

# 2BPR



BLIND PASSWORD REGISTRATION FOR TWO-SERVER PASSWORD AUTHENTICATED  
KEY EXCHANGE AND SECRET SHARING PROTOCOLS

*Franziskus Kiefer & Mark Manulis*

# Gliederung

1. Hintergrund
2. Motivation
3. Begriffe
4. Protokoll
5. Sicherheitsanalyse
6. Fazit



# Hintergrund - Multiusersystem



# Hintergrund

*Wie sichert man Multi-User-Systeme gegen Missbrauch ab?*



- Lange Passwörter
- Sonderzeichen & Zahlen
- Regelmäßiges Ändern



- Kontrolle der Richtlinien
- Plain Passwort Datenbanken
- Passworthashes nicht sicher

# Hintergrund

*Wie sichert man Multi-User-Systeme gegen Missbrauch ab?*



Individueller Hash durch Salt  
Verhindert Lookuptables  
Regelmäßiges Ändern



Zu kurzer oder schlechter Salt  
Kontrolle der Richtlinien

# Hintergrund

*Wie sichert man Multi-User-Systeme gegen Missbrauch ab?*



Mehrere Server nutzen  
Passwort „verteilen“  
2PAKE oder 2PASS



Kontrolle der Richtlinien

# Hintergrund – 2PAKE & 2PASS

## PASSWORD AUTHENTICATED KEY EXCHANGE

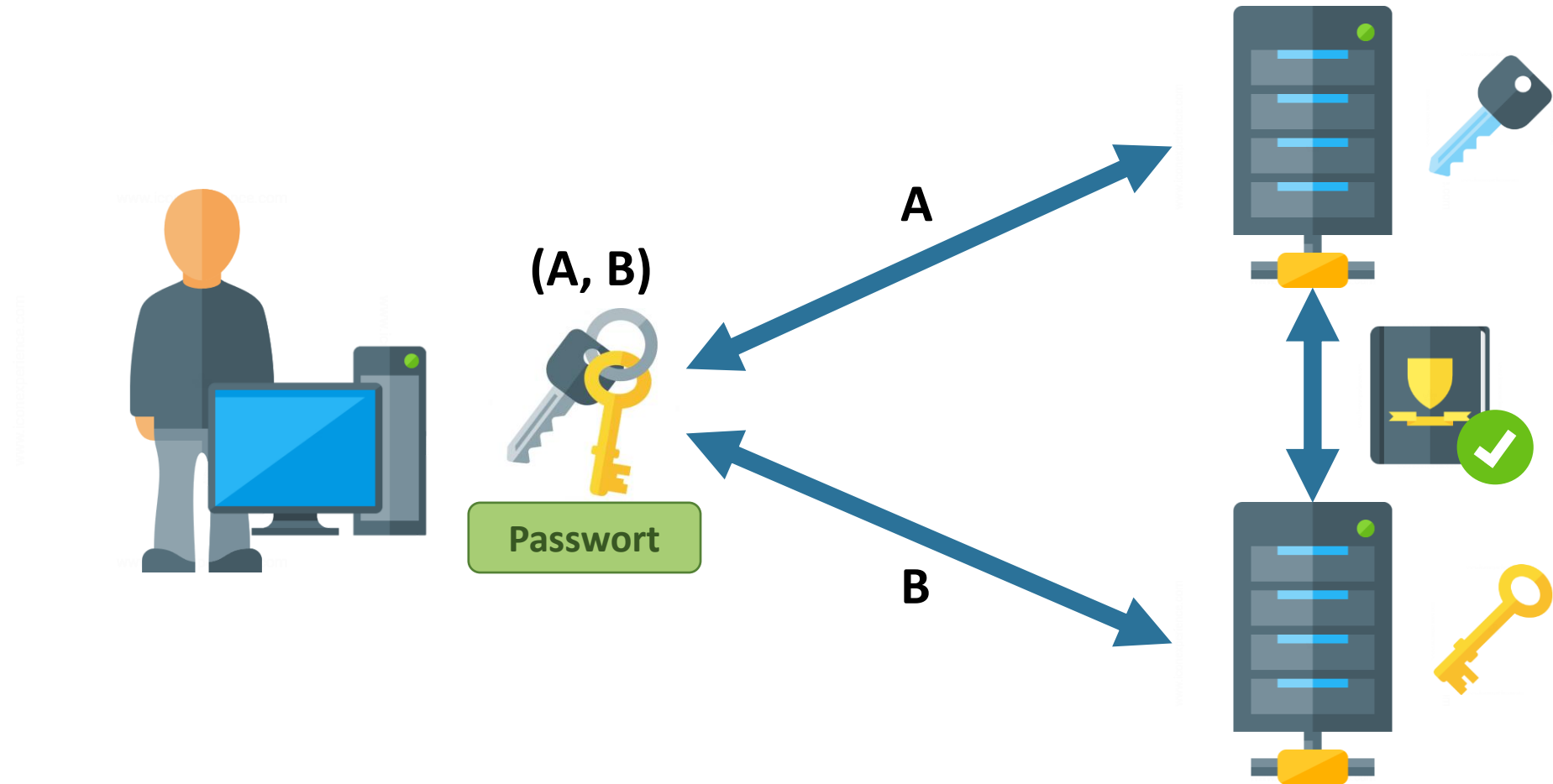
- Passwort wird in  $s_1$  und  $s_2$  geteilt
- $s_1$  und  $s_2$  auf zwei Servern speichern
- Zusammenarbeit der Server bei Login
- Kein Server kennt das ganze Passwort

## PASSWORD AUTHENTICATED SECRET SHARING

- Passwort mit hoher Entropie auf mehreren Servern verteilen
- Passwort mit niedriger Entropie autorisiert den Abrufprozess des ganzen Passworts

