# SIT / SRN
# PROJEKTPRÄSENTATION

Vortragende:

David Seemann

Jochen Schwander

Marcel Math

Phil-Patrick Kwiotek

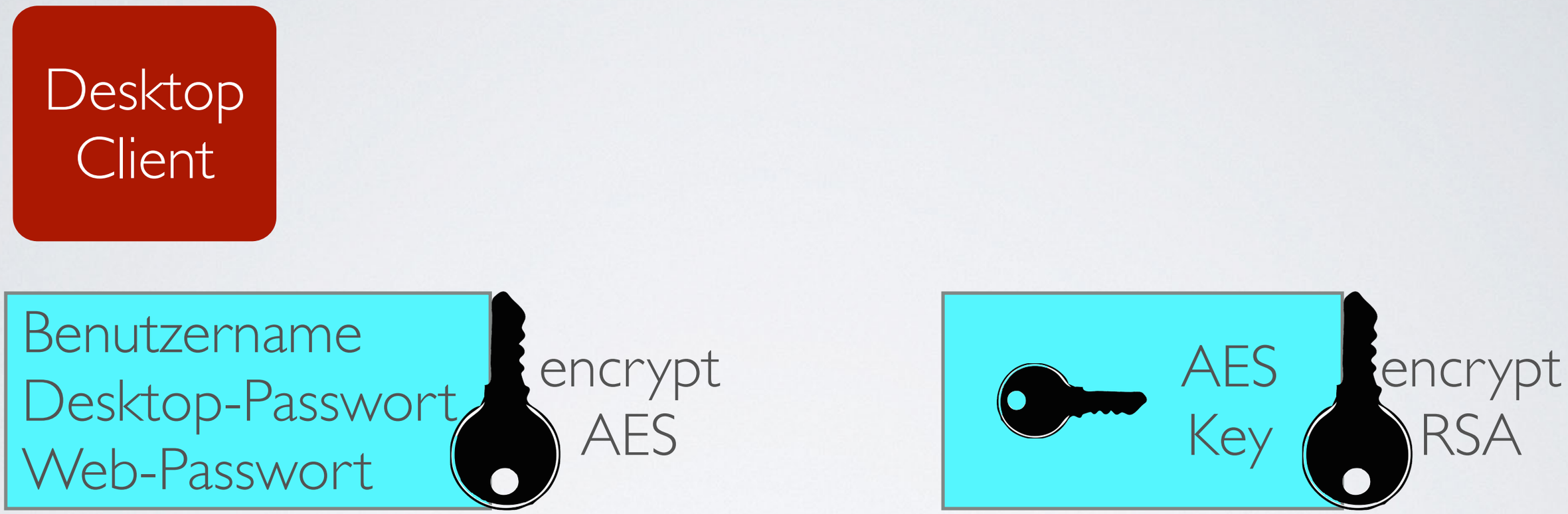
# GLIEDERUNG

- Verschlüsselung
- Registrierung
- Login
- Sicherheitsvorkehrungen
- Live Demo

# —— Verschlüsselung ——

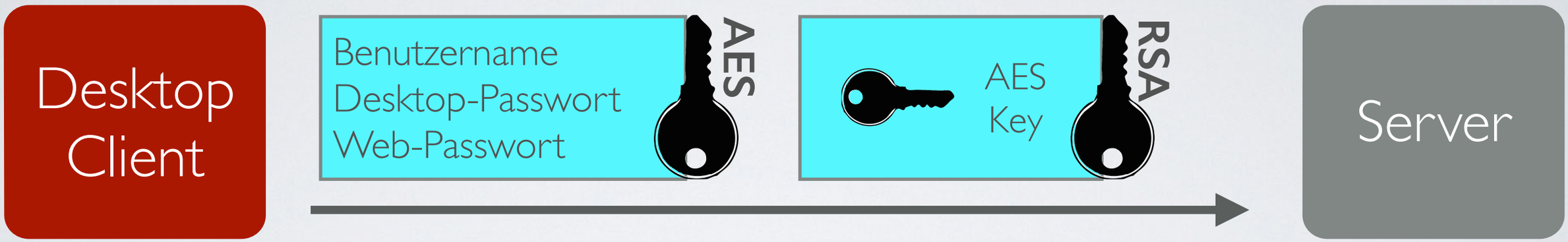
- Hash SHA-512 mit 1000 Iterationen
- RSA mit 4096 Bit
  & AES im CBC-Modus mit 256 Bit Schlüssel
- Stromchiffre: AES im CFB-Modus

