

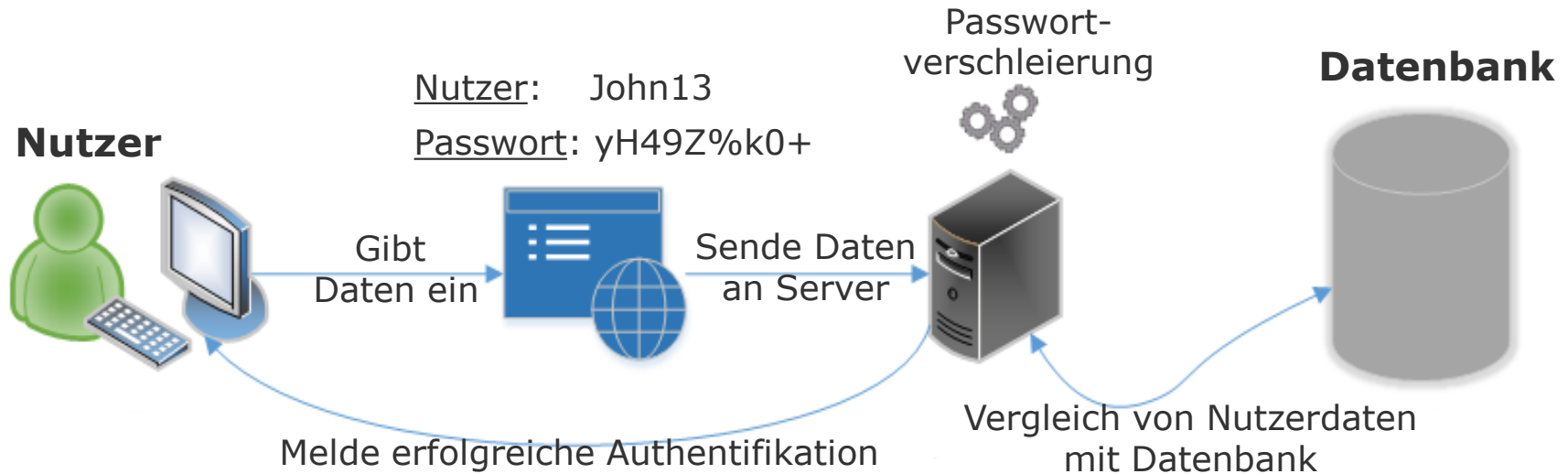
openHPI-Kurs „Digitale Identitäten“
Passwortsicherheit –
Authentifizierung mithilfe von Passwörtern

Prof. Dr. Christoph Meinel
Hasso-Plattner-Institut, Universität Potsdam

Authentifizierung mithilfe von Passwörtern (1/2)

- Person beweist, dass eine digitale Identität ihr gehört durch Eingabe des richtigen **Passwords** (Wissen)
- Onlinedienst überprüft die Korrektheit des Passwords durch Abgleich mit den bei der Registrierung erhobenen und in der Nutzerdatenbank gespeicherten Informationen
 - Bei Übereinstimmung erhält Person Zugang und die mit der digitalen Identität verbundenen **Autorisierungen**
- Die in der Nutzerdatenbank gespeicherten Passwords sollten **niemals** im Klartext gespeichert werden, sondern nur in verschleierter Form
 - Verschleierung vermittelt kryptographischer Hash-Funktion

Authentifizierung mithilfe von Passwörtern (2/2)



- Nutzer gibt Nutzernamen und Klartext-Passwort auf Webseite im Browser ein
- Webseite sendet Passwort zusammen mit dem Nutzernamen an den (Server des) Onlinedienst
- Server verschleiert (*hasht*) Passwort und vergleicht das mit verschleiertem Passwort, das in der Nutzerdatenbank für den Nutzer mit dem angegebenen Nutzernamen gespeichert ist

Schwachstelle: Klartextpasswörter

- Anbieter speichert Passwort bei Anmeldung und Einrichtung eines Nutzer-Accounts (digitaler Identität) im Klartext in der Nutzerdatenbank
- Gelingt es Cyberkriminellen, sich über das Internet Zugriff auf IT-System der Onlinedienstes zu verschaffen, dann können sie
 - ➔ Nutzerdatenbank mit allen Passwörtern herunterladen
- Wieso sind gestohlene Passwörter so gefährlich?
 - Angreifer kann sich mit gestohlenem Passwort bei dem Onlinedienst einloggen
 - Angreifer kann versuchen, **gleiche** oder **leicht abgewandelte** Nutzer-Passwort-Kombination bei anderen Dienst zu verwenden
 - Nutzer **wissen** meist nichts vom Diebstahl ihrer Identitätsdaten

Sichere Speicherung von Passwörtern

Wie können Passwörter sicher gespeichert werden?