 Man kann beiden Servern zu 100% vertrauen

# Hintergrund – 2PAKE & 2PASS





# Hintergrund – 2PAKE & 2PASS



Man in the Middle



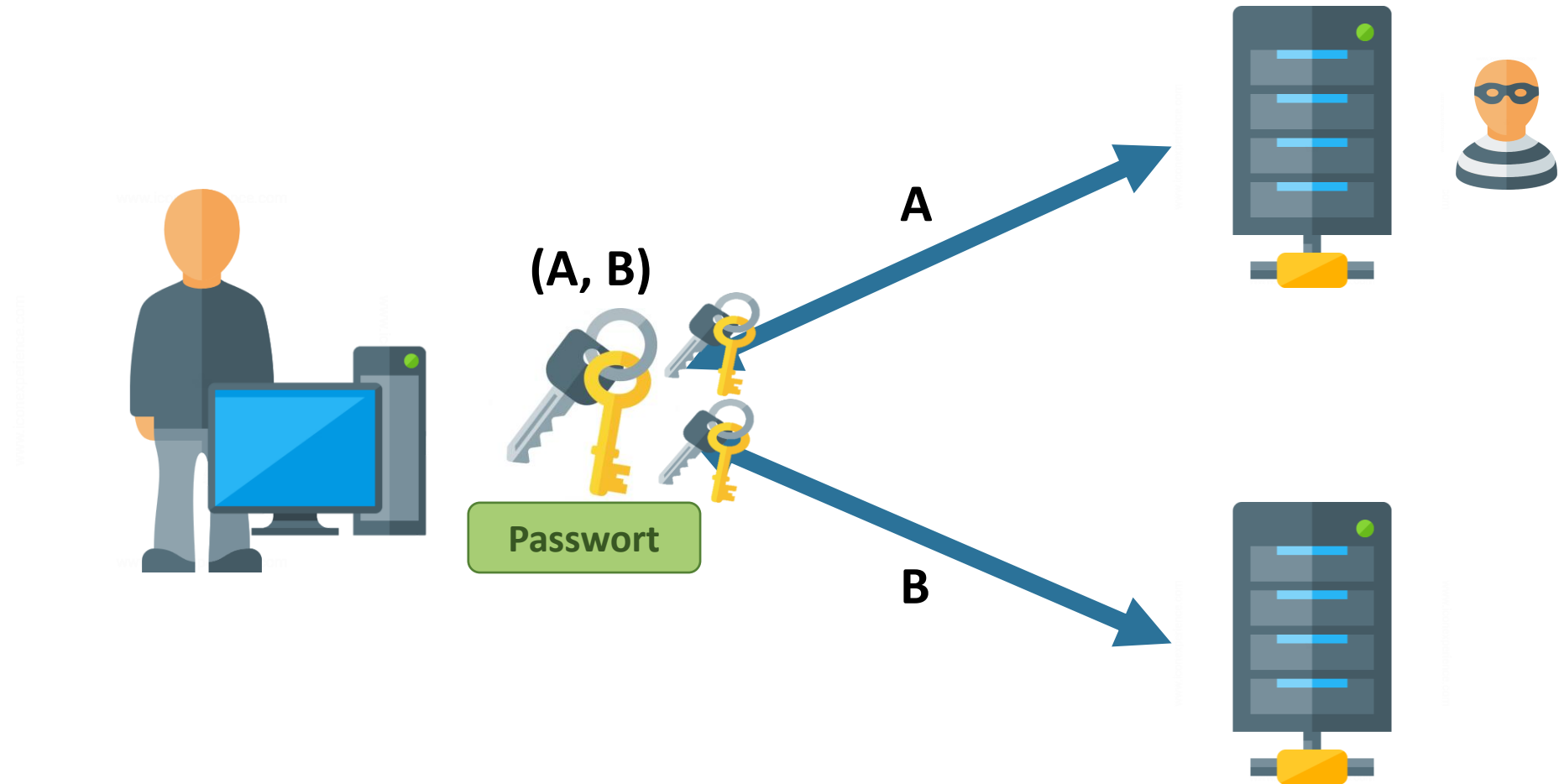
Brutforce

# Gliederung

1. Hintergrund
2. **Motivation**
3. Begriffe
4. Protokoll
5. Sicherheitsanalyse
6. Fazit



# Motivation



# Motivation



2BPR



~~Kontrolle der Richtlinien~~

- Commitments & Zero Knowledge Password Policy Checks (ZKPPC)
  - Keine offline Wörterbuch Attacken möglich
  - Sichere Registrierung von neuen Passwörtern
- ➔ Sicherer Registrierungsprozess in 2PAKE & 2PASS Multiusersystemen

# Motivation - Protokoll

2BPR ist ein Protokoll, doch was ist ein Protokoll?

*„Ein Protokoll ist in der Datenkommunikationstechnik eine definierte Vereinbarung über die Art und Weise des Informationsaustauschs zwischen zwei Systemen“ [7]*

- + Fester zeitlicher Verlauf mit klaren Regeln
- + Klar definierte Pakete/Nachrichten

vgl. OAuth2.0



# Gliederung

1. Hintergrund
2. Motivation
3. Begriffe
4. Protokoll
5. Sicherheitsanalyse
6. Fazit



# Commitment

- Szenario: SSP oder Münzwurf über Internet spielen
- Bedingung: Kein TrustCenter vorhanden



- ⚡ Keiner der Spieler darf auf den Zug des anderen reagieren
- ⚡ Das Festlegen auf Schere / Stein/ Papier muss verbindlich sein
- 💡 Verwendung eines Commitments <sup>[9]</sup>