# Registrierung




- Der Desktop Client wählt Username und Passwörter und verschlüsselt diese mit AES
- Der AES-Schlüssel wird mit RSA verschlüsselt

# Registrierung




- Desktop Client sendet Passwörter & Benutzername mit Public-Key verschlüsselt an Server

# Registrierung


- Server entschlüsselt Benutzername und Passwörter
- Server prüft ob Benutzername noch verfügbar ist

# Registrierung

Server

Salt

**h(** Web-Passwort **+** Salt **)**

**h(** Desktop-Passwort **+** Salt **)**

- Server generiert Salt
- Server erstellt Salted-Hash von Passwörtern

# Registrierung


- Server legt Benutzer in Datenbank

# Login

Desktop Client

a, g, p

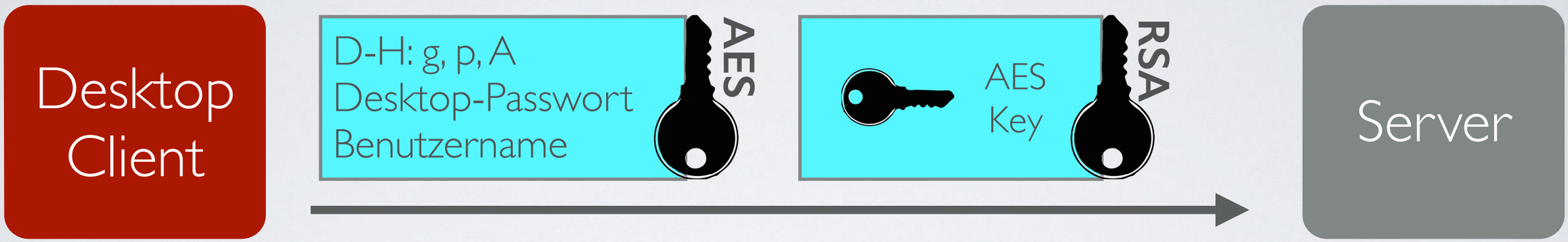
$A = g^a \bmod p$

Desktop-Passwort

Benutzername

- Der Desktop Client beginnt Diffie-Hellman-Schlüsselaustausch

# Login




- Desktop Client sendet D-H-Parameter, Desktop-Passwort & Benutzername mit AES verschlüsselt und den mit dem Public-Key verschlüsselten AES-Key

