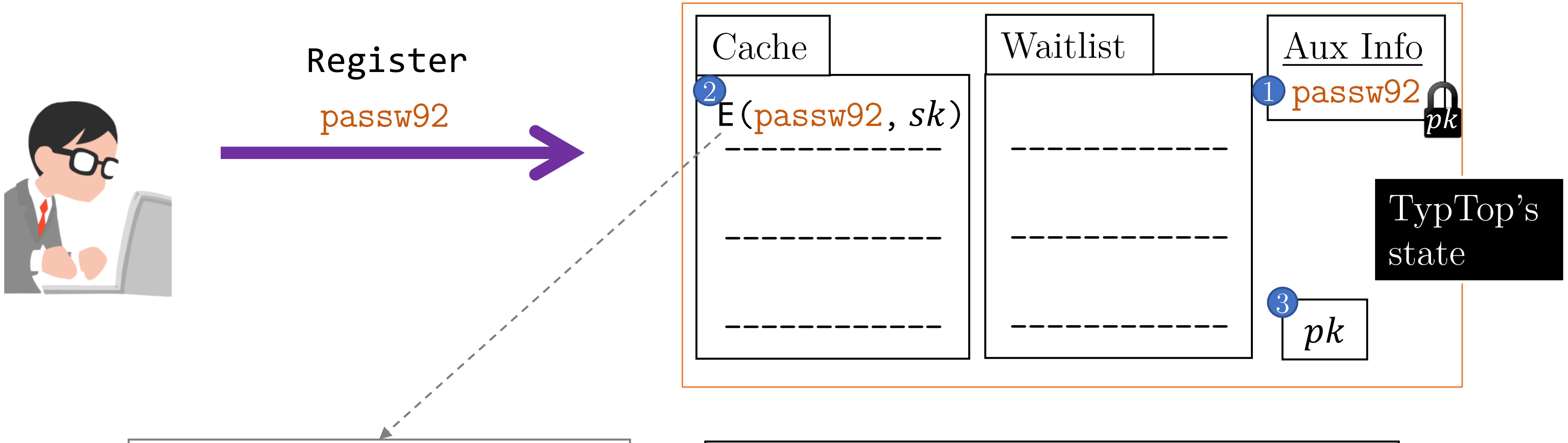
50% more users will benefit compared to prior approach

Is it legitimate?

$$\mathrm{H}(\texttt{pass}\underline{\texttt{e}}\texttt{92}) \overset{?}{=} \boxed{\texttt{a5idoiaU7p...}}$$

How to build a secure adaptive typo-tolerant password checking?

Design of TypTop : Registration



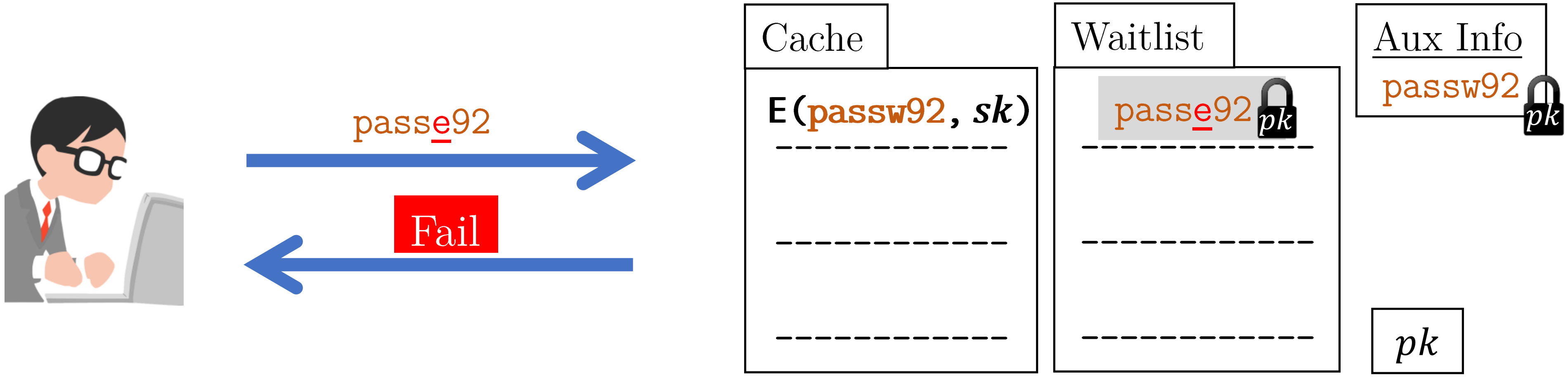
(E, D): password-based encryption scheme

```
E(passw92, sk):
    $k \leftarrow$ PBKDF(passw92)
    $C \leftarrow$ AEnc $(k, sk)$
    return $C$
```

Generates a public-key key pair: sk, pk

Slow, secure, authenticated encryption scheme

Design of TypTop : Login



passe92

Fail

Cache

E(**passw92**, ***sk***)

