Honywords

Making Password Cracking Detectable

Presented By

Mona Al-Shahrani

Supervised By

Dr. Esam Al-Wagait

April, 2015

Password Hashing

1. P = Alice’s password
2. System stores mapping “Alice” –> h(P) in database, for a suitable hash function h.
3. When user login (perhaps Alice), system computes h(P’) and compares it to h(P).
4. If equal, login is allowed.
5. Hash function h should be easy to compute, hard to invert.

Motivation of Using Honeywords

Real passwords are often weak and easily guessed.

Motivation: ***Theft of Password Hash Files***

1. Adversary compromises system
2. Steals password hashes
3. Adversary cracks hash
   - By finding password that corresponds to stored hash value by brute-force search.
4. Finding passwords
5. Adversary succeeds in impersonating legitimate user and login.
   - It is often undetected.

Examples

6 million hashed user passwords stolen from LinkedIn in 2012

LinkedIn®

Hashed passwords of Evernote 50 million users stolen in 2013

In Mar 2015, Twitch accounts, but we did not exactly how many accounts were compromised.

twitch

Common Defense Approaches

Make password hashing more complex and time-consuming

- “Salting”: an additional value is appended to the password before it is hashed.
- Hashing passwords using strong hashing algorithms e.g. SHA2 .
- Encrypt the password hashes using a strong encryption algorithm such as AES.

Using this approach slows down authentication process for legitimate users.

Set up fake user accounts (“honeypot accounts”)

- Trap set to detect unauthorized use of information systems.
- This approach does not detect attack on legitimate user accounts.

Honeypots

A honeypot is a computer system that is expressly set up to attract and trap hackers.

In computer security, the term ***honey*** is often favored to denote decoys.

For example deploying servers to lure attackers for observation.

There are many varieties of honey system

- Honeyfiles
- Honeytokens
- **Honeywords**

After detecting the breach, an appropriate action occurs, such as

- The administrator can watch the hacker exploit the vulnerabilities of the system
  - learning where the system has weaknesses that need to be redesigned.
- The hacker can be caught and stopped while trying to obtain root access to the system.

What is Honeywords

Set multiple possible passwords for each account, only one of which is genuine. The others we refer to as “honeywords.”

The attempted use of a honeyword to log in sets off an alarm, as an adversarial attack has been reliably detected.

