Mitigating Password Database Breaches with Intel SGX

Helena Brekalo **Raoul Strackx** Frank Piessens

imec - Distrinet, KU Leuven

December 12, 2016

Outline

Passwords: The ugly truth

- Passwords are everywhere and heavily depended upon
- Developers are not always careful
- . . . nor are users
- Passwords are often not the most sensitive data

Passwords: The ugly truth

- Passwords are everywhere and heavily depended upon
  - Only a password and “security questions” are required
  - I’d like stronger protection for my money
- Developers are not always careful
- . . . nor are users
- Passwords are often not the most sensitive data

Passwords: The ugly truth

]HackingTeam[

- Passwords are everywhere and heavily depended upon
- Developers are not always careful
  - Sells offensive intrusion and surveillance capabilities
  - 400GB of lost data
- . . . nor are users
- Passwords are often not the most sensitive data

Passwords: The ugly truth

Booz | Allen | Hamilton

- Passwords are everywhere and heavily depended upon
- Developers are not always careful
- . . . nor are users
  - BAH: Consulting firm for Homeland Security, . . .
  - MD5-hashes *without* a salt
  - “123456” appeared 22x in the database
- Passwords are often not the most sensitive data

Passwords: The ugly truth

- Passwords are everywhere and heavily depended upon
- Developers are not always careful
- . . . nor are users
- Passwords are often not the most sensitive data
  - “Discretion matters”
  - 30M entries
  - Dates of birth, Names, Passwords, Sexual orientations, Website activity, . . .

Outline

Attacker Model

- Server-side protection
- Potential malicious cloud provider/compromised kernel
- Complete password data may be leaked

Security Properties

- Offline attacks:
  - Focus on stored passwords
  - Computational infeasible to break passwords
- Online attacks:
  - Enforce strong passwords
  - Guess-limit attacker

  $\rightarrow$ Out of scope
- Other sensitive data: Out of scope

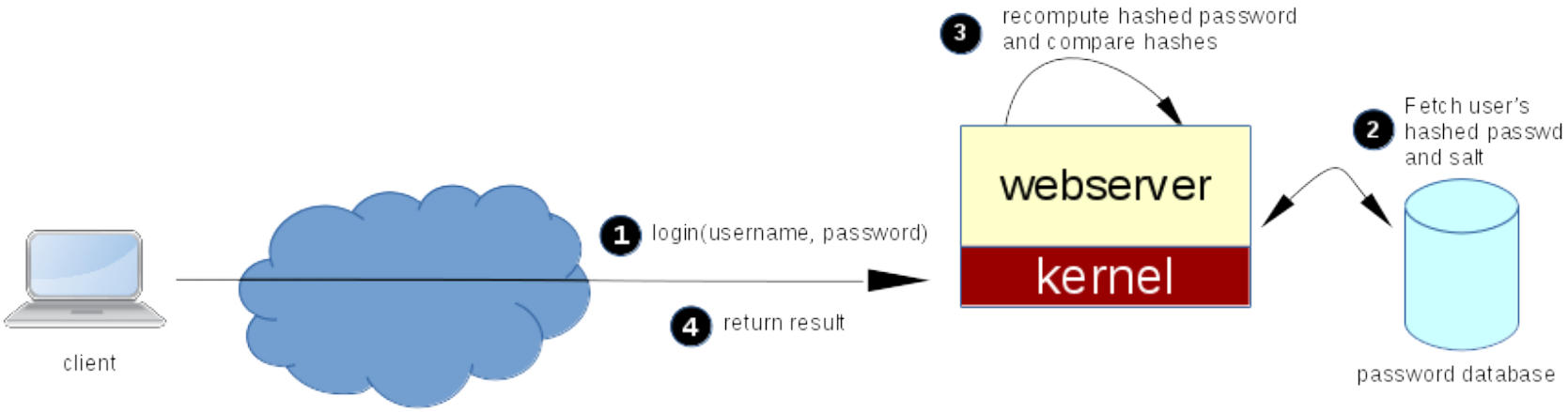
Why is this so hard?

"Weaken the attacker, without putting strain on the defender"

- PBKDF2: Reduce speed to hash passwords
- Scrypt: Increase required memory
- Separate HW: Keep (a part of) the password secret