Waitlist

passe92 *pk*

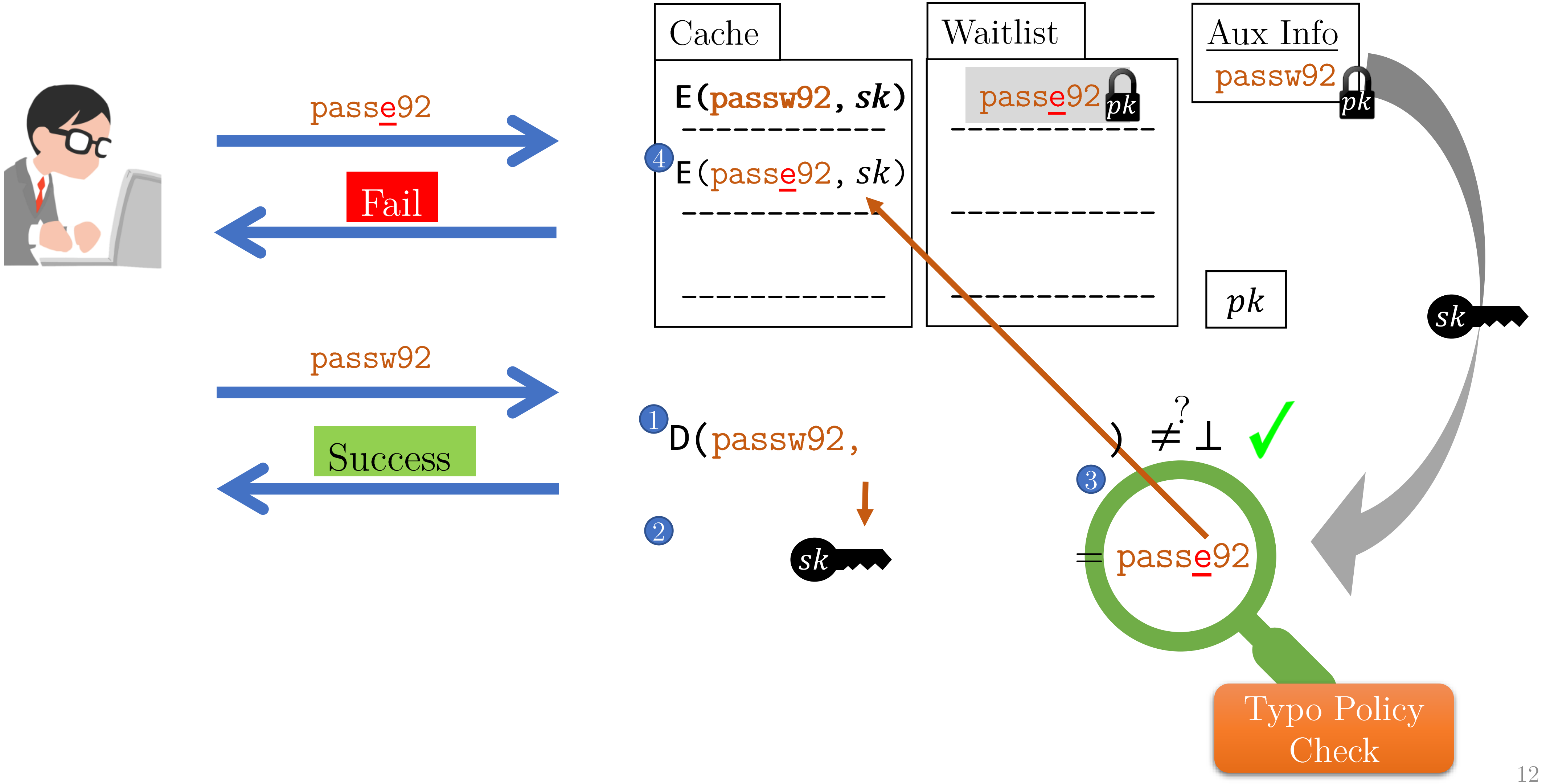
Aux Info

passw92 *pk*

pk

$$\mathrm{D}(\text{passe9}\underline{\text{e}}\text{92}, \qquad\qquad) \overset{?}{\neq} \perp$$ ✗

Design of TypTop : Login



passe92

Fail

passw92

Success

Cache

E(passw92, sk)

4 E(passe92, sk)

Waitlist

passe92 pk

Aux Info

passw92 pk

pk

sk

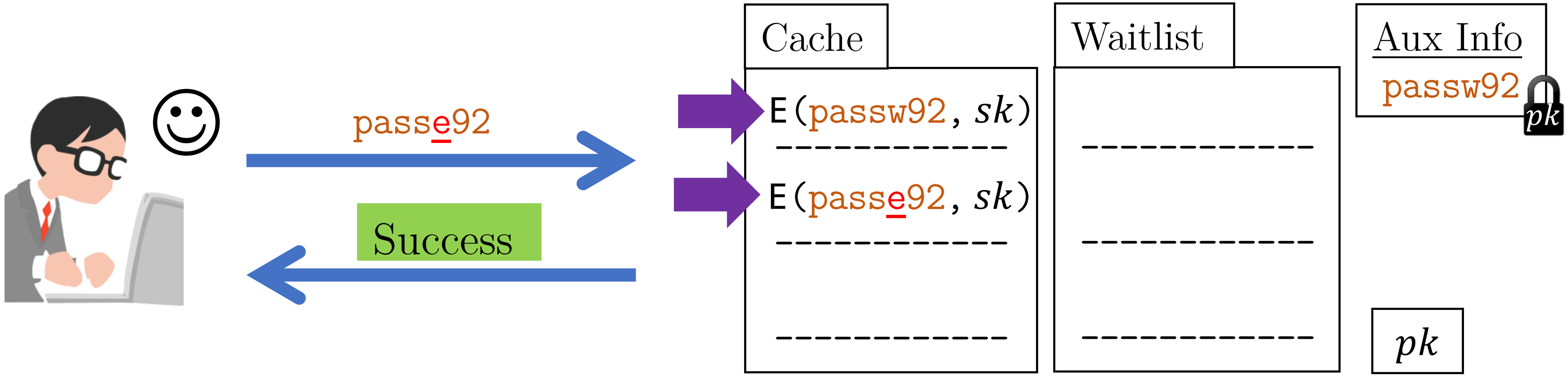
1 D(passw92, sk) $\overset{?}{\neq} \perp$ ✓