What is Honeywords

Honeywords proposed 2013 by

Dr. Ari Juels

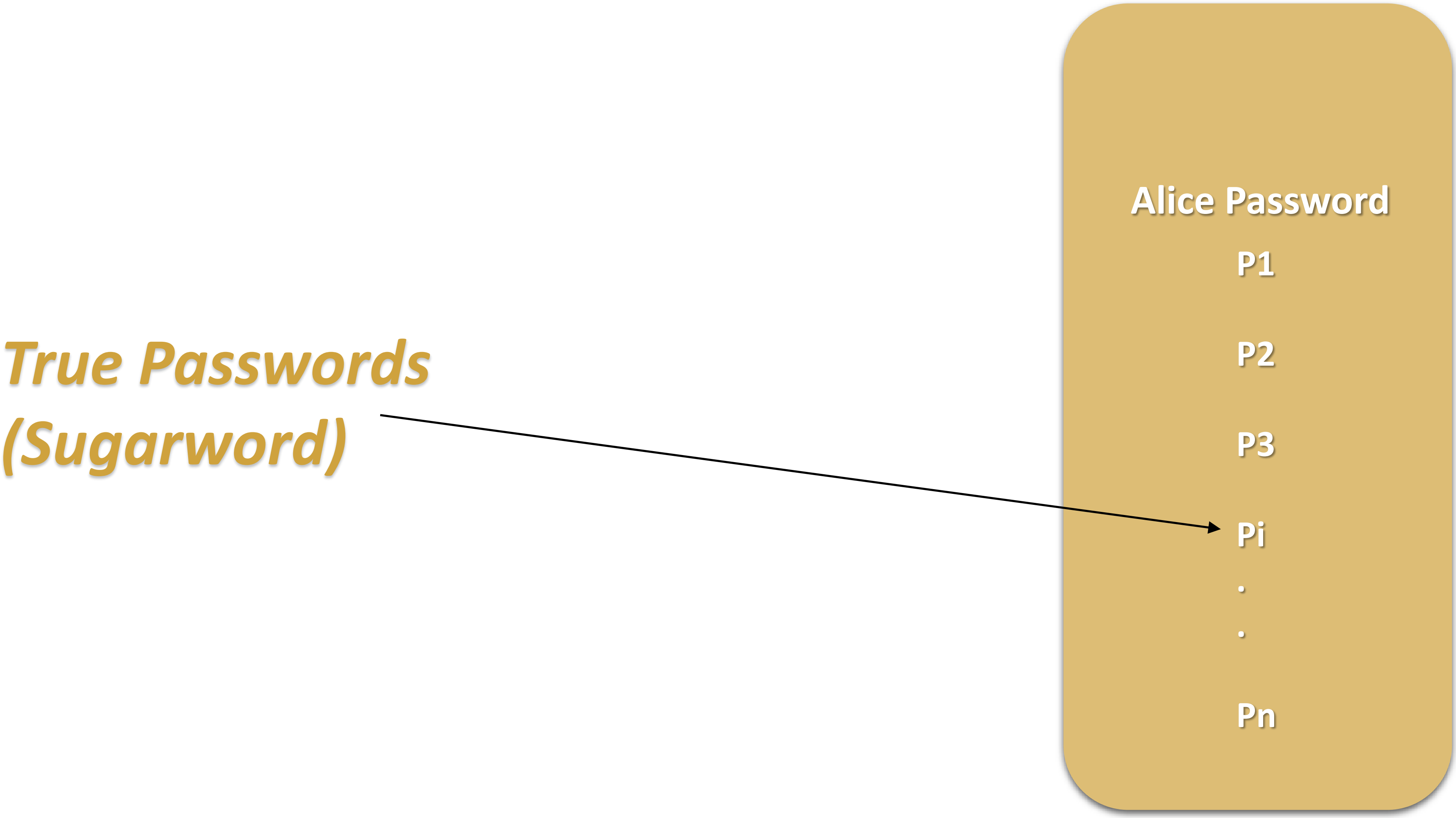
He was the Chief Scientist of RSA.

Dr. Ronald Rivest

He is Professor of Computer Science at MIT

He is one of the inventors of the RSA algorithm

Terminology



True Passwords (Sugarword)