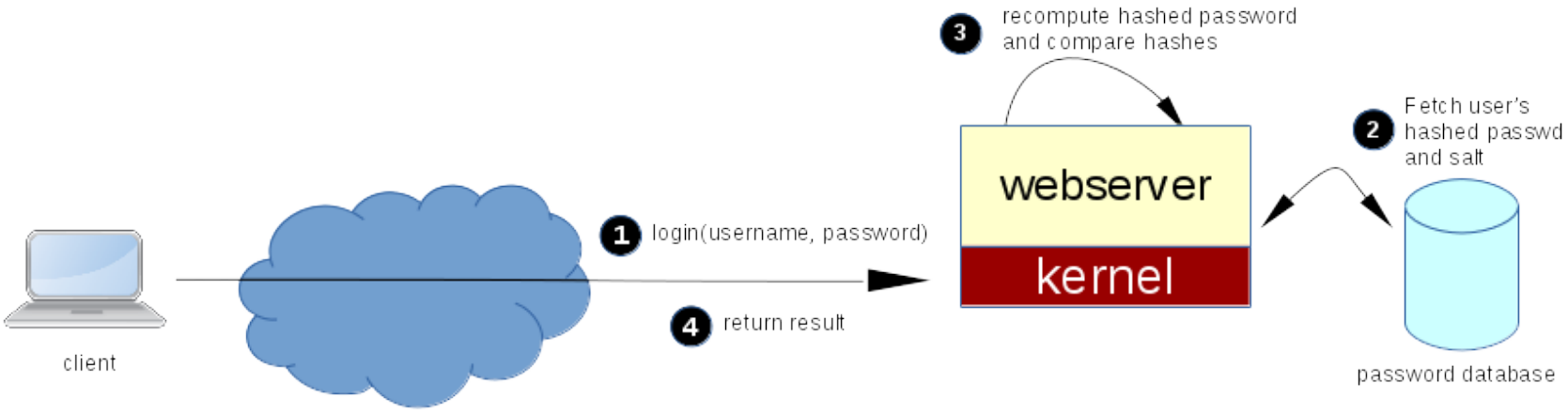
Setup

The bad approach



Setup

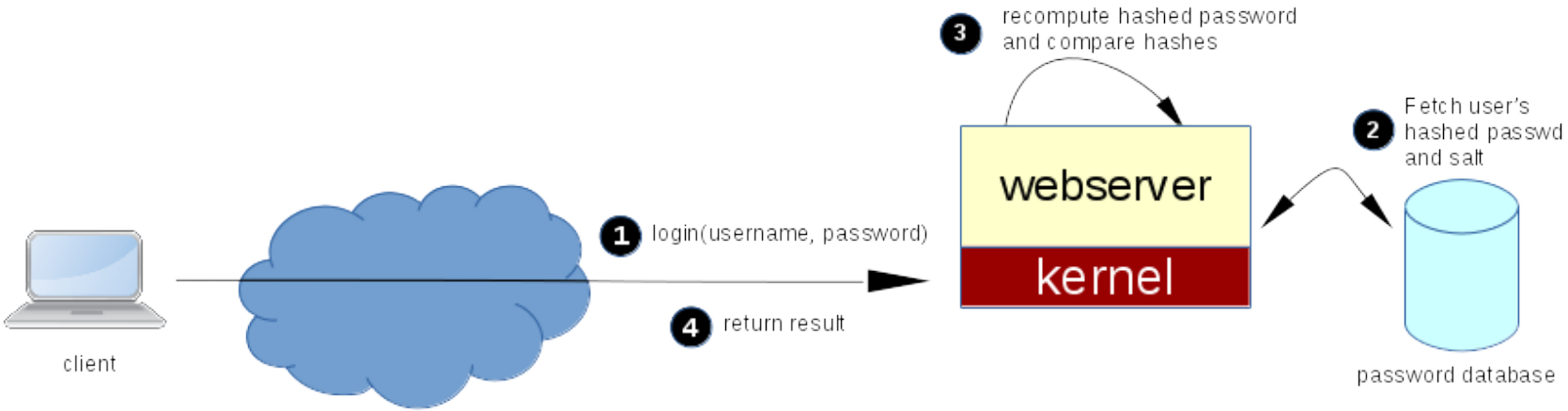
The bad approach



$$\text{password}_{\text{stored}} = \text{SHA1}(\text{password})$$

Setup

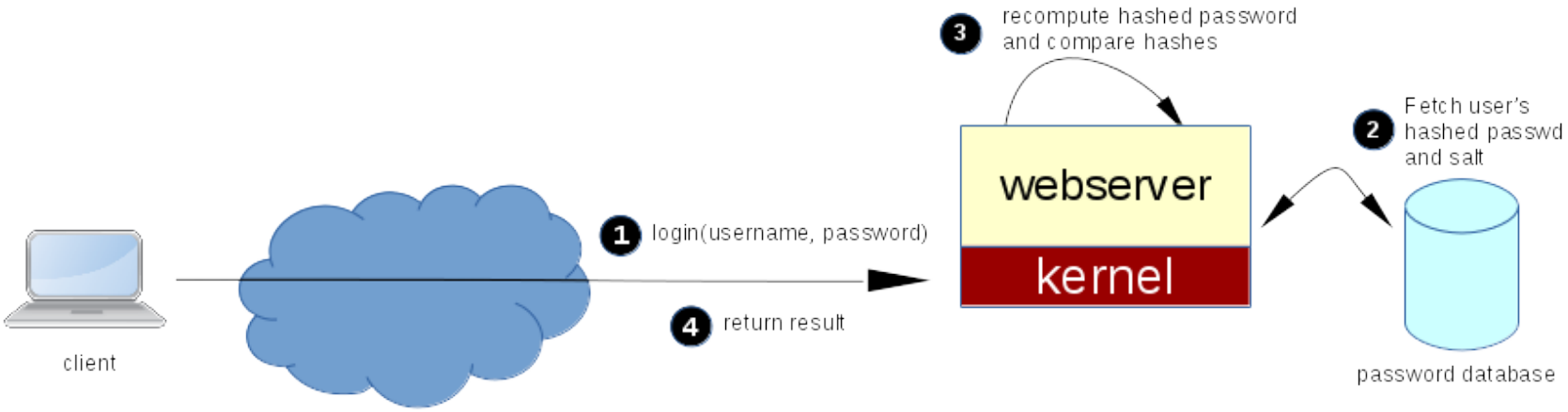
The bad approach



