

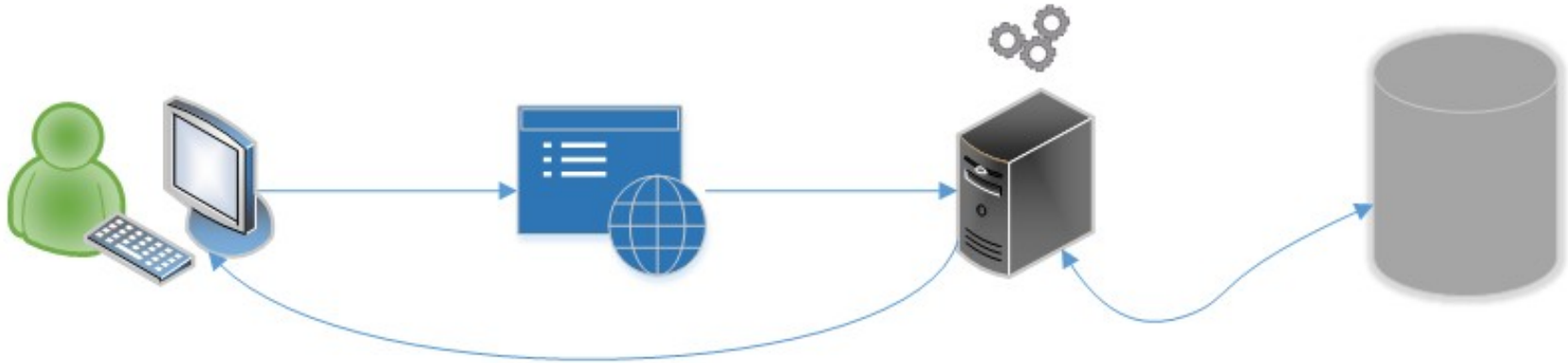
openHPI course "Digital Identities
Password security -
Password authentication

Prof. Dr. Christoph Meinel

Hasso Plattner Institute, University of Potsdam

- person proves that he owns a digital identity by entering the correct password (**knowledge**)
- Service checks the correctness of the password by comparing it with the information stored in the user database.
 - In case of a match, the person is granted access to the Digital Identity and the associated authorizations.
- Password is stored in a database during the registration of the digital identity.
 - Passwords should **never** be stored in plain text.
 - The use of concealment methods (cryptographic hash functions) is better.

Authentication using Passwords (2/2)



- User enters username and plaintext password on website
- The website sends the password together with the username to the server.
- Server *hashes* the password and compares it with the password stored in the user database for the user with the specified user name.

Vulnerability: Plain text passwords

- The provider stores the password in plain text in the user database when registering and setting up a user account (digital identity).
- If cybercriminals succeed in gaining access to this database system via the Internet, they can
 - ➔ download the user database with all passwords
- Why are stolen passwords so dangerous?
 - Attacker can log into service with password where password was stolen
 - Attacker can try to use **same** or **slightly modified** user password combination for other services
 - Users usually **know** nothing about the theft of their data

Secure storage of passwords (1/3)

How can passwords be stored securely?

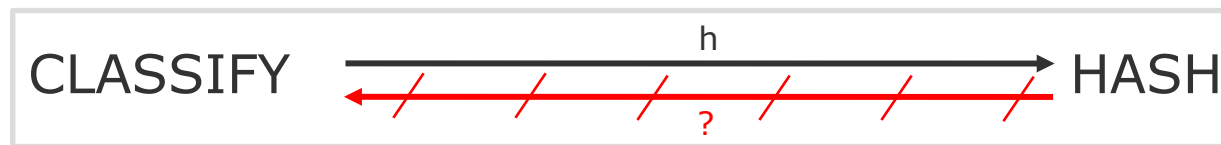
- Password storage with cryptographic methods
 - **Hashing** - Obfuscation
 - encoding

- Experience with the **HPI Identity Leak Checker**
 - Many providers still store passwords in **plain text**
 - If hashes are used, relatively **easily crackable procedures such as MD5** are often used.

Secure storage of passwords (2/3)

Password obfuscation by hashing

- Password is obfuscated using a hashing function
- Hashing function maps the password to an obfuscated text of fixed length ("Hash").



- Hashing functions are constructed in such a way that it is only possible to recover the original password from the hash (worth) with a great deal of effort (one-way function).