Alice Password

P1

P2

P3

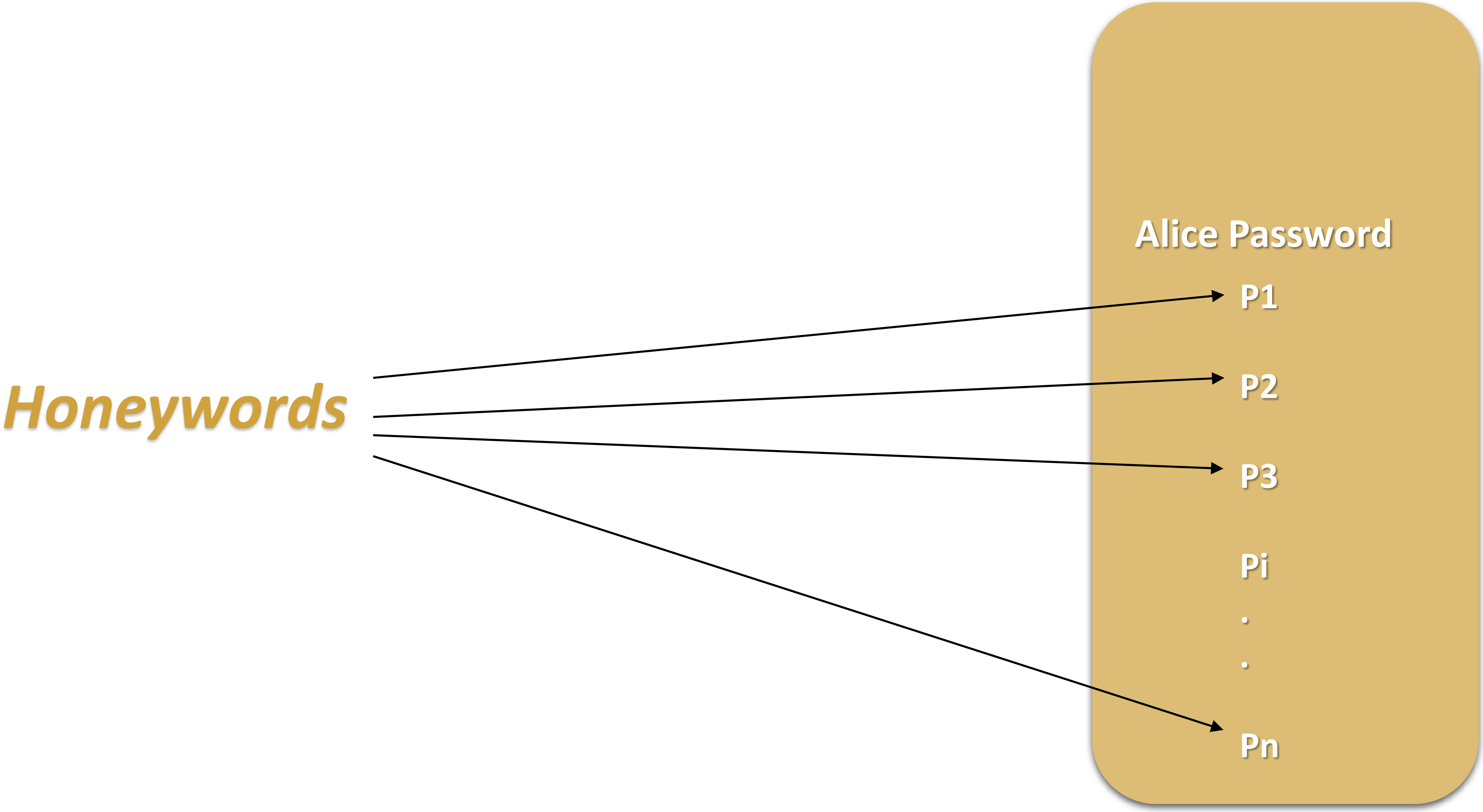
Pi

.

.