2

3 = passe92

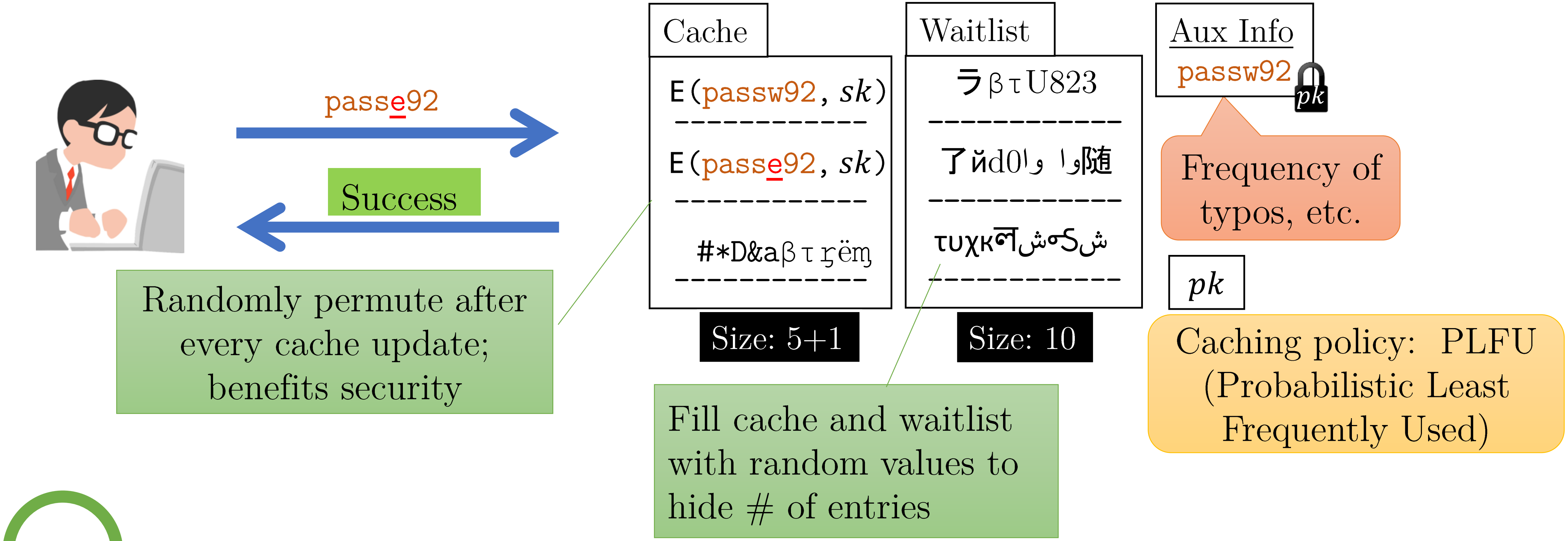
Typo Policy Check

Design of TypTop : Login with a typo



Adaptive typo-tolerant password checking without storing the password or any typos in clear

Design of TypTop : Some more details



passe92

Success

Cache

E(passw92, *sk*)

E(passe92, *sk*)

#*D&aβτɽëɱ

Size: 5+1

Waitlist

ラβτU823

了йd0وا وا随

τυχκল شॱSش

Size: 10

Aux Info

passw92 *pk*

Frequency of typos, etc.

pk

Caching policy: PLFU (Probabilistic Least Frequently Used)

Randomly permute after every cache update; benefits security

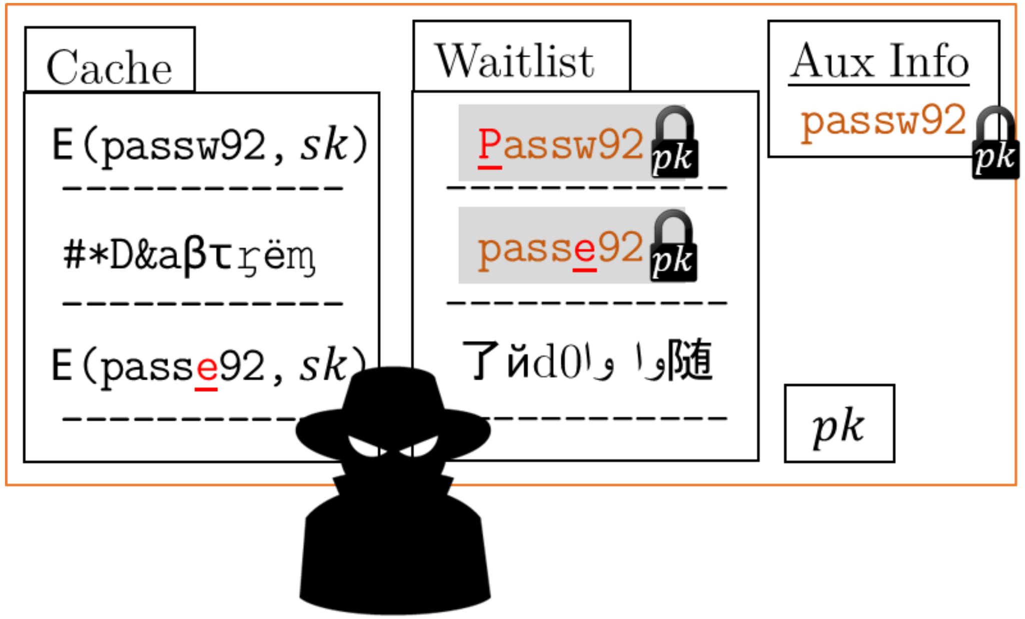
Fill cache and waitlist with random values to hide # of entries