# Commitment

- Binding



Bob legt sich auf Zahl fest → kein Umentscheiden möglich



Bob wählt Stein, Alice wählt Papier → kein Umentscheiden möglich

- Hiding

Gleichzeitiges abgeben der Commitments ist nicht möglich

Der Inhalt muss bis zum „Aufdecken“ versteckt bleiben



# Pedersen Commitment

Basis: Diskreter Logarithmus

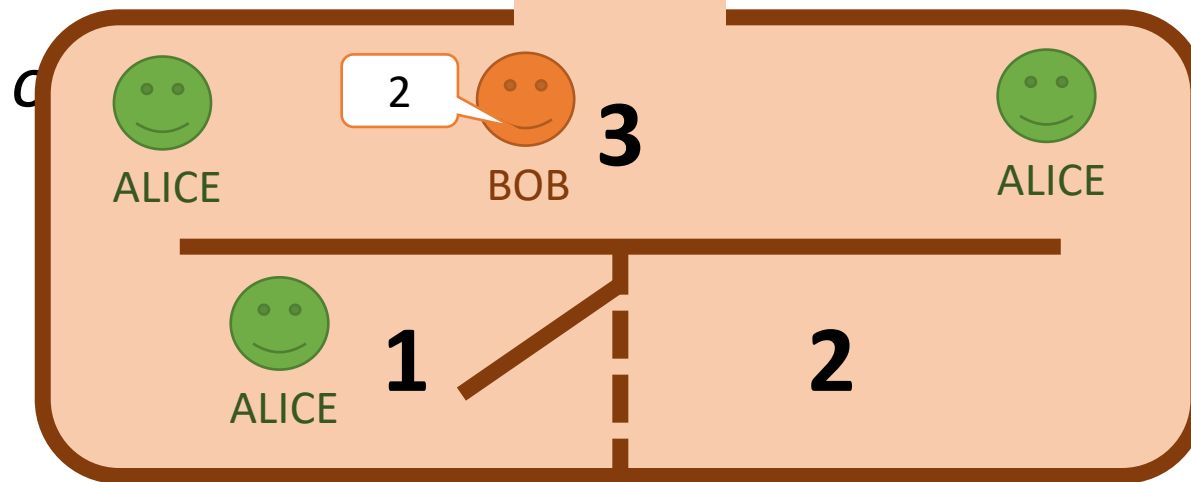
**TRAPDOOR COMMITMENT**

1. Setup      Bob legt Primzahlen  $p, q, g$  und  $v$  fest
2. Committed      Alice berechnet Commitment aus  $c = g^r v^m$
3. Aufdecken      Alice sendet  $r$  und  $m$  an Bob

- + Unconditional hiding
- + Computational binding
- + Großer Wertebereich für Nachricht
- + Additiv homomorph

# Zero Knowledge Proof

„Beweiser  $P$  überzeugt Verifizierer  $V$  davon, dass er ein Geheimnis kennt



Informationen zu offenbaren.“<sup>[6]</sup>

- Kennt kein Geheimnis
- Zu 50% falsche Seite
- Kennt Geheimnis
- Sicher zu  $P = 1 - 2^{-n}$

# Zero Knowledge Proof

---

- Vollständigkeit

Ist  $x$  ein Element der Sprache  $L$ , dann soll  $V$  fast immer akzeptieren.

- Zuverlässigkeit

Ist  $x$  kein Element der Sprache  $L$ , also ist  $P$  unehrlich, soll  $V$  fast immer ablehnen.

- Zero-Knowledge-Eigenschaft

Es darf nur Wissen über die Gültigkeit einer zu beweisenden Aussage gewonnen werden. Ein dritter, der das Verfahren beobachtet gewinnt keine Informationen.