# Login

Server

Private Key

AES Key

D-H: g, p, A

**Desktop-Passwort**

Benutzername

- Server entschlüsselt D-H-Parameter, Desktop-Passwort & Benutzername

# Login

überprüfe:
**x** ?= **h( Desktop-Passwort + Salt )**

Server

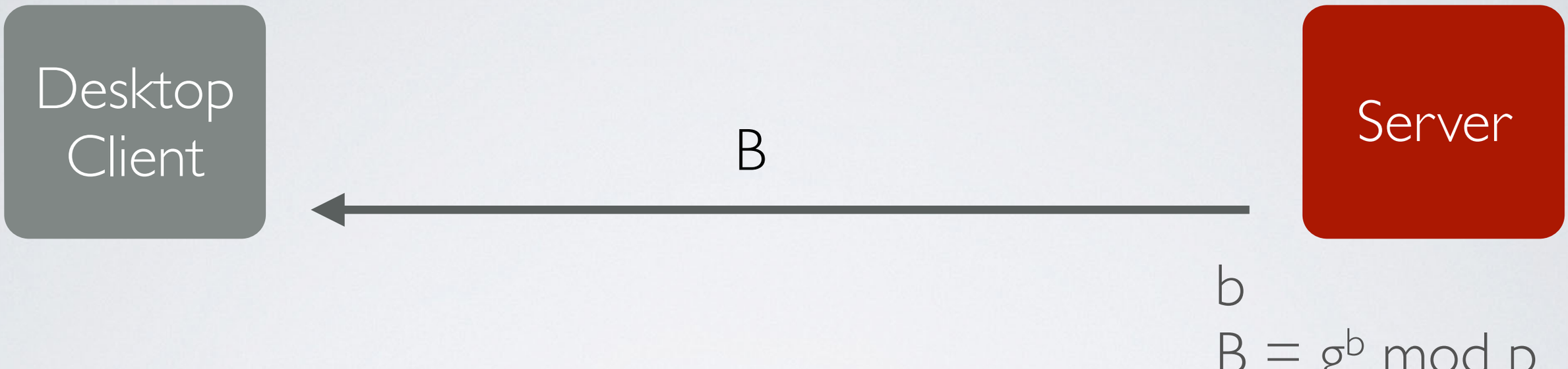
DB

**h(** Desktop-Passwort **+** Salt **)** = **x**

**Salt**

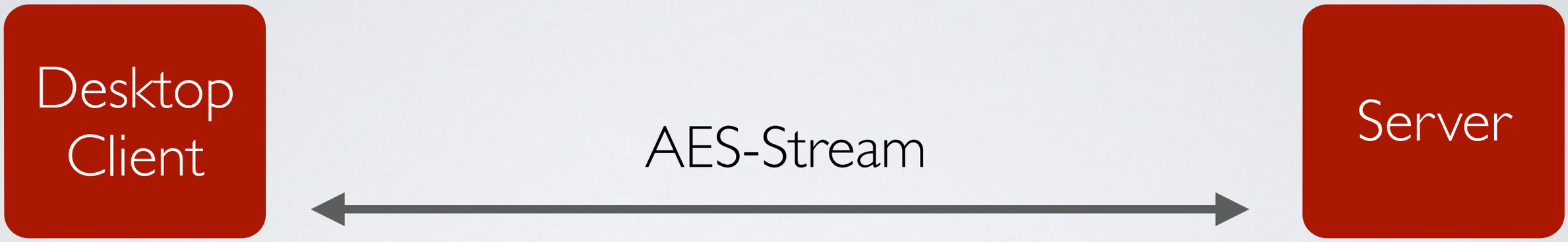
- Sever holt sich Benutzerdaten aus der Datenbank und hinterlegt diese für später.
- Server überprüft ob Hash von Desktop-Passwort dem Datenbankeintrag für diesen User übereinstimmt

# Login




- Server setzt Diffie-Hellman-Schlüsselaustausch fort und sendet Desktop Client sein B

# Login




**K** = $B^a$ mod p
new Stream(AES,**K**)

**K** = $A^b$ mod p
new Stream(AES,**K**)

- Server & Desktop Client berechnen **K**, der geheime Schlüssel
- Server & Desktop Client starten AES-verschlüsselten Stream. **K** dient dabei als geheimer Schlüssel.

# Login


- Server berechnet One Time Password (OTP) und schickt es zusammen mit dem Salt aus der Datenbank an den Desktop Client

# Login


- Desktop Client zeigt Benutzer OTP & Salt an
- Der **Benutzer** fügt OTP & Salt im Web Client ein
- Der **Benutzer** gibt zusätzlich sein Web-Passwort und seinen Benutzernamen ein

# Login

Web Client

**h ( h (** Web-Passwort **+** Salt **) +** OTP**)**

Server

- Der Web Client berechnet den Hash aus dem Web-Passwort und dem Salt
- Der Web Client berechnet den Hash davon und dem OTP
- Der Web Client sendet diesen Hash an den Server

# Login

überprüfe:

**h( x +** OTP**)** ?=

**h( h( Web-Passwort + Salt ) + OTP)**

Server

DB

**h(** Web-Passwort **+** Salt **)** = **x**

**Salt**

- Der Server vergleicht den Hash mit dem Hash aus Datenbank und OTP

# Login

Der Benutzer ist vollständig authentifiziert!

# Sicherheitsvorkehrungen
## **Man In The Middle**

Ein **Man in the Middle Angriff** ist nur möglich, wenn der Public-Key des Servers ausgetauscht wird oder der Private-Key des Servers kompromittiert ist.

# Sicherheitsvorkehrungen
## Brute Force

**Brute Force Angriffe** sind nicht möglich, da der Server nur je 3 Login-Versuche zulässt und danach der User sperrt.

Der User muss danach z.B. über Telefon / den Postweg neu aktiviert werden.

# Sicherheitsvorkehrungen
## SQL Injection

**SQL Injections** werden durch Prepared Statements und Whitelisting unterbunden.

# Sicherheitsvorkehrungen
## **OneTimePasswort**

Das **OTP** ist nur **einmal** gültig und verfällt nach 5 Minuten.
Außerdem sorgt der Hash des Web-Passwortes mit dem OTP dafür, dass das übertragene Web-Passwort immer einmalig über die unsichere Leitung *(Web-Client zu Server)* geschickt wird.

# Live Demo

nun folgt die Live Demo