Pn

Terminology



Honeywords

Alice Password

P1

P2

P3

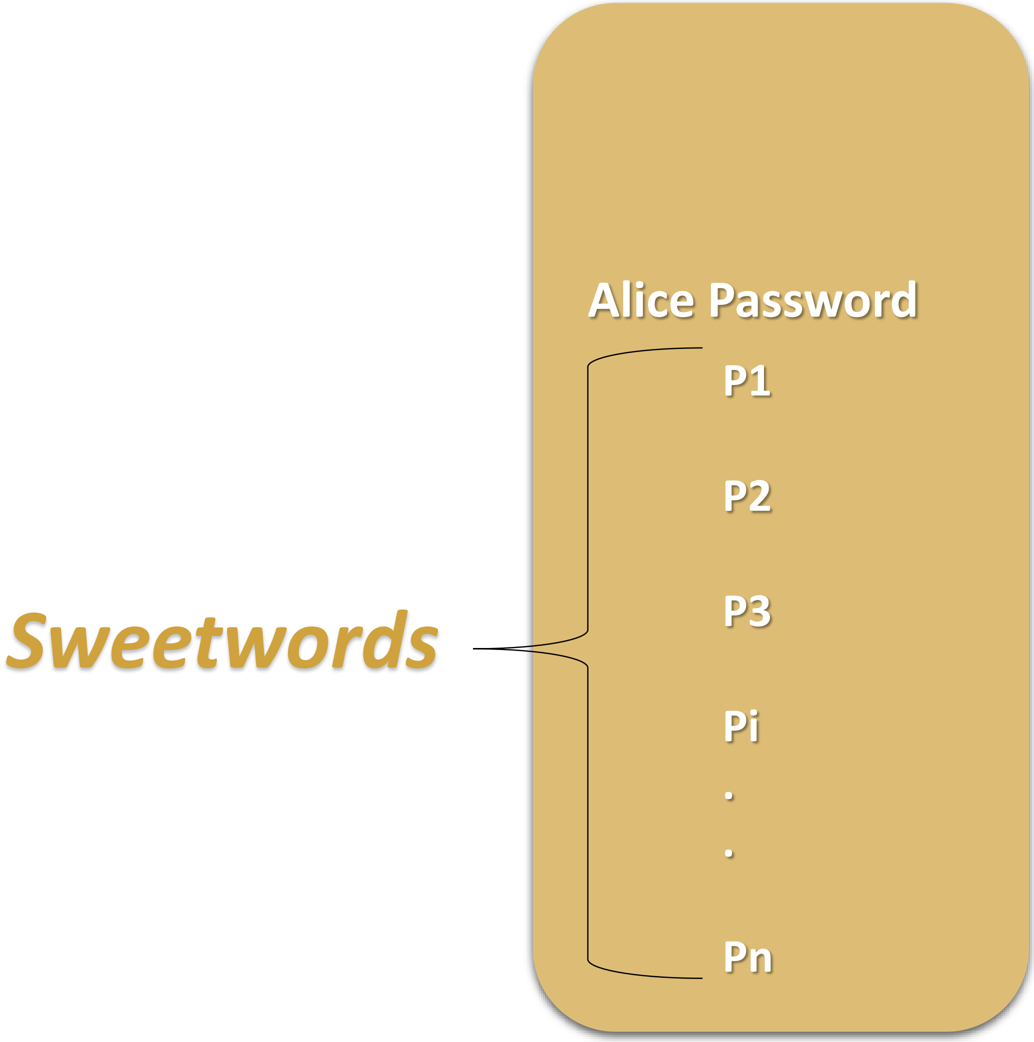
Pi

.

.

Pn

Terminology



Sweetwords

Alice Password

P1

P2

P3

Pi

.

.

Pn

How Honeywords work

There are two basic processes :

Verification

How does check the submitted password is the true password.

Generation

How to generate Honeywords?

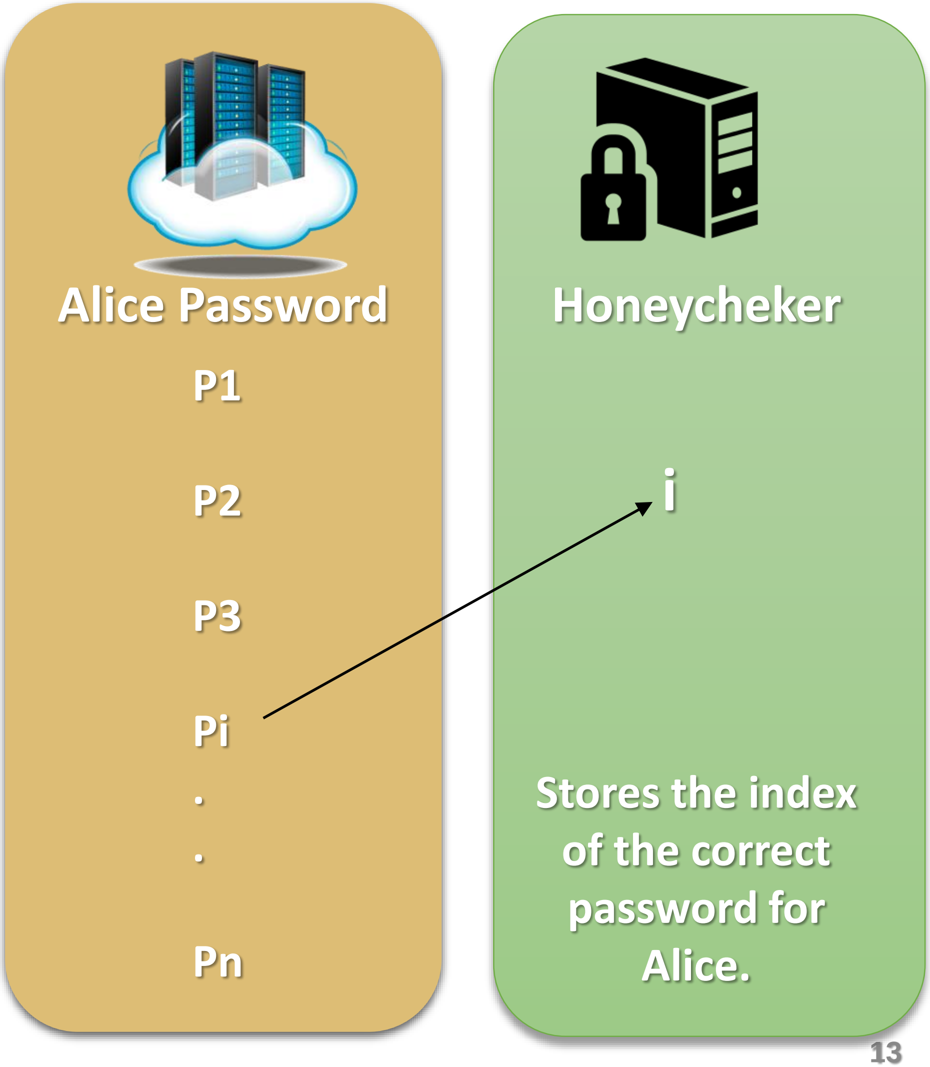
How to make realistic decoy passwords?