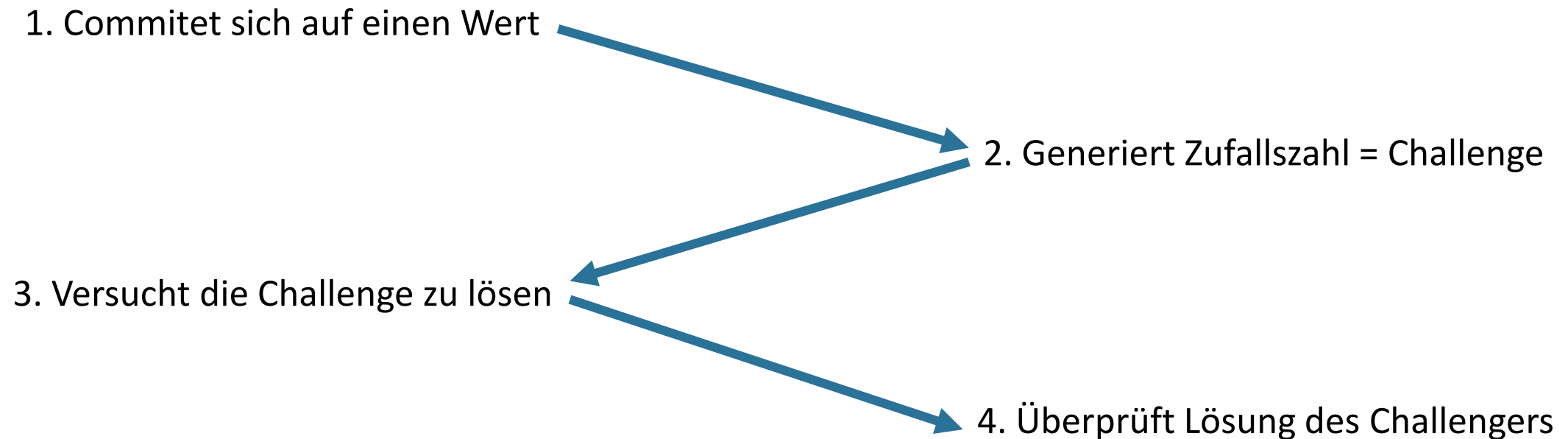
# Zero Knowledge Proof



ALICE



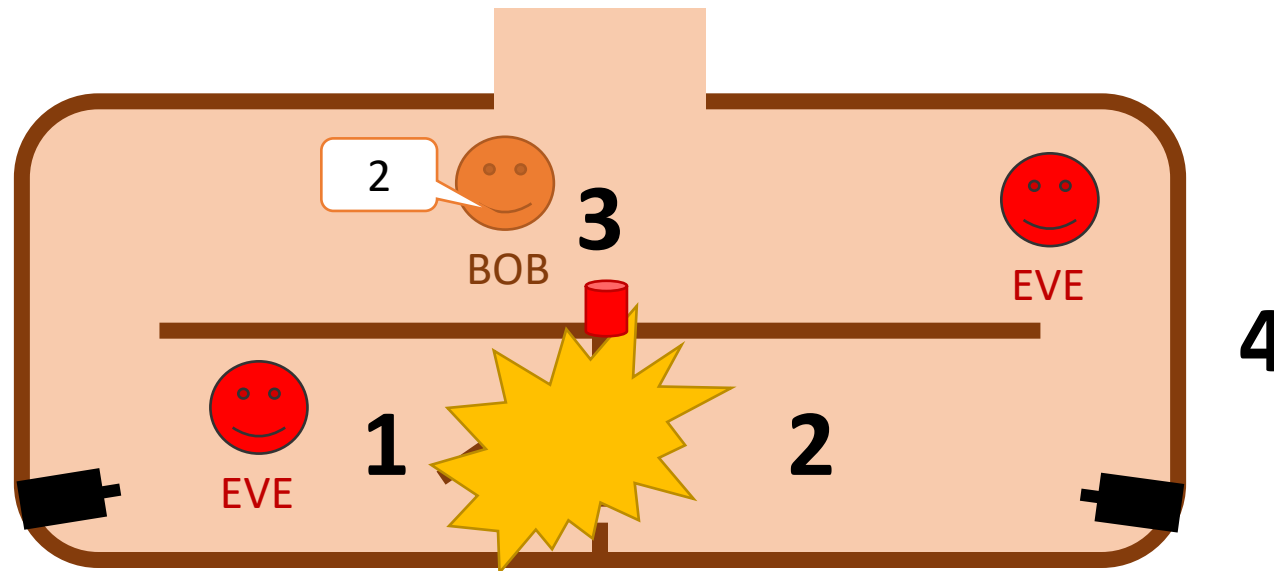
BOB



# Zero Knowledge Proof of Knowledge

- Zuverlässigkeit

Es gibt einen Extraktor *Ext*, der den korrekten Beweis aus einem bösen *P* extrahieren kann, sodass *V* den Beweis doch noch ablehnt.



# Passwörter

- Passwort Richtlinien

Besteht aus regulärem Ausdruck & Angabe für die Mindestlänge des Passworts

Beispiel:  $f = f(R, n) = (ulldds, 10)$

Um die beiden Richtlinien zu kombinieren wird  $f = f_1 \cap f_2$  gebildet

Beispiel:  $f1 \cap f2 = (max(R_1, R_2), max(n_1, n_2)) \rightarrow$  Mutual Password Policy

- Passwörter werden in Integer umgewandelt

$$\pi = PWDtoINT(pw)$$

# Passwörter

- Passwörter können zerteilt werden (Password Sharing)

$$\pi = s_1 + s_2$$

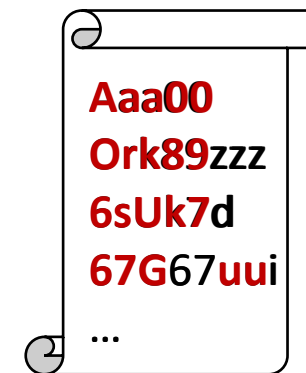
$s_1$  kann auf Server 1 hinterlegt werden