- Speicherung der Passwörter mit kryptographischen Hash-Verfahren
 - **Hashing** - Verschleierung
 - Verschlüsselung in einem String fester Länge

- Erfahrung mit dem **HPI Identity Leak Checker**
 - viele Anbieter speichern Passwörter noch immer im **Klartext**
 - Falls Hashs verwendet werden, werden oft veraltete und relativ **leicht knackbare Verfahren**, wie z.B. MD5, verwendet

Passwortverschleierung durch Hashing (1/2)

- Passwort wird mit Hilfe einer kryptographischen Hash-Funktion verschleiert
- Hash-Funktion bildet dazu das Passwort auf einen Schlüsseltext fester Länge („Hash“) ab, z.B.

$\text{hMD5}(\text{yH49Z\%k0+}) = 3\text{cd6acc0f0fce929723014b8a9f32f91}$

Passwortverschleierung durch Hashing (2/2)

- Hash-Funktionen sind so konstruiert, dass es nur mit extrem viel Aufwand möglich ist, das ursprüngliche Passwort aus dem Hash(wert) wieder zu gewinnen (Einweg-Funktion)



- Häufige verwendete Hash-Verfahren sind MD5 und SHA1
 - **Aber:** MD5 und SHA1 gelten inzwischen als unsicher, weil sie mit der heute zur Verfügung stehenden Rechnerpower umkehrbar sind
- Als sicher gelten die Hash-Verfahren SHA2 und SHA3

