Knock Yourself Out

Secure Authentication with Short Re-Usable Passwords

Benjamin Güldenring

Joint work with Volker Roth and Lars Ries

Freie Universität Berlin, Germany
Knock Yourself Out (KYO)...

- Is neither a password manager, nor a password generator, but something of both
- Allows short passwords and password re-use
- Protects against
  - password manager loss
  - multiple, simultaneous disclosure of server databases
  - computationally unbounded adversaries
Authentication - Acceptable Risk

- What is an “acceptable (individual) risk”?
- Look at ATM cards: 4 digits (0-9), three attempts allowed
- → Probability to guess PIN correctly is

\[ \Pr[\text{guess PIN}] = 3 \cdot 10^{-4} = 0.0003. \]

- To break the scheme, attacker needs to steal ATM card (first factor), and guess the correct PIN (second factor)

\[ \Pr[\text{break ATM scheme | stolen card}] = \Pr[\text{guess PIN}] \]
Authentication - Security and Safety

Alice uses her PW $p$ and PW manager / -generator to create a secret $\mathcal{A}(\text{Bob}, p)$

- **Security Threat**: Adversary finds $p$ or predicts $\mathcal{A}(\text{Bob}, p)$
- **Safety Threat**: Bob blocks Alice due to a wrong secret
Authentication - Security Threats

Alice

Password Manager / Password Generator Algorithm \( A \)

Bob

Carol

Dave

Adversary might learn:

- up to \( N \) out of Bob, Carol or Dave: e.g. (virtual) servers
- either \( \text{PW manager:} \{ \text{stolen, lost} \} \{ \text{computer, phone} \} \)
- or password \( p \) (e.g. shoulder surfing)
Authentication - Security Threats

Alice

Password Manager / Password Generator Algorithm $\mathcal{A}$

$p$

Bob

Carol

Dave

$\mathcal{A}(Bob, p)$

$\mathcal{A}(Carol, p)$

$\mathcal{A}(Dave, p)$

Adversary might learn:

- up to $N$ out of Bob, Carol or Dave: e.g. (virtual) servers
Authentication - Security Threats

Alice

Password Manager / Password Generator Algorithm $\mathcal{A}$

Adversary might learn:

- up to $N$ out of Bob, Carol or Dave: e.g. (virtual) servers
- either PW manager: {stolen, lost} {computer, phone}
Authentication - Security Threats

Alice

Password Manager / Password Generator
Algorithm $\mathcal{A}$

$\mathcal{A}(Bob, p)$
Bob

$\mathcal{A}(Carol, p)$
Carol

$\mathcal{A}(Dave, p)$
Dave

Adversary might learn:

- up to $N$ out of Bob, Carol or Dave: e.g. (virtual) servers
- **either** PW manager: \{stolen, lost\} \{computer, phone\}
- **or** password $p$ (e.g. shoulder surfing)
Authentication - Security Threats: Guessing

- Mallory tries to guess Alice’s PW, repeatedly.
- To limit Mallory’s tries, Bob blocks Alice’s account once a critical limit of failed attempts is reached (e.g. three)
Authentication - Safety Threat: Input Errors

- Did Alice mistype her PW? Allowing Alice to retry is a **safety mechanism**
- Does Mallory know the PW? Limiting Mallory’s tries is a **security mechanism**.
KYO: safety check
KYO: Input Errors

- KYO catches input errors client-side
- Bob blocks Alice’s account *immediately*, once Mallory shows a wrong password
KYO - Safety Check

- Generic safety check: For some $H$, is $H(p) = c$?
- Q1: How “good” is the safety check?
- Q2: What does an adversary learn through $H, c$?

- (Token $t$ prevents DOS attacks: see paper for details.)
Q1: How good is the safety check?