$s_2$  kann auf Server 2 hinterlegt werden

- Passwort Wörterbuch

Liste aller richtlinienkonformen Passwörter

Beispiel:  $f = (ulldd, 5)$



**SIGNIFIKANT**

# Gliederung

1. Hintergrund
2. Motivation
3. Begriffe
4. Protokoll
5. Sicherheitsanalyse
6. Fazit





# 2BPR – Sicherheitsmodell

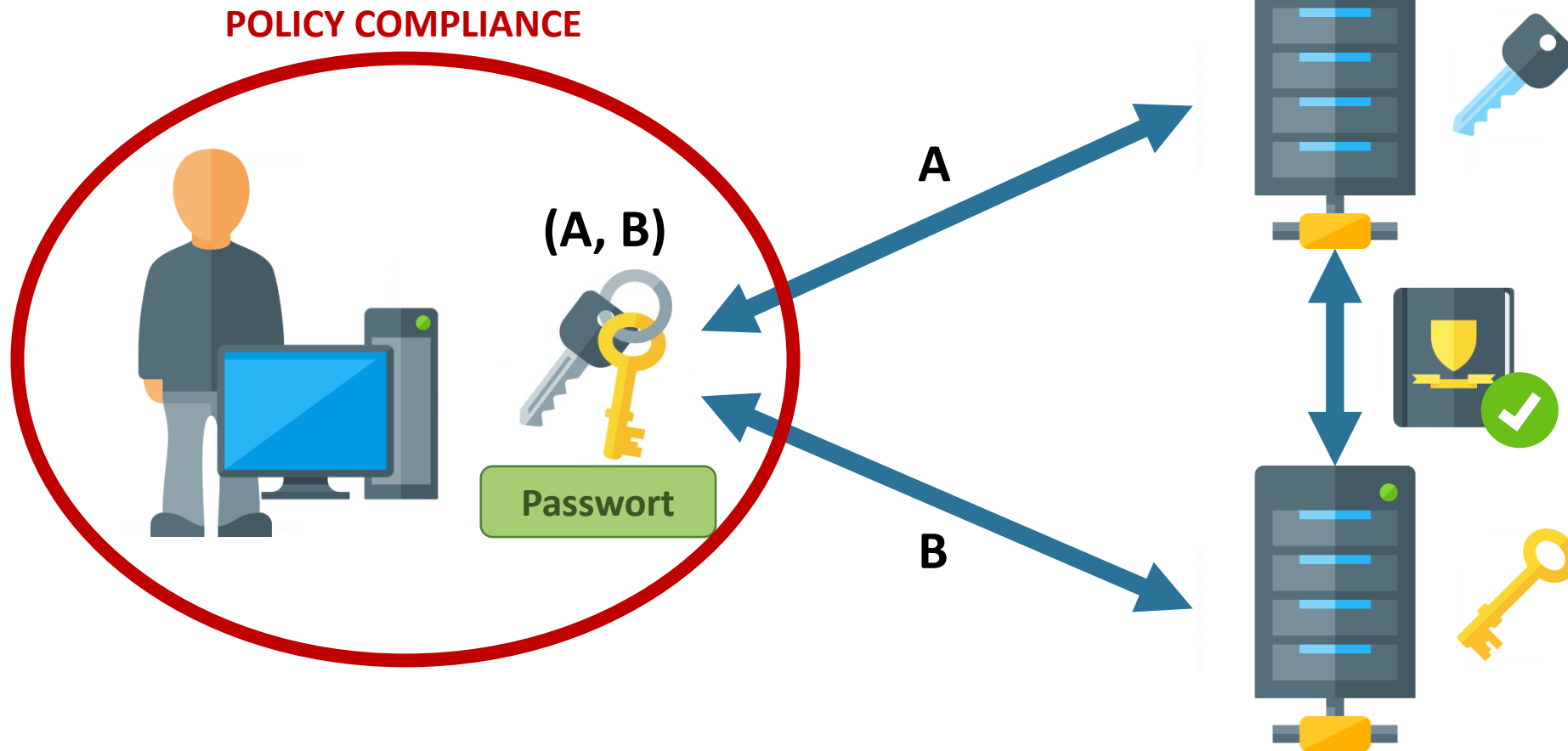
---

## 1. POLICY COMPLIANCE

Die beiden ehrlichen Server akzeptieren ihren Passwortshare, wenn dieser Policy konform ist, ansonsten lehnen sie den Share ab.

→ Wenn beide den Share akzeptieren ist das Passwort konform zur Mutual Password Policy.

# 2BPR – Sicherheitsmodell



# 2BPR – Sicherheitsmodell

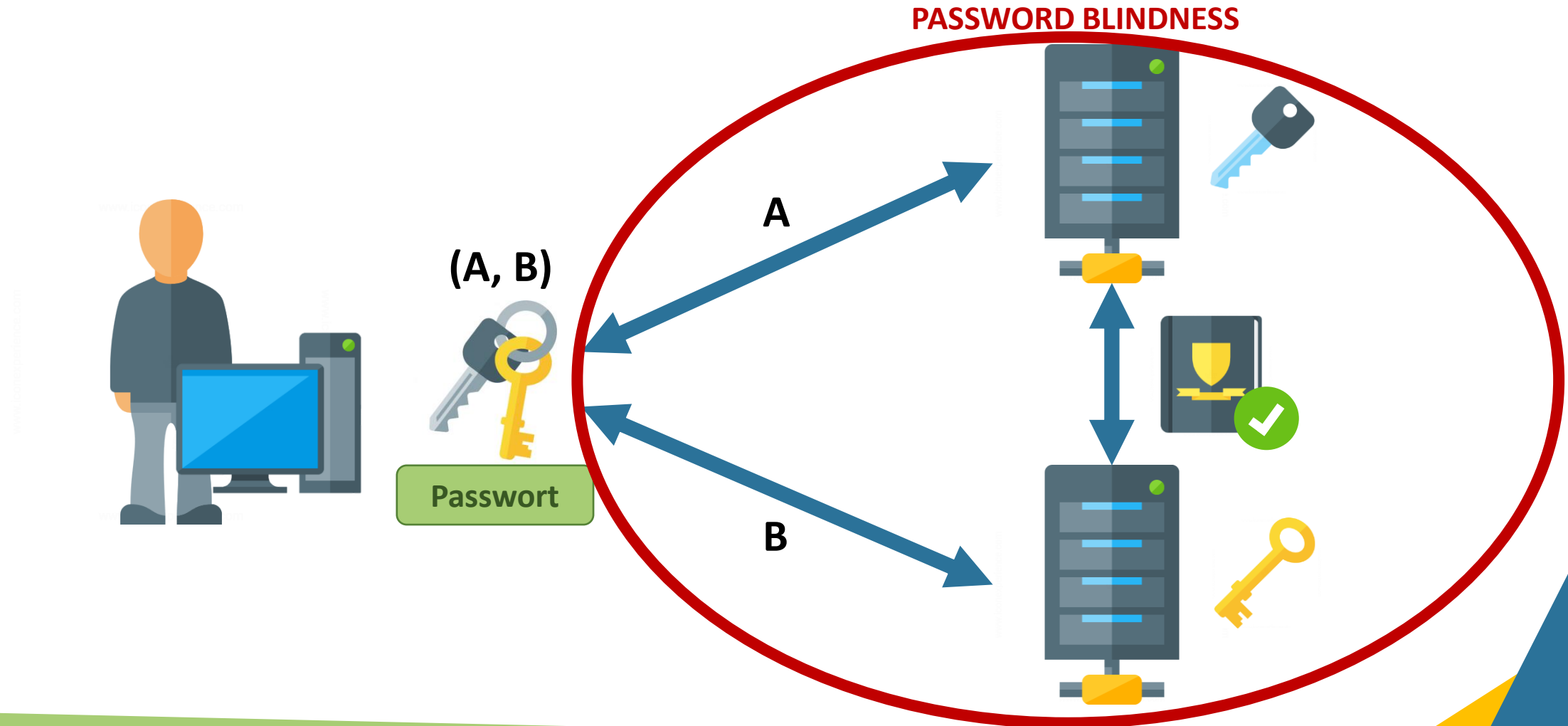
---

## 2. PASSWORD BLINDNESS

Ein (potentiell) korrumpierter Server soll nur erfahren, ob das Passwort Policy konform ist. Weitere Infos über das Passwort bleiben geheim.

→ Offline Wörterbuch Attacken sind dadurch zwecklos auch wenn ein Server korrumpiert wird oder die Kommunikation länger beobachtet wird.