$$\text{password}_{\text{stored}} = \text{SHA1}(\text{password}||\text{salt})$$

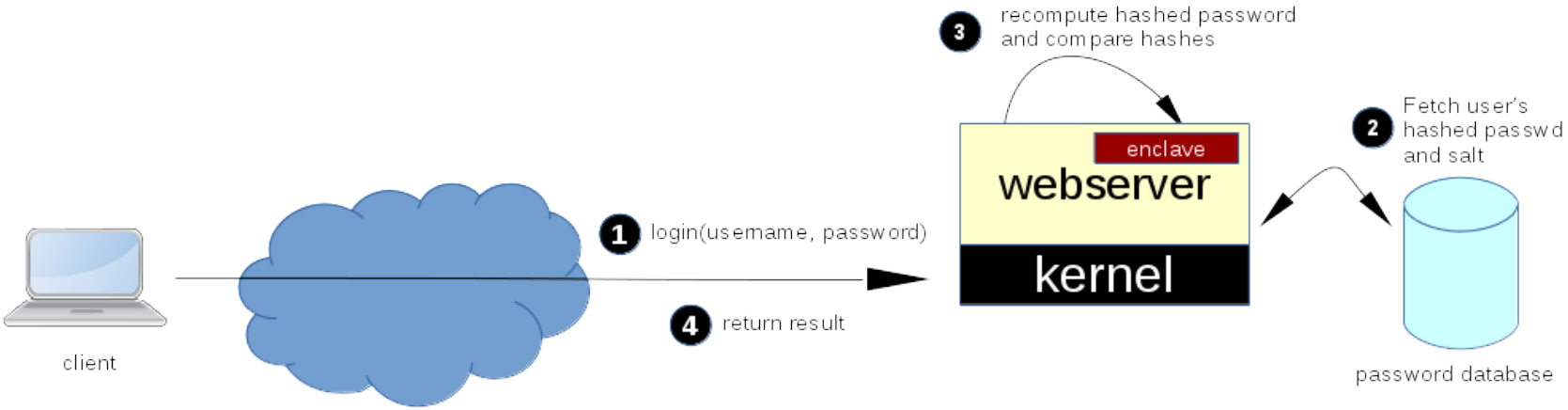
Setup

The bad approach

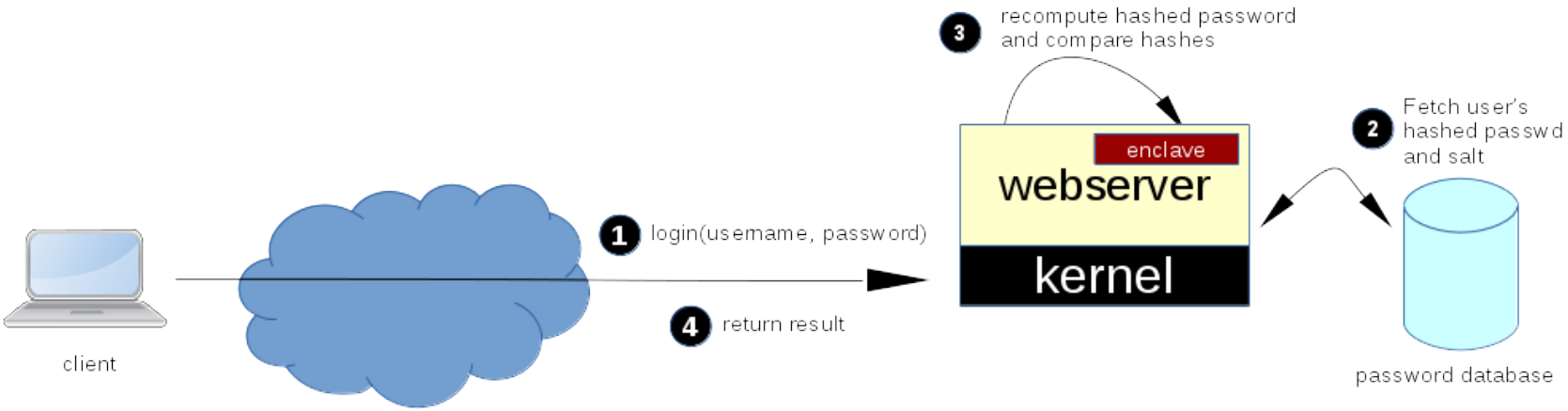


Problem: Offline bruteforce attacks

Iteration 1: Intel SGX to the rescue!