Typo Policy Check

Check edit-distance + typo strength
- Used `zxcvbn` strength meter

What about Security?

Smash and grab attack (Offline attack)



Cache

E(passw92, sk)

#*D&aβτʐëɱ

E(passe92, sk)

Waitlist

Passw92 pk

passe92 pk

了ŭd0ا و ا随

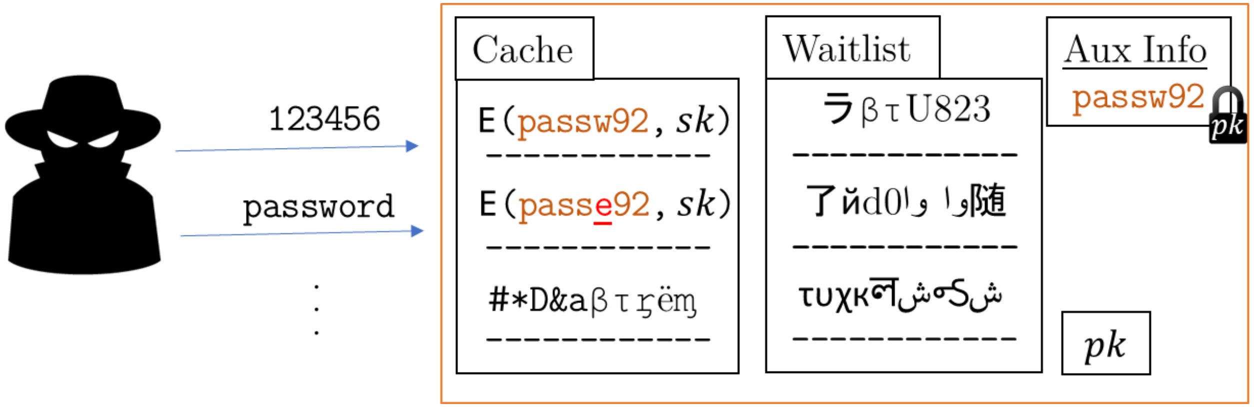
Aux Info

passw92 pk

pk

More interesting, and we detail this in the talk

Remote guessing attack (Online attack)



123456

password

⋮

Cache

E(passw92, sk)

E(passe92, sk)

#*D&aβτʐëɱ

Waitlist

ラβτU823

了ŭd0ا و ا随

τυχκलشઽش

Aux Info

passw92 pk

pk

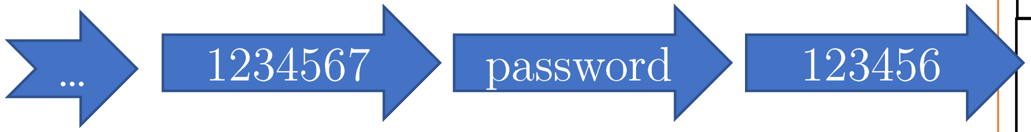
- Analysis is similar to Oakland ’16 paper
- Showed negligible security loss
- Please see paper for details

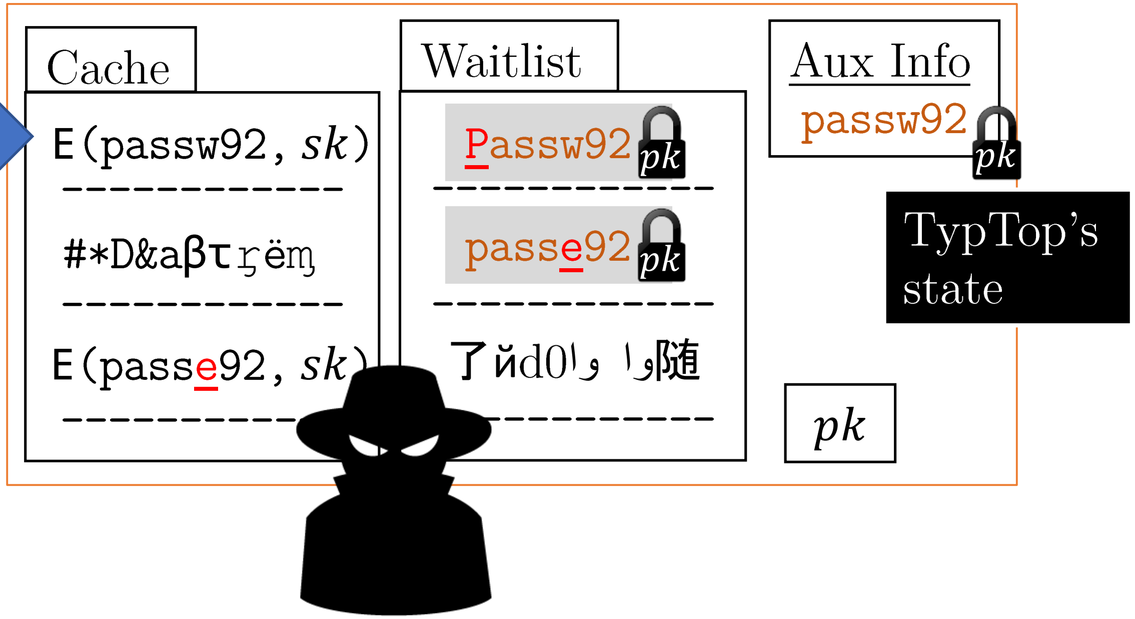
Smash and grab attack (Offline attack)