# 2BPR – Sicherheitsmodell



# 2BPR – Phasen

---

## 1. Client Vorbereitung

Der Client bereitet Primzahlen, Passwort und Commitments vor

## 2. Passwort Registrierung

Der Client bestätigt die Konformität des Passworts gegenüber den Servern

## 3. Share Verifikation

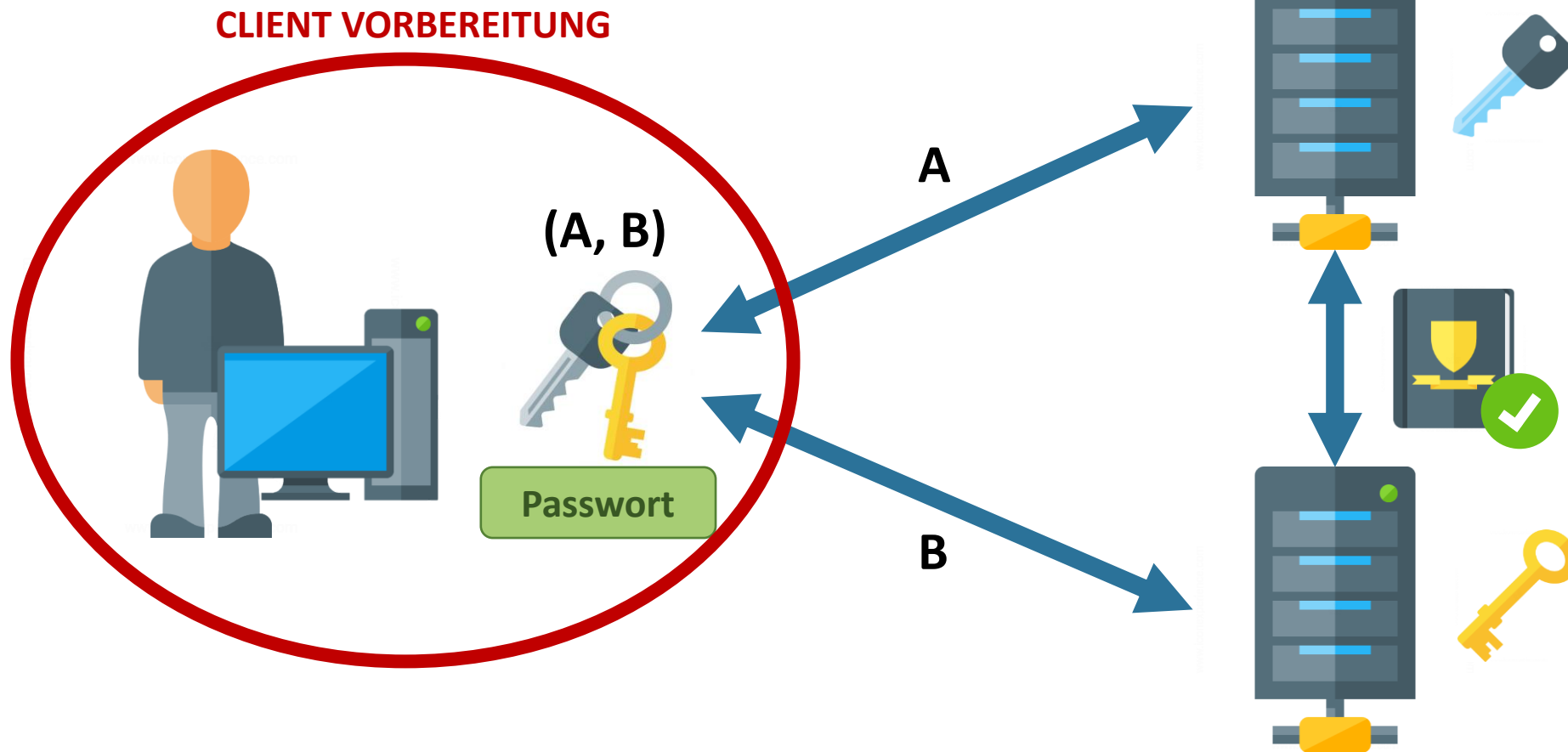
Die Server testen ob der Client mit beiden Servern dasselbe Passwort und dieselben Shares verwendet hat

# 2BPR – Client Vorbereitung

---

- 1.1 Der User wählt ein Passwort
- 1.2 Der Client wandelt das Passwort in einen Integer um
- 1.3 Der Client berechnet die Password Shares
- 1.4 Der Client berechnet Commitments für die Shares und das Passwort

# 2BPR – Client Vorbereitung



# 2BPR – Client Vorbereitung

1.1 Der User wählt ein Passwort

$$pw = P4s5$$

1.2 Der Client wandelt das Passwort in einen Integer um

$$\pi = 450 = PWDtoINT(pw)$$

1.3 Der Client berechnet die Password Shares


$$S_0 = 300 \quad S_1 = 450 - 300 = 150$$

1.4 Der Client berechnet Commitments für die Shares und das Passwort

$$C_0 = \text{Commitment zu } S_0 \quad P_0 = \text{Commitment aus } C_0 \text{ und } S_1$$



# 2BPR – Passwort Registrierung

- 2.1 Der Client wandelt jeden Passwortcharacter in einen Integer um
- 2.2 Der Client commitet sich auf jeden Character
- 2.3 Der Client commitet sich auf jedes zuvor erstellte Commitment
- 2.4 Der Client shuffelt die Commitments aus 2.3
- 2.5 Der Client erzeugt Menge  $w$  die für die Policykontrolle verwendet wird
-  2.6 Der Client führt mit beiden Servern POM, POC und POS aus

# 2BPR – Proof of Correctness

Ziel: Verknüpft die beiden Shares der Server und beweist, dass die beiden Shares zu einem Policy konformen Passwort gehören und zeigt, dass es einen anderen Passwortshare gibt.

$$\pi = s_0 + s_1 \rightarrow f(\pi) = \text{korrekt}$$

Mittel:  $\pi$ , Commitment für Share des anderen Server,  
Commitment für Passwort, Zero Knowledge Proofs

# 2BPR – Proof of Membership

---

Ziel: Beweist, für jeden Character aus dem Passwort, dass der Integerwert in  $w$  enthalten ist → Policykontrolle findet statt