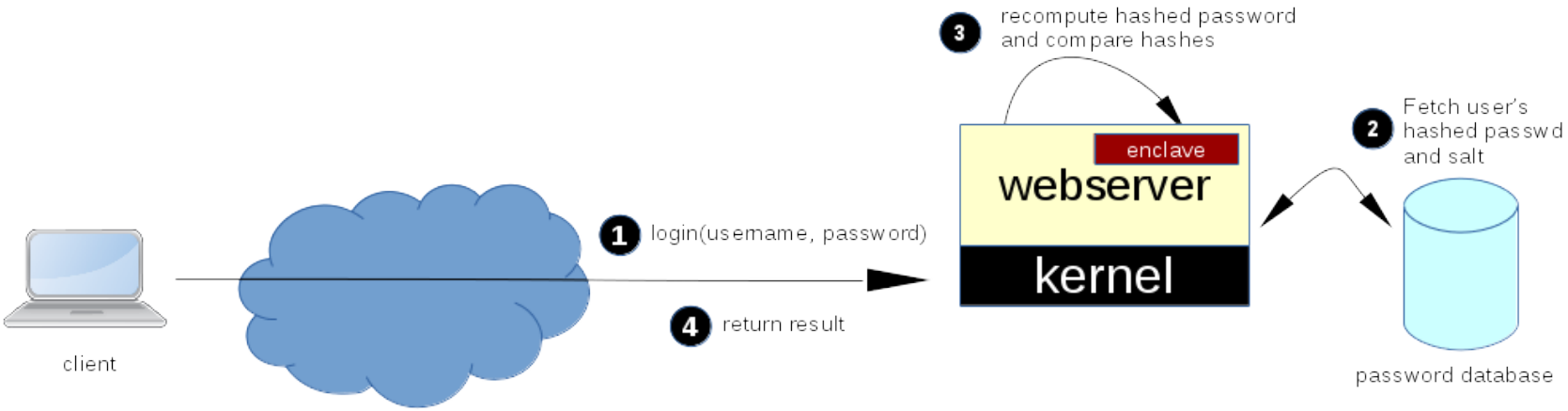


Iteration 1: Intel SGX to the rescue!



$$\text{password}_{\text{stored}} = \text{HMAC}(k, \text{password}||\text{salt})$$

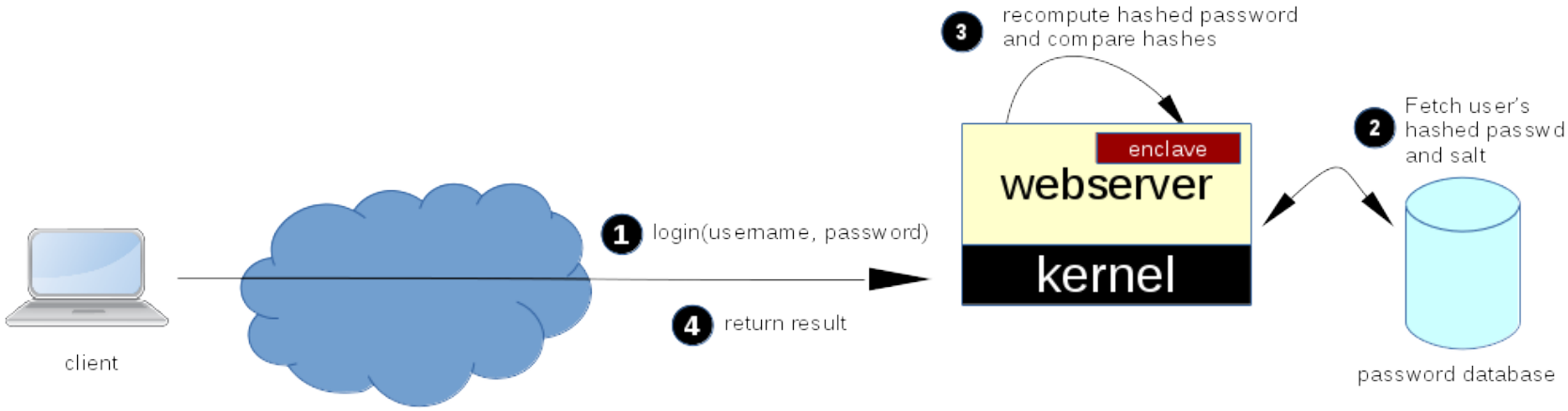
Iteration 1: Intel SGX to the rescue!



$$\text{password}_{\text{stored}} = \text{HMAC}(k, \text{password}||\text{salt})$$

k must never leave the enclave!

Iteration 1: Intel SGX to the rescue!