Attacker's Goal
Learn the registered password

Obvious Strategy
Brute-force guess the password

just like attacking traditional password checkers

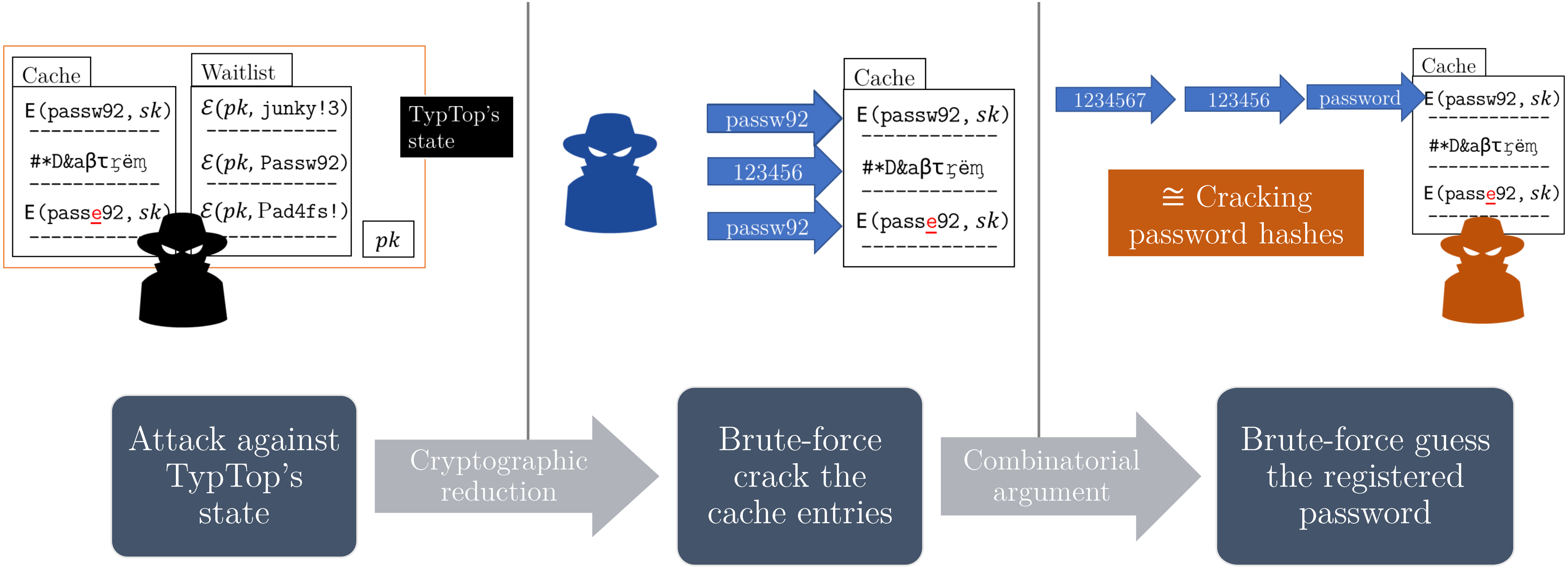




Can the attacker do better?

No!

Obvious strategy is the best an attacker can do



Cache

E(passw92, sk)

#*D&aβτɽëɱ

E(passe92, sk)

Waitlist

$\mathcal{E}(pk,$ junky!3)

$\mathcal{E}(pk,$ Passw92)

$\mathcal{E}(pk,$ Pad4fs!)

pk

TypTop's state

passw92

123456

passw92

Cache

E(passw92, sk)

#*D&aβτɽëɱ

E(passe92, sk)

1234567

123456

password

Cache

E(passw92, sk)

#*D&aβτɽëɱ

E(passe92, sk)