- Measure the probability that safety checks fail, assuming a wrong password $P$ was entered:

$$\Pr[H(P) = c \mid P \neq p]$$

- Unknown: types of errors a user might make
- ($\rightarrow$: users may need a custom solution)

- Idea: if $H$ is a randomly selected function, the probability is the same for every distribution of $P$
Q2: Adversary learning $H, c$

- For a randomly chosen function $H : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$, $|H^{-1}(c)|$ is binomial distributed with average value $2^{n-\ell}$.

```
\begin{align*}
|H^{-1}(c)| &= 2^{n-\ell} \\
H(p) &= c
\end{align*}
```

Conceptually similar to “collisionful hash functions”, PolyPassHash, Kamouflage, Honeywords
Q2: Adversary learning $H, c$

- For KYO security: Make $|H^{-1}(c)|$ large enough

\[ |H^{-1}(c)| = 2^{n-\ell} \]

\[
\begin{align*}
\{0,1\}^n &\quad \rightarrow \quad \{0,1\}^\ell \\
H(p) = c &\quad \text{(for some } p) \\
\text{Pr[guess } p \mid \text{stolen KYO]} &\leq \text{Pr[guess PIN]} 
\end{align*}
\]
Q1: How good is the safety check?

- For KYO safety: Make $|H^{-1}(c)|$ small enough

$$|H^{-1}(c)| = 2^{n-\ell}$$

Pr[KYO check fails | input error] $\leq$ Pr[guess PIN]
KYO: re-using and managing passwords
KYO - re-using passwords

- Randomly choose functions $F_1$ and $F_2$
- Secrets: $s_1 = F_1(p)$ and $s_2 = F_2(p)$
- What does an adversary learn about $p$ and $s_1$, given $H, c, F_1, F_2, s_2$?
KYO - re-using passwords

- Set $M := H^{-1}(c) \cap F_2^{-1}(s_2)$

- For randomly selected $H, F_2 : \{0, 1\}^n \to \{0, 1\}^\ell$, the size of $M$ is binomial distributed with average value $2^{n-2\cdot \ell}$.

- $F_1(M)$ is a bit smaller
Given $p, s$, it is easy to select a $F : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ with $n > \ell$ randomly, so that

$$F(p) = s.$$
Renew Alice’s password $p_1$: 
KYO - managing passwords

Renew Alice’s password $p_1$:

- choose a new $p_2$
- select $F_3, F_4$ with $F_3(p_2) = s_1$ (Bob), $F_4(p_2) = s_2$ (Carol)
Different password for Carol:
Different password for Carol:

- choose a new $p_3$
- choose $H_2$, set $c_2 := H(p_3)$
- select $F_5$
To merge passwords:

- dispose of $H_2, c_2$
- select $F_6$
KYO: evaluation results
Theoretical results

- minimum password length for baseline risk $3 \cdot 10^{-4}$.
- 4 ASCII (5 alphanumeric) chars withstand KYO loss.
- Each server breach costs about 2 characters ($\sim 10$ bit)
Theoretical results

- $|p_2| = 7 \text{ ch}$

- What the average user could get:
  - Florençio found 6-7 alphanum. chars average ($\sim 40 \text{ bit}$)
  - 7 alphanum. chars withstand KYO loss and 1 breach
From theory to practice

- In analysis: functions are chosen uniformly at random
- But: descriptions of $H, F_i$ too large to store in practice:

$$13 \cdot 2^{38} \text{ bit } \sim 200 \text{ gigabytes each}$$

- Use decent hash functions (But: neither collision-resistance nor pseudorandomness required)

- (One would usually just assume $H, F$ output “random” values. However, it is better to assume $H, F$ are taken from a random subset of all functions instead)

- For details: talk to me afterwards
Implementation and preliminary results

- 2-Univ: $F_\sigma(p) = (a(\sigma) \cdot p + b(\sigma) \mod p) \mod 2^\ell$
- E.g. 30 bit password, three 6-bit secrets:
  - Avg candidate probability: $0.016 \pm 0.011$ (0.015 pred).
  - Best candidate probability: $0.019 \pm 0.005$ (0.015 pred).
Outlook

- Interested in easy-to-invert hash functions
- Pen & paper KYO?
(Thank you)

Acknowledgements: The first author and the third author were supported by the CONFINE project while doing this research.