Verification

Honeywords and true password are placed into a list of Sweetwords, in a random order.

Exactly one of these Sweetwords is equal to the password Pi which is the true password.

Honeychecker is an auxiliary secure server that maintains a correct index of user's password.



Alice Password

P1

P2

P3

Pi

.

.

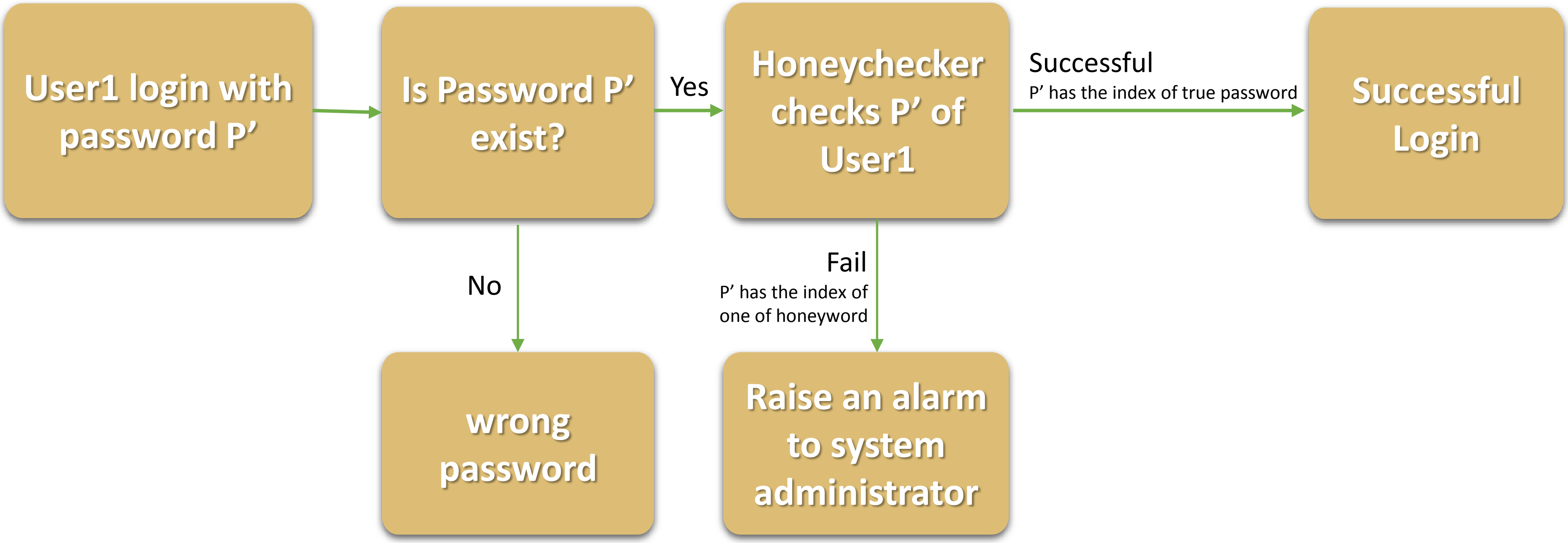
Pn

Honeycheker

i

Stores the index of the correct password for Alice.