≅ Cracking password hashes

Attack against TypTop's state

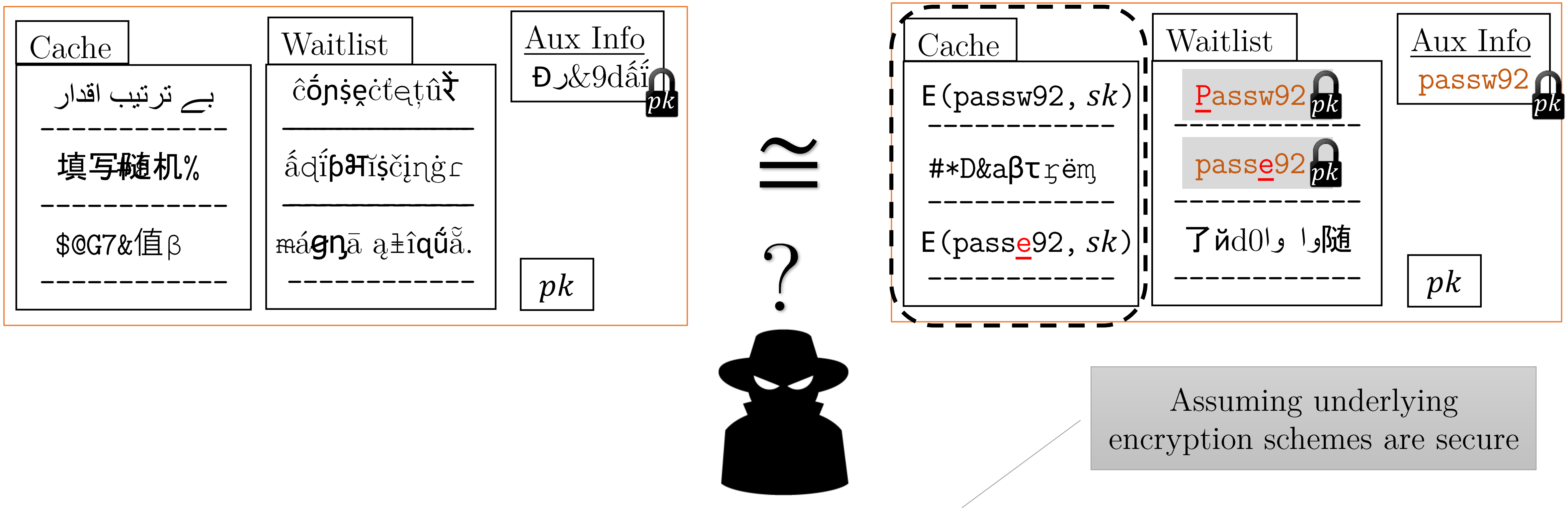
Cryptographic reduction

Brute-force crack the cache entries

Combinatorial argument

Brute-force guess the registered password

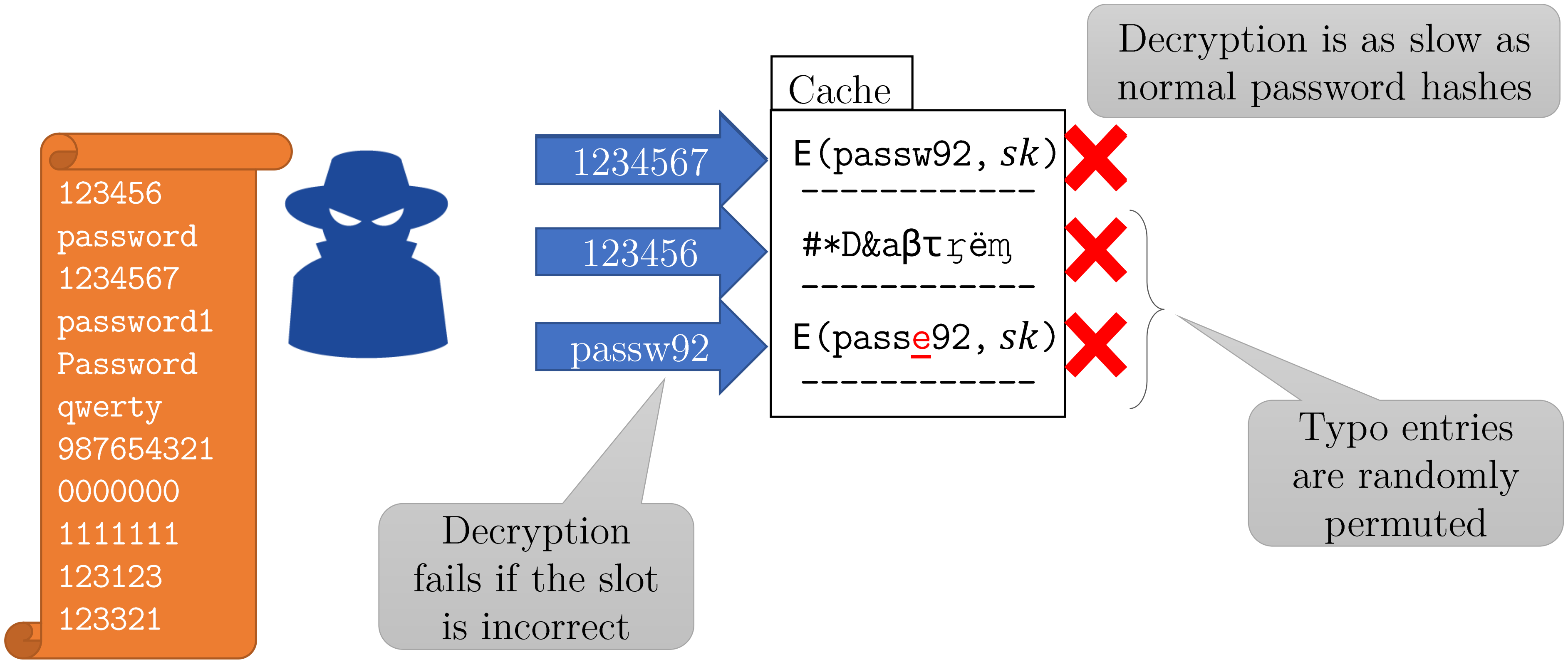
TypTop’s state appears random



Assuming underlying encryption schemes are secure

⇒ Attacker learns nothing unless he can guess an entry in the cache

Guessing against the cache entries



123456
password
1234567
password1
Password
qwerty
987654321
0000000
1111111
123123
123321