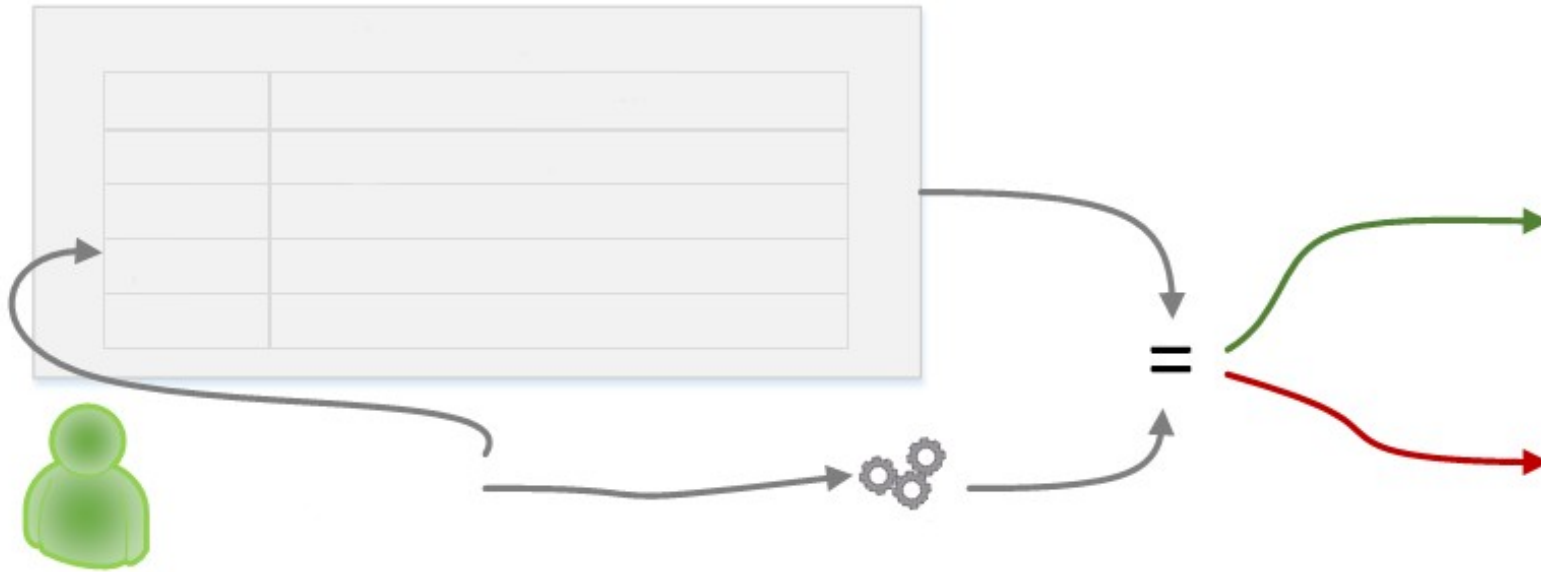
Secure storage of passwords (3/3)

Password obfuscation by hashing

- Common hashes are MD5 and SHA1.
 - But: MD5 and SHA1 are now considered unsafe because they are reversible due to the increased computer power available today.
- Therefore, rather use safe procedures such as SHA2 or SHA3
 - Use MD5 for demonstration, because the resulting hash value is shorter.

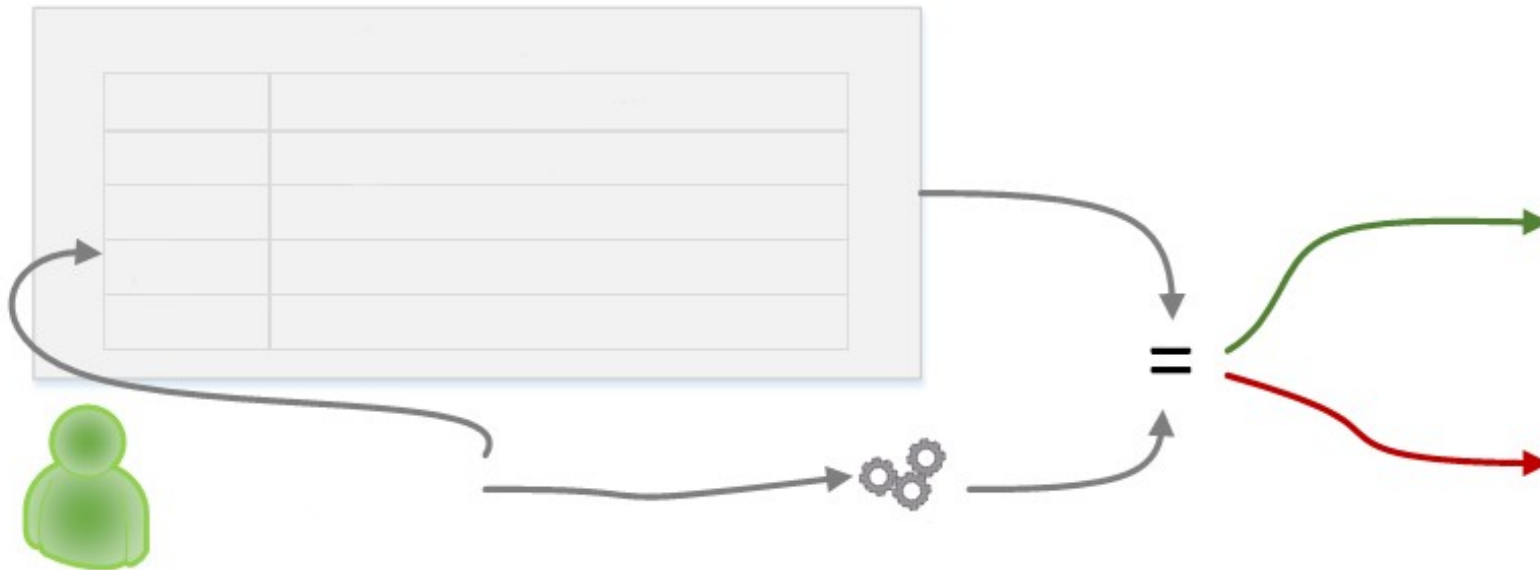
$\text{hMD5}(\text{yH49Z\%k0+}) = 3\text{cd}6\text{acc}0\text{f}0\text{fce}929723014\text{b}8\text{a}9\text{f}32\text{f}91$