Validierung eines Passworts

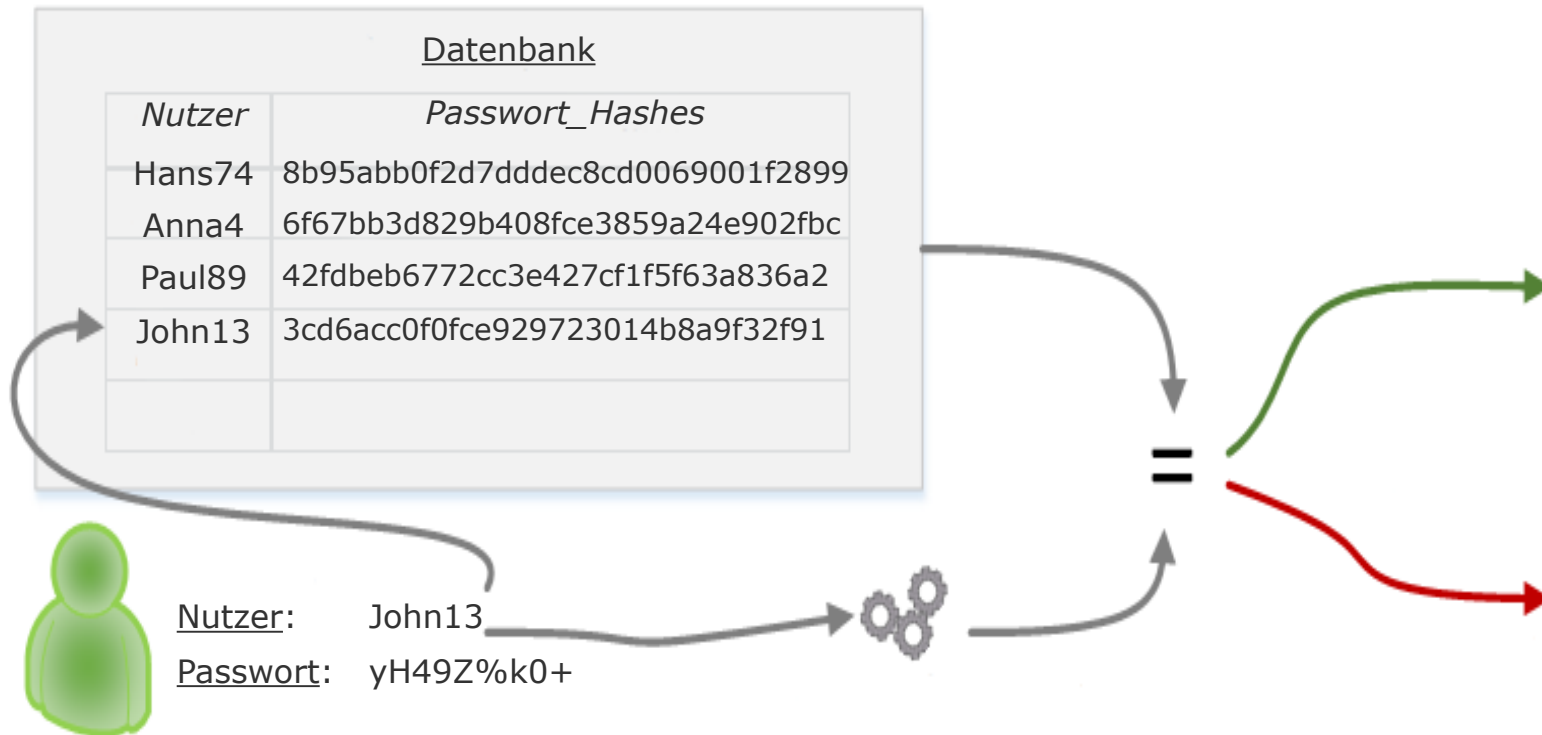


Nutzer: John13

Passwort: yH49Z%k0+

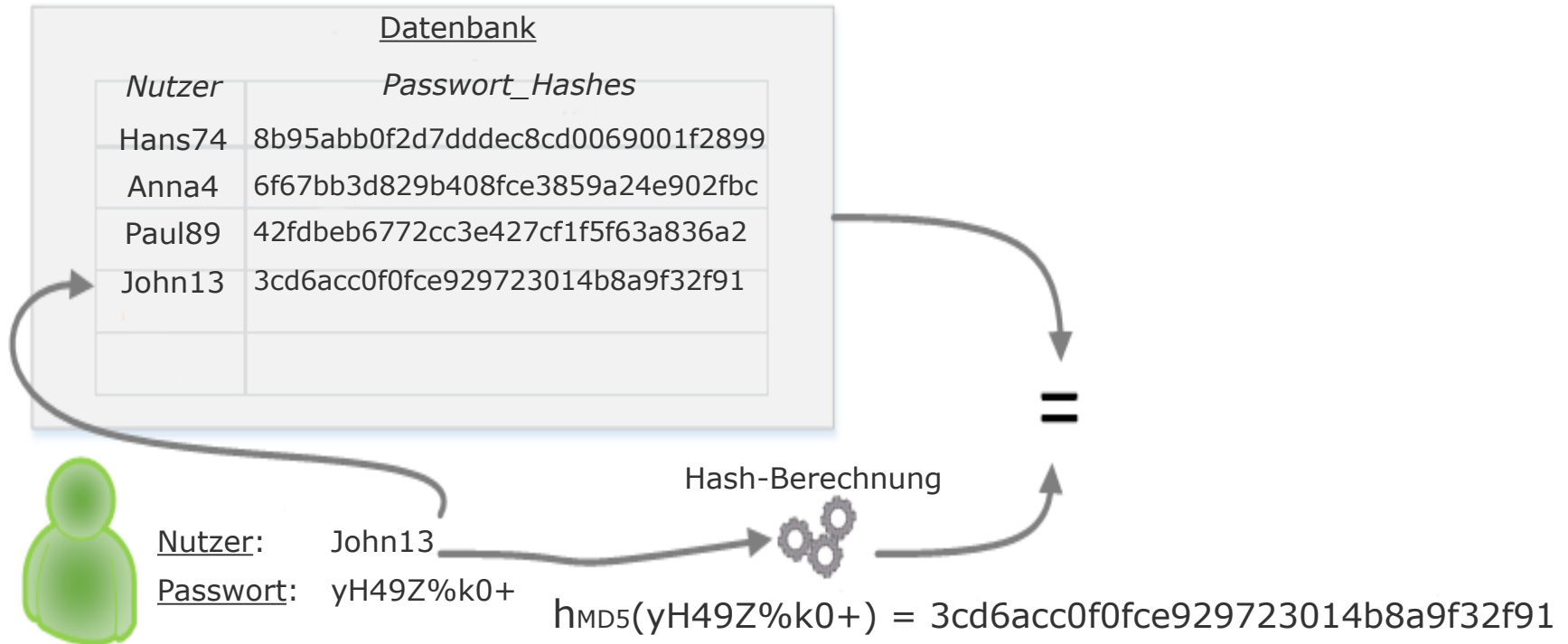
Nutzer gibt sein Klartext-Passwort ein

Validierung eines Passworts



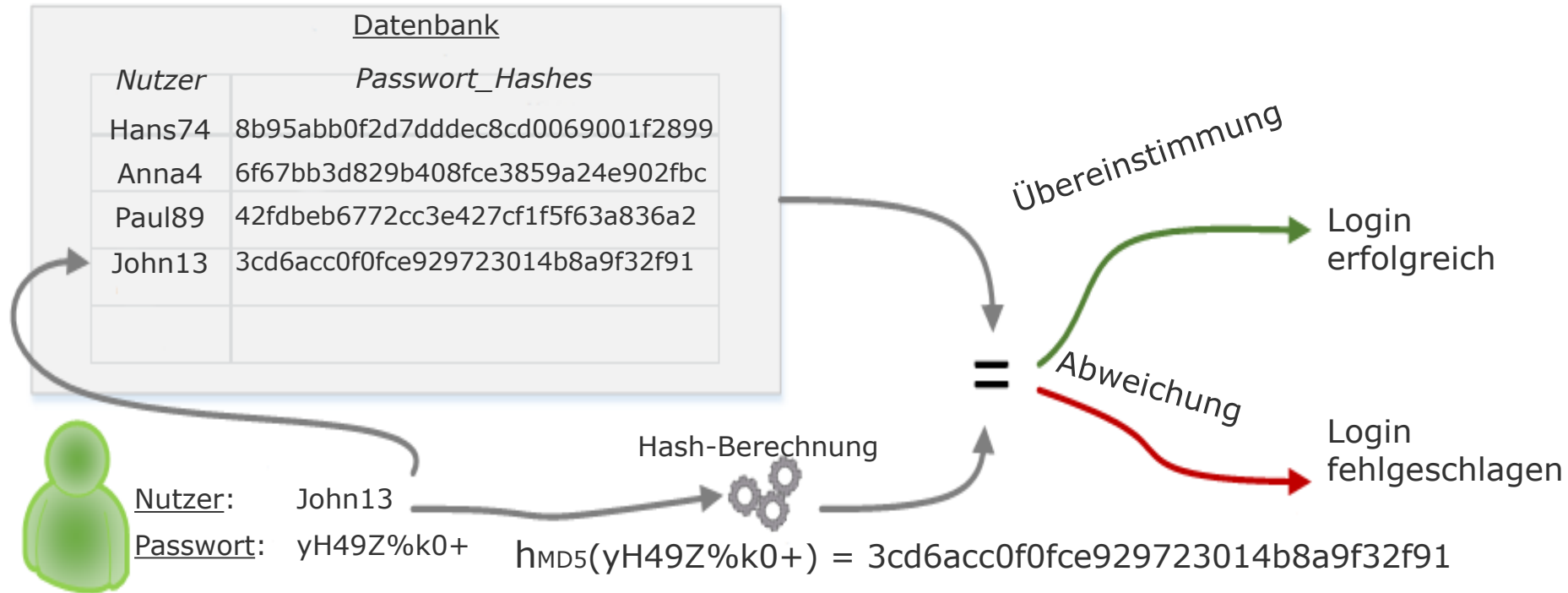
Gehashtes Passwort wird aus der Datenbank geholt

Validierung eines Passworts



Berechnung des Hashs für das eingegebene Klartext-Passwort

Validierung eines Passworts



- Vergleich beider Hashs
- Login ist erfolgreich, wenn beide Hashs übereinstimmen

Sichere Hashes mit Salt (1/2)

Passwortverschleierung durch Hashing hat weitere Schwachstellen

<i>Nutzer</i>	<i>MD5-Passwort-Hash</i>	<i>Passwort</i>
Nutzer A	3e45af4ca27ea2b03fc6183af40ea112	Passwort
Nutzer B	3e45af4ca27ea2b03fc6183af40ea112	Passwort

- Zwei Nutzer mit gleichem Passwort haben auch gleichen Hash in der Datenbank
- Kennt man das Passwort eines Nutzers, kennt man die Passwörter von allen Nutzern mit gleichem Passwort
- Hash kann erraten werden, wenn man sämtliche Wörter aus einem Wörterbuch, wie z.B. Passwort, hasht (siehe Angriffe)