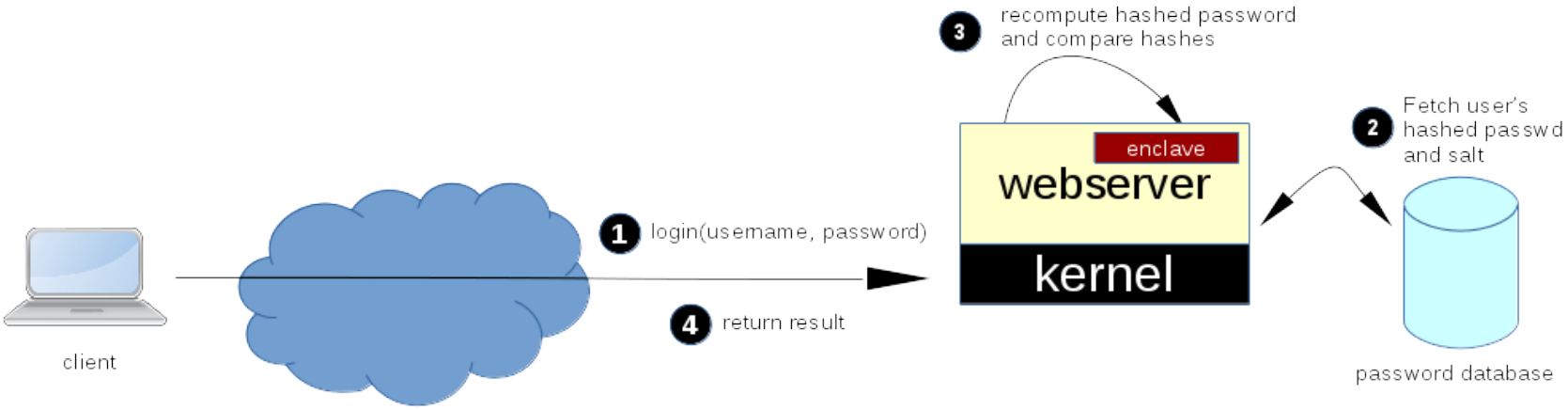
$$\text{password}_{\text{stored}} = \text{HMAC}(k, \text{password}||\text{salt})$$

Scenario 1: Attacker leaks PWD database

✓ no bruteforce attacks against passwords as k is never leaked.

✓ no hash-collisions for the same password

Iteration 1: Intel SGX to the rescue!

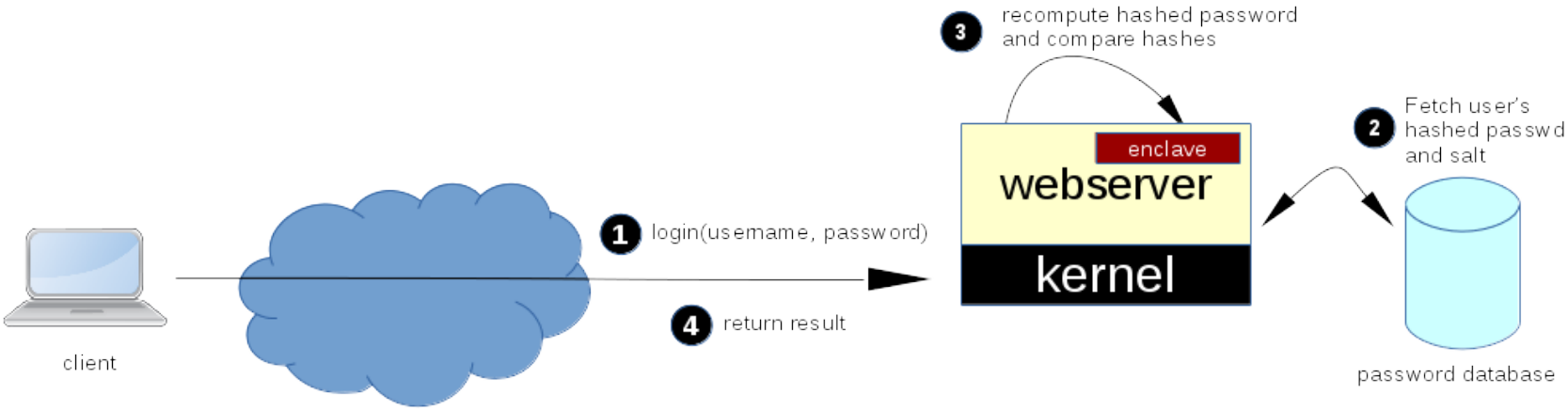


$$\text{password}_{\text{stored}} = \text{HMAC}(k, \text{password}||\text{salt})$$

Scenario 2: Attacker creates fake passwords then leaks PWD database

✓ no bruteforce attacks against passwords as *salt* ensures different hashes for different users.