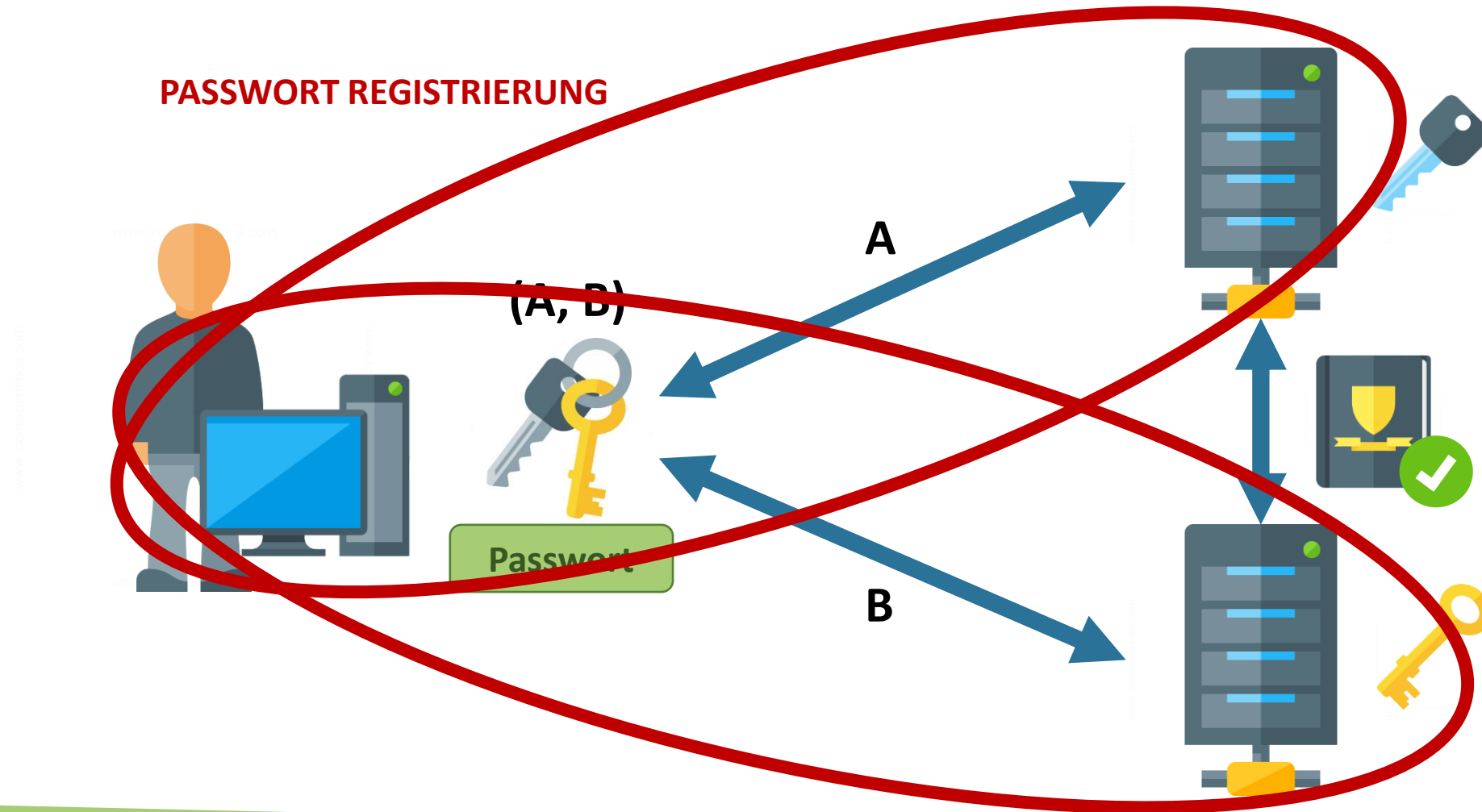
# 2BPR – Proof of Shuffle

Ziel: Beweist, dass das Passwort so durchmischt wurde, sodass ein Angreifer nicht zurückverfolgen kann, in welcher Reihenfolge die signifikanten und nicht signifikanten Character ursprünglich waren.

**SELBSTSTUDIUM ;)**

1. Paper: Efficient and verifiable shuffling and shuffle-decryption
2. Paper: An efficient scheme for proving a shuffle

# 2BPR – Passwort Registrierung



# 2BPR – Passwort Registrierung

2.1 Der Client wandelt jeden Passwortcharacter in einen Integer um

$$\pi_0 = 30 = \text{CHRtoINT}(pw_0) \quad pw_0 = „P“$$

2.2 Der Client commitet sich auf jeden Character

$$K_i = \text{Commitment zu } \pi_i \text{ mit Zufallszahl } r_i$$

2.3 Der Client commitet sich auf jedes zuvor erstellte Commitment

$$K_i' = \text{Commitment zu } K_i \text{ mit Zufallszahl } r_i'$$

2.4 Der Client shuffelt die Commitments aus 2.3

$$w = \text{geschuffelte Menge von } K_i$$

2.5 Der Client erzeugt Menge w die für die Policykontrolle verwendet wird

$$f = (UL, 2) \quad 45sP \rightarrow \sum \sum L U = w$$



2.6 Der Client führt mit beiden Servern POM, POC und POS aus

$$\text{Sende } (K_i, K_i', w, \Phi, \pi_i, l_i, r_i, r_i')$$

# 2BPR – Share Verifikation

---

Ziel: Verifikation, dass der Client mit beiden Servern das selbe Passwort und die selben Shares verwendet hat.

Mithilfe der in der „1. Phase Client Vorbereitung“ erzeugten Commitments für die Shares  $C_0$ ,  $C_1$  und das Passwort  $P_0$ ,  $P_1$  wird überprüft, ob beide Server das selbe Passwort erhalten haben.