Sichere Hashes mit Salt (2/2)

Lösung: Gleiches Passwort erzeugt jedes Mal unterschiedlichen Hash

- **Idee:** Bevor man den Hashwert eines Passwortes berechnet, wird dem Passwort eine zufällige Zeichenfolge (Salt / Salz) angehängt

$$h_{MD5}(\text{PASSWORT} + \text{SALT}) = \text{HASH}$$

- Dann wird Salt zusammen mit dem Passwort-Hash gespeichert

<i>Nutzer</i>	<i>MD5-Passwort-Hash</i>	<i>Salt</i>	<i>Passwort</i>
Nutzer A	ec58ef6c6f345d99054e419d19be6e3f	5<§	Passwort
Nutzer B	188623e1be9c480214838bf596b22d0d	=3(	Passwort

- Authentifizierung durch **Wissen**
 - Person beweist durch die Angabe des richtigen Passworts, dass eine digitale Identität zu ihr gehört
- Onlinedienst speichert das Passwort bei der Registrierung einer digitalen Identität und validiert das Passwort beim Login-Vorgang
 - Passwörter sollten **nie** im Klartext, sondern nur in verschleierter Form gespeichert werden (*Password Hashing*)
 - Zur Steigerung der Sicherheit sollten bei der Passwortverschleierung nur sichere Hashverfahren und Salts verwendet werden