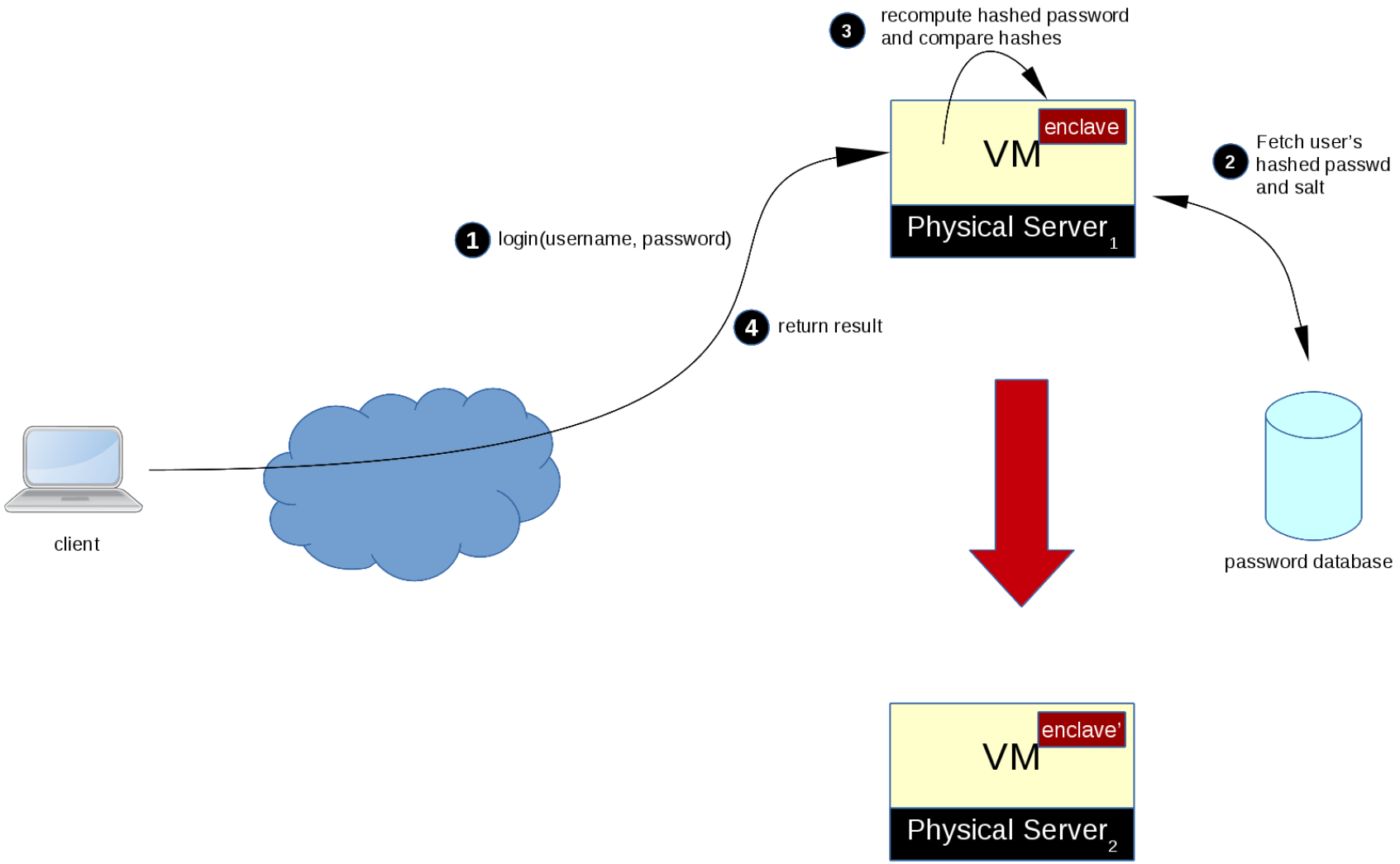
Iteration 1: Intel SGX to the rescue!



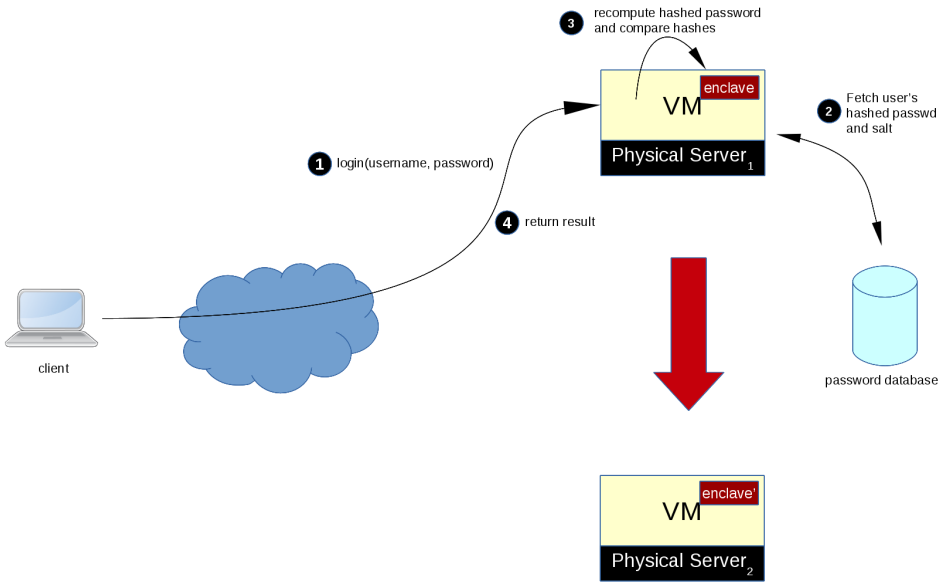
$$\text{password}_{\text{stored}} = \text{HMAC}(k, \text{password}||\text{salt})$$