1234567

123456

passw92

Cache

E(passw92, *sk*)

#*D&aβτɼëɱ

E(passe92, *sk*)

Decryption is as slow as normal password hashes

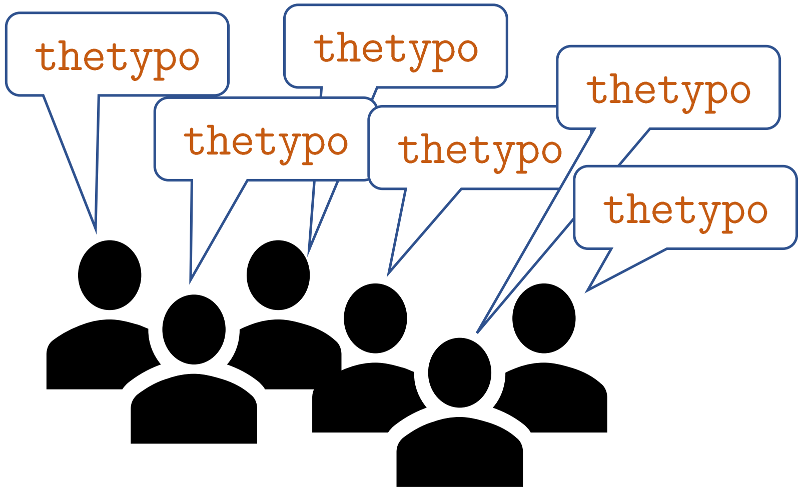
Typo entries are randomly permuted

Decryption fails if the slot is incorrect

Can attacker ever get higher advantage by trying to decrypt a typo entry in the cache?

Guessing typo is beneficial if...

...there is a typo that is always in the cache, the attacker can break TypTop by guessing that typo against all slots.



That scenario is quite unnatural

t-Sparse

t-sparse: if no typo is frequently in the cache of many passwords

$$\boxed{\forall \tilde{w},} \quad \boxed{\sum_{w} \tilde{\tau}_w(\tilde{w})} \le \boxed{t}$$

w : Password
$\tilde{w}$: Typo
t : # of typos allowed in cache
$\tilde{\tau}_w$: Cache inclusion probability

Cache inclusion probability ($\tilde{\tau}_w$)

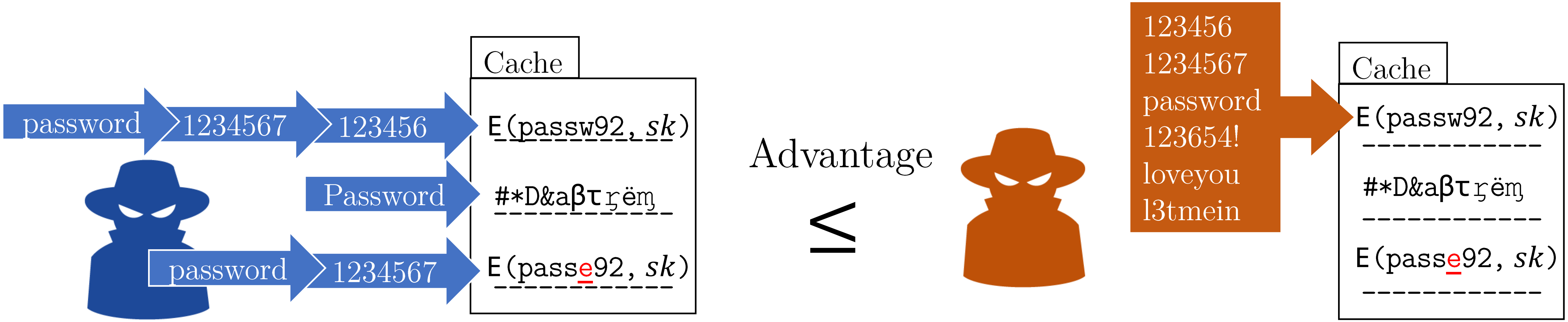
$\tilde{\tau}_w(\tilde{w}) =$ Pr[$\tilde{w}$ in cache | w],
Depends on the typo-distribution, and TypTop's caching policy

t-Sparse ⇒ TypTop ≡ Normal Pw checker

Theorem

If typo-distribution is t-sparse under TypTop's caching policy, then best attack is to brute-force guess the registered password.

Guessing typo is sub-optimal if t-sparse



password 1234567 123456

Password

password 1234567

Cache

E(passw92, sk)

#*D&aβτŗëm̧

E(passse92, sk)

Advantage

$\leq$

123456
1234567
password
123654!
loveyou
l3tmein

Cache

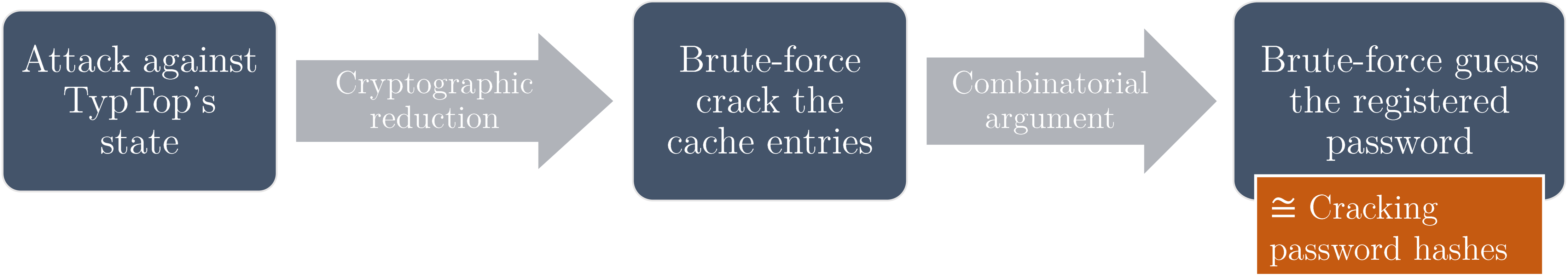
E(passw92, sk)