Validation of a password



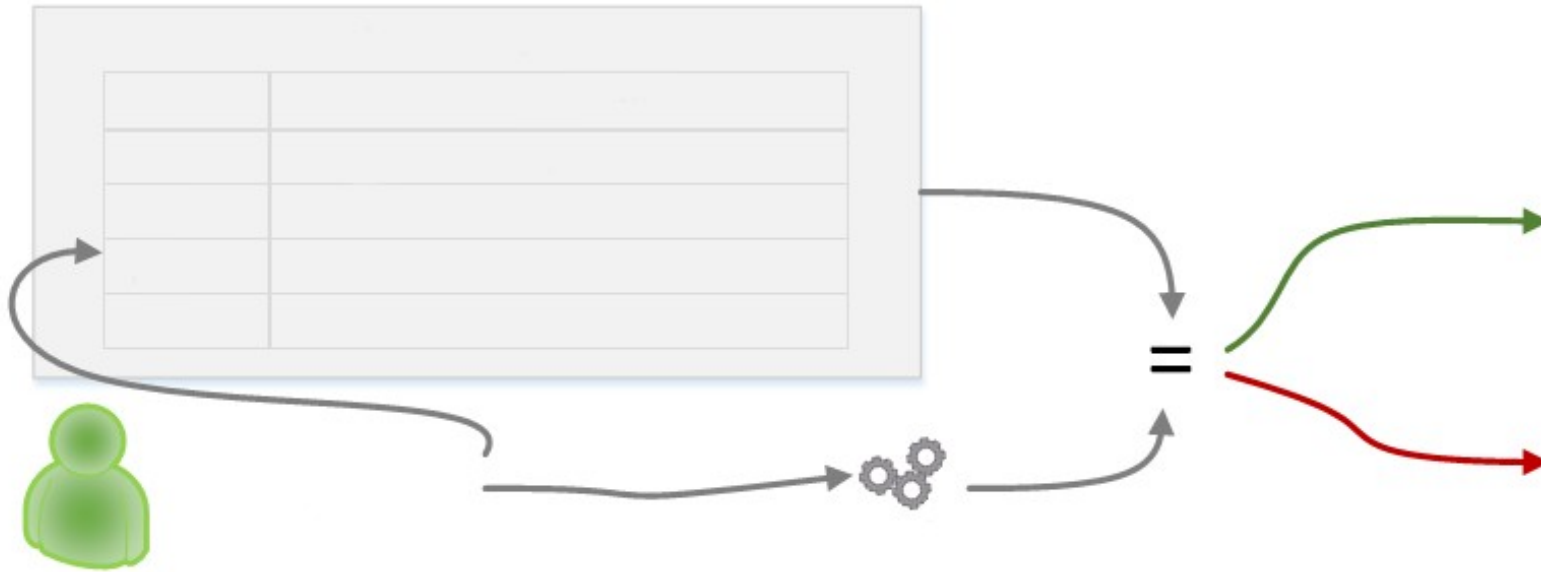
- User enters his plain text password

Validation of a password



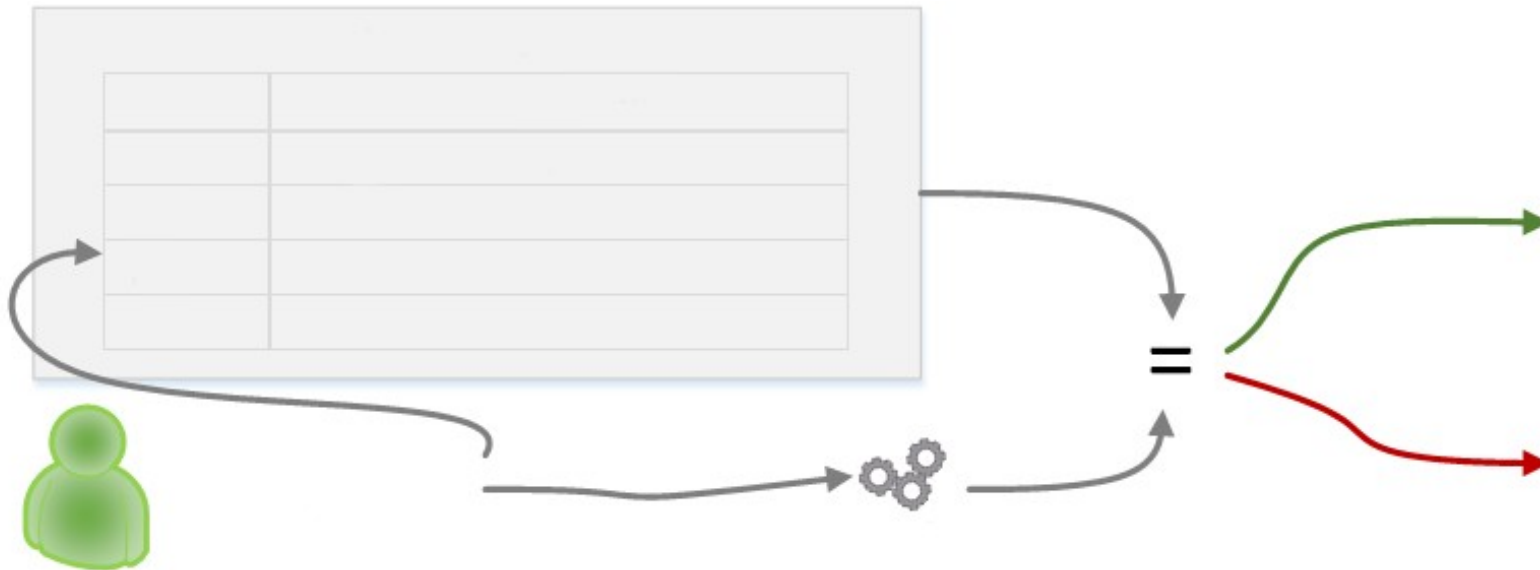
- Hashed password is retrieved from the database

Validation of a password



- Calculation of the hash for the entered plaintext password

Validation of a password



- Comparison of both hashes
- Login is successful if both hashes match

Safe hashes with salt (1/2)

- Password concealment by hashing has further weaknesses

<i>user</i>	<i>MD5 password hash</i>	<i>password</i>
User A	3e45af4ca27ea2b03fc6183af40ea112	password
User B	3e45af4ca27ea2b03fc6183af40ea112	password

- Two users with the same password also have the same hash in database
- If you know the password of a user, you know the passwords of all users with the same password.
- Hash can be guessed if you hash all words from a dictionary, such as password (see Attacks)

Safe hashes with salt (2/2)

Solution: Same password generates different hash each time

- **Idea:** Before calculating the hash value of a password, a random character string (salt / salt) precedes the password.

$$\text{hMD5}(\text{PASSWORD} + \text{SALT}) = \text{HASH}$$

- Then Salt is saved together with the password hash.

<i>user</i>	<i>MD5 password hash</i>	<i>salt</i>	<i>password</i>
User A	ec58ef6c6f345d99054e419d19be6e3f	5<§	password
User B	188623e1be9c480214838bf596b22d0d	=3(	password

Summary:

Authentication with passwords

- Authentication through **knowledge**
 - Person proves that he or she owns a digital identity by providing the correct password.
- Service stores the password when registering a Digital Identity and validates the password during the login process.
 - Passwords should **never** be stored in plain text, but in obfuscated form (*Password Hashing*)
 - To increase security, only secure hash procedures and salts should be used for password obfuscation.

Summary:

Authentication with passwords

- Authentication through **knowledge**
 - Person proves that he or she owns a digital identity by providing the correct password.
- Service stores the password when registering a Digital Identity and validates the password during the login process.
 - Passwords should **never** be stored in plain text, but in obfuscated form (*Password Hashing*)
 - To increase security, only secure hash procedures and salts should be used for password obfuscation.