Verification


User1 login with password P’
Is Password P’ exist?
Yes
Honeychecker checks P’ of User1
Successful
P’ has the index of true password
Successful Login
No
wrong password
Fail
P’ has the index of one of honeyword
Raise an alarm to system administrator

Honeywords Generation

It is a process to generate the honeywords beside the user password in the system.

But How to make the true password undistinguishable from generated honeywords.

If a Honeyword generation is *flat*, in the case that each one is equally likely to be chosen as the true password.

Adversary can guess the true password with probability only 1/n.

For example n=20, adversary has a chance of at most 5% of picking the correct password.

Two approaches to generate Honeywords

<u>Legacy-UI password changes</u>

- Chaffing by Tweaking
- Chaffing-with-a-Password Model

<u>Modified-UI password changes</u>

- Take-a-tail" Method

Chaffing by Tweaking

- “**Tweak**” selected character positions of the password to obtain the Honeywords

  .

- For each selected position the character of the real password is replaced by a randomly-chosen character of the same type .

Chaffing-by-tail-tweaking

Tweak last t positions of password

- **But** if it tweaks the last position of this password **57*flavors**

Honeywords : 57*flav**rbn**, 57*flav**ctz** ?

Chaffing by Tweaking

Chaffing-by-tweaking-digits

- Tweak last t positions containing digits
- Example : the desired number of positions to tweak is 4
- BG+1a745 -> BG+7a305 BG+2a177 BG+9a587 BG+0a602

Chaffing-with-a-Password Model

Modeling syntax

- The password is splitted into character sets.
- Example: **mice**3*blind* is decomposed as 4-letters + 1-digit + 5-letters

Honeywords: **gold**5*rings* **name**8*honey* **flat**7*sorts*

Simple model

- Generates Honeywords using a probabilistic model of real passwords.
- This model based on a given list of thousands/millions of passwords.
- However, attacker might have access to the list of passwords.

Take-a-tail Method

The **take-a-tail** method is identical to the **chaffing-by-tail-tweaking**.

Except that the tail of the new password is now randomly chosen by the system, and required when user is entering new password.

Example

Propose a password: *myPassword*
Append “*413*” to password.
Enter new password: *myPassword413*

Generated honeywords:
myPassword798
myPassword982
myPassword113
myPassword056
myPassword935
myPassword664

Take-a-tail Method

Advantage

The password tail is picked randomly to ensure that the password and Honeyword generation procedure is perfectly **flat.**

Disadvantage

Users need to remember the random numbers that is appended to their passwords.

Hybrid Generation Methods

- Combining more than one honeyword generation methods.
- For example chaffing-with-a-password-model and chaffing-by-tweaking-digits.

Example

The correct password is **Apple190**.

Chaffing-with-a-password-model with **a** = 3

Apple190 , Angel554 and Happy969

Chaffing-by-tweaking-digits where **t**=3 and **b** = 4

The total number of Sweetwords will be **k** =a x b, **k**= 12

Happy969	Angel554	Apple223
Happy669	Angle772	Apple190
Happy346	Angle485	Apple643
Happy182	Angle876	Apple254

Policy Choices

Honeyword entered – possible actions

- Setting off an alarm a system administrator
- Letting login proceed as usual
- Letting the login proceed, but on a honeypot system
- Tracing the source of the login carefully
- Shutting down that user‘s account or the computer system

Policy Choices

Failover mode

- The system is set to failover mode when honeycecker is down.
- The system will accept temporarily honeywords as the correct password.
- To prevent Denial-of-Service attacks.

Attack Scenarios

- Attacking the Honeychecker
- Denial-of-Service
- Targeted password guessing
- Multiple systems
  - Intersection attack
  - Sweetword-submission attack

Conclusion

- Defense in the security of hashed passwords
- Decreases the value of the stolen password hash files
- Makes password cracking detectable
- Published password files (e.g., one stolen from LinkedIn) provide attackers with *insight* into how users compose their passwords.
  - Attackers lose the potential to improve their future attacks.