#*D&aβτŗëm̧

E(passse92, sk)

Every guess against a typo can be replaced by a guess against the real password that provides equal or more probability of success

Empirically verified that real world typo-distributions are t-sparse for the configurations we considered for TypTop



Attack against TypTop's state

Cryptographic reduction

Brute-force crack the cache entries

Combinatorial argument

Brute-force guess the registered password

≅ Cracking password hashes

Attacking TypTop is no easier than attacking traditional password checkers

TypTop is secure against online and offline attacks, and it improves utility.

Let’s build one!

TypTop: a smart password checker for Unix

- Created a password authentication module (PAM)
- Renders computer logins typo-tolerant
- Added a logging module
  - To collect anonymous statistics about typos for our study
  - Users can disable logging, and still keep using TypTop

***___

TypTop

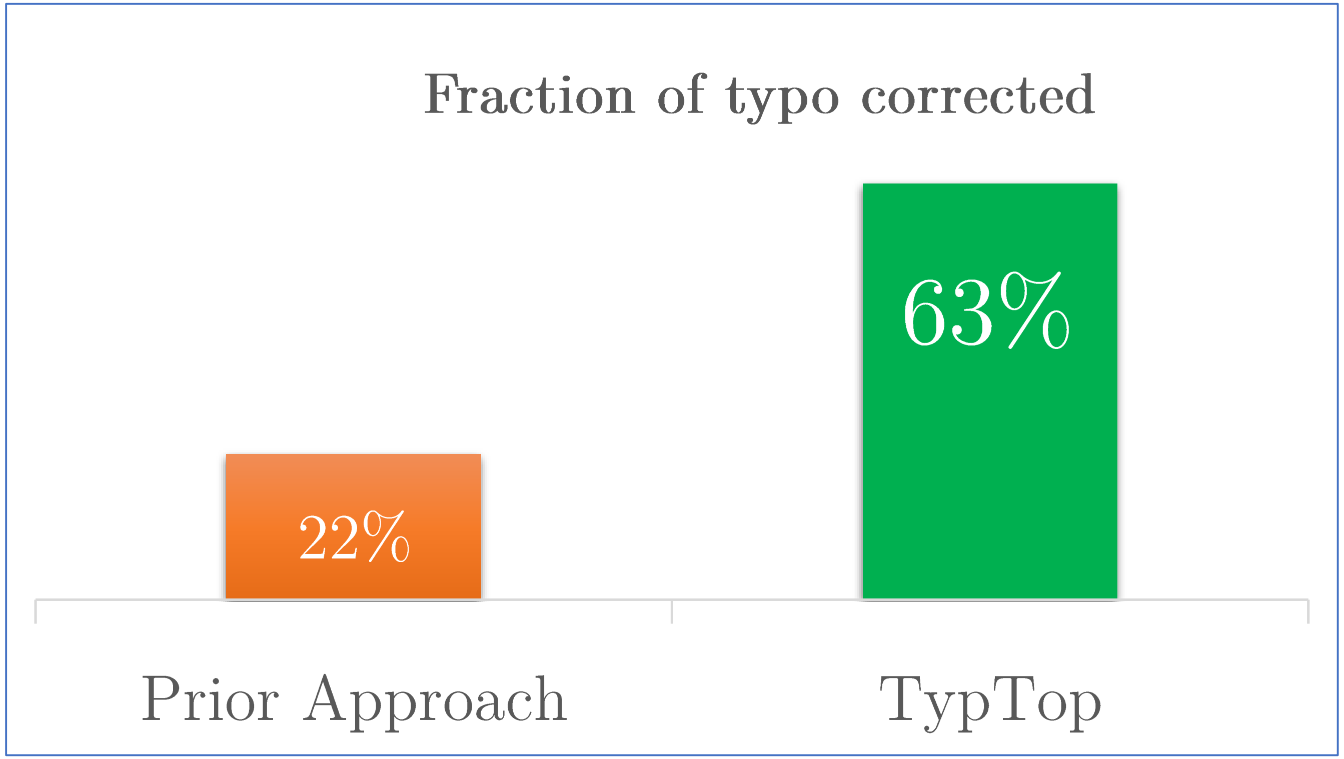
A smart password checker that lets you make mistakes

`https://typtop.info`

TypTop pilot deployment study

- Installed TypTop in 24 volunteers' laptops
  - 5 on Linux platform, 19 on MAC
  - for median 27 days.
  - Total typos observed: 501

TypTop provides
3x
improvement over prior approach



TypTop in one slide

Thanks!

- Designed TypTop, a secure personalized typo-tolerant password checking system, that adapts to user's mistakes
- Rigorously analyzed its security
- You can try TypTop now! Visit `https://typtop.info`

Typo-tolerant password checking might encourage users to adopt better security practices