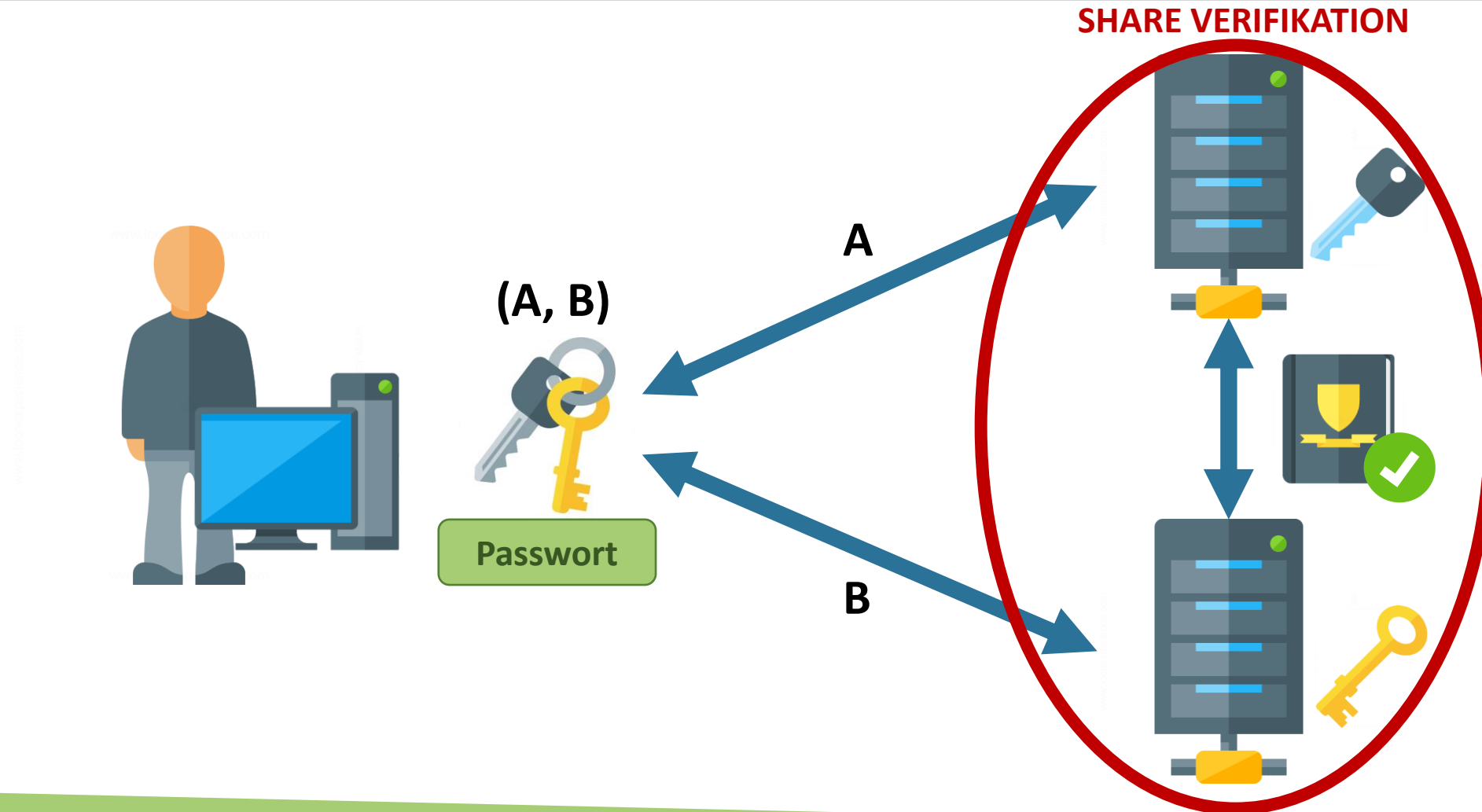
## 2BPR – Share Verifikation

---

- 3.1  $\text{Server}_0$  deckt das Commitment zum Passwort auf mit  $\text{Share}_0$
- 3.2  $\text{Server}_0$  sendet den aufgedeckten Wert an  $\text{Server}_1$
- 3.3  $\text{Server}_1$  deckt das Commitment zum Passwort auf mit  $\text{Share}_1$
- 3.4  $\text{Server}_1$  sendet den aufgedeckten Wert an  $\text{Server}_0$
- 3.5  $\text{Server}_0$  und  $\text{Server}_1$  überprüfen die Commitments auf Korrektheit



# 2BPR – Share Verifikation



# Gliederung

1. Hintergrund
2. Motivation
3. Begriffe
4. Protokoll
5. Sicherheitsanalyse
6. Fazit



# Sicherheitsanalyse

---

- Proof of Membership

Sorgt dafür, dass alle Passwörter Policy konform sind

$$pw \in D_f$$

- Proof of Correctness

Sorgt dafür, dass die beiden Shares zu einem Policy konformen Passwort gehören

$$S_0 + S_1 = \pi$$

- Proof of Shuffle

Zeigt, dass der Client dazu gezwungen wird, dass Passwort zu durchmischen

# Gliederung

1. Hintergrund
2. Motivation
3. Begriffe
4. Protokoll
5. Sicherheitsanalyse
6. Fazit



# Fazit

---

## *Ziel*

Sichere Registrierung von Passwörtern in 2PAKE / 2PASS Systemen mit Kontrolle der Passwortrichtlinien.

## *Mittel*

Pedersen Commitments und Zero Knowledge Proofs of Knowledge

## *Ergebnis*

Fertiges Protokoll mit Beispielimplementierung in Python

# Fazit

---

## *Sicherheit*

- + Kein Server besitzt genügend Informationen um Schaden anzurichten
- + Sicherheit kommt maßgeblich durch 2PAKE und 2PASS
- Pedersen Commitments sind nur computational binding

## *Performance*

Python Implementierung benötigt für

10-stelliges Passwort 2,76 Sek.

20-stelliges Passwort 6,34 Sek.

# Eigenes Fazit – 2BPR / 2PAKE / 2PASS

---

- Erschreckend wie viele Passwort Datenbanken gehackt wurden
- Erschreckend wie viele User schwache Passwörter verwenden

➔ Klar spezifizierte Protokolle sind äußerst sinnvoll um User zu schützen und um Entwickler bei Ihrer Arbeit zu unterstützen