Problem: In the cloud the enclave will have to move to a different VM

Iteration 2: Migratable Enclaves



Iteration 2: Migratable Enclaves

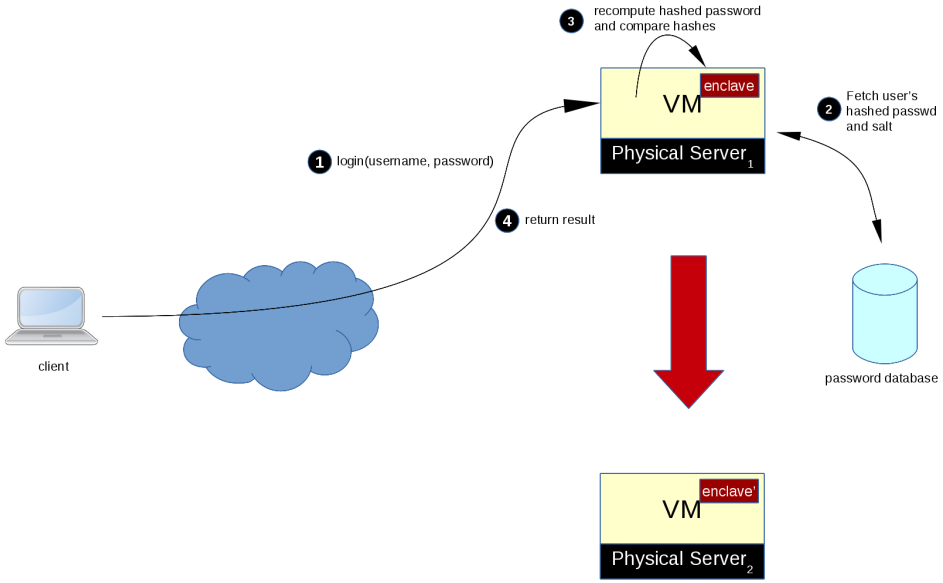


Option 1: Dedicated server to keep k

+ Easy
- Single point of failure
- Active defense mechanisms (e.g., guess limiting) need to communicate with the server

[1] Strackx and Lambrigts. “Idea: State-Continuous Transfer of State in Protected-Module Architectures”. 2015. *Engineering Secure Software and Systems*

Iteration 2: Migratable Enclaves

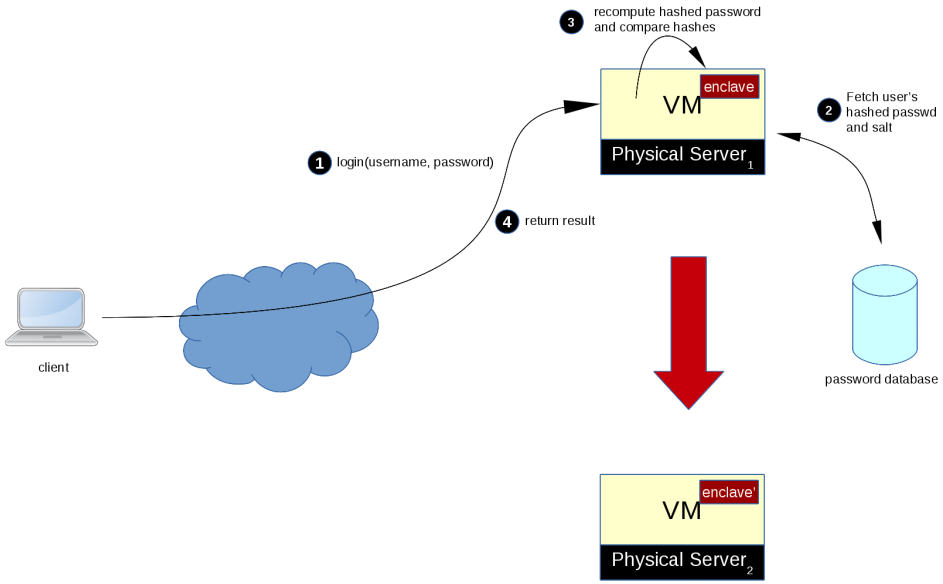


Option 2: End-to-end transfer of state-continuous enclave state

- \+ No single point of failure
- \- Both endpoints need to be active at the same time
- \- Active defense mechanisms (e.g., guess limiting) pose a challenge[1]

[1] Strackx and Lambrigts. “Idea: State-Continuous Transfer of State in Protected-Module Architectures”. 2015. *Engineering Secure Software and Systems*

Iteration 2: Migratable Enclaves

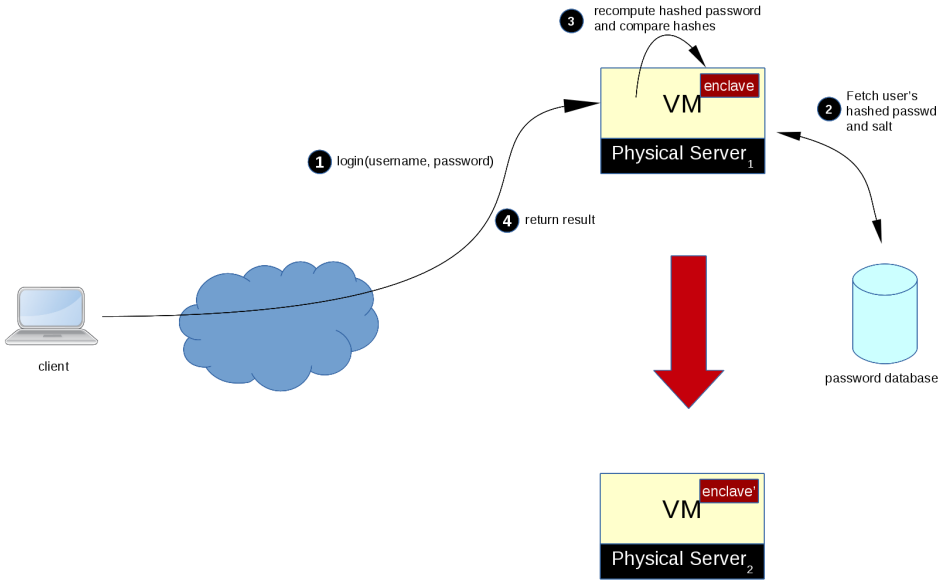


Option 3: True P2P network of enclaves

+ No single point of failure
+ Flexible
- Harder to implement
- Active defense mechanisms (e.g., guess limiting) pose an (unsolved) challenge

[1] Strackx and Lambrigts. "Idea: State-Continuous Transfer of State in

Iteration 2: Migratable Enclaves



Problem: Preventing an attacker to move the enclave to her own machine

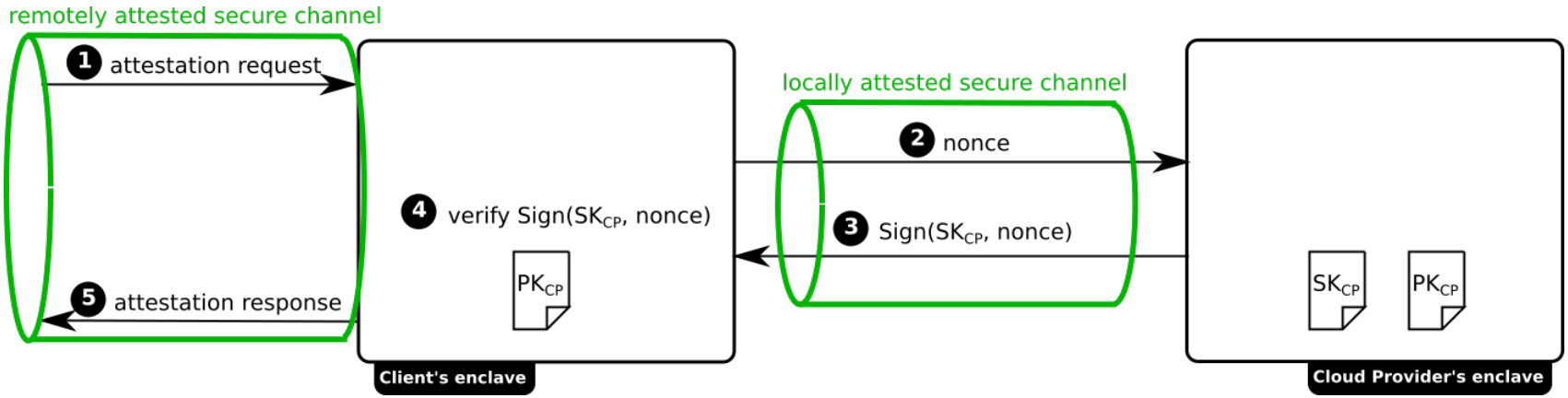
[1] Strackx and Lambrigts. “Idea: State-Continuous Transfer of State in Protected-Module Architectures”. 2015. *Engineering Secure Software and Systems*

A Modified Attestation Scheme

General idea:

1. Provide each VM with a cloud provider's enclave
2. Check during attestation that the password enclave is executing on the same machine as one of those particular enclaves

A Modified Attestation Scheme



✓ An attacker should not be able to create a cloud provider's enclave: keep SK_{CP} securely sealed

Outline

Performance Evaluation

Overhead is caused by:

- HMAC uses 2 SHA3 calls
- Entering/exiting the enclave is time consuming

	Without SGX	SGX
Algorithm	SHA3	SHA3-HMAC
Time (ms)	0.006788	0.046023

Table: Performance measures of the creation of passwords with and without the use of SGX.

Outline

Conclusion

- **Yes** SGX is a prime candidate to harden password mechanisms
- That can be implemented with minimal effort
- But it's a short-term solution
- SGX' sealing and attestation mechanism makes it ideal for 2FA: Proof that you possess 1 or multiple devices, secure from malware on these devices

Questions?

raoul.strackx@cs.kuleuven.be

Outline

Iteration 3: Active defense mechanisms

Easy to implement:

- Enforcing strong passwords
- No re-using passwords
- . . .

Hard to implement:

- Max. number of password guess
- Increasing timeout per guess

Iteration 3: Active defense mechanisms

Potential attacks:

- Rolling back state of the password enclave
- Forking multiple instances of the enclave

$\rightarrow$ State-continuity: Once a password is provided, the enclave should continue execution based on that input, or never advance at all.
How do you do this in a distributed environment?