Strong Authentication without Tamper-Resistant Hardware and Application to Federated Identities

Zhenfeng Zhang[†,⚛], Yuchen Wang[†], Kang Yang[✧]

[†] Institute of Software, Chinese Academy of Sciences;
[⚛] The Joint Academy of Blockchain Innovation;
[✧] State Key Laboratory of Cryptology

Presenter: Long Chen (New Jersey Institute of Technology)

Shared Credential Authentication

- Mechanism has dominated the realm of authentication for decades
  - e.g., password (weak authentication)
  - User's credentials stored in centralized repositories at servers
  - Explicitly transferred from user to server
- The shared credentials can be stolen in batches or captured
  - From breached centralized repositories
  - Through phishing attacks

Strong Authentication

- Strong authentication — cryptographic identification protocol
  - A claimant proves its identity to a verifier via challenge-response
  - The claimant demonstrates the knowledge of secret keys with crypto
  - Secret keys are not transferred over the channels, eliminate the risks
- Mechanisms can be built with symmetric-key/public-key cryptos
  - The claimant generates a MAC value on a challenge with a secret-key
  - The claimant digitally signs a challenge message with a private-key
  - e.g., HMAC and ECDSA algorithms

How to Store Secret-keys for Strong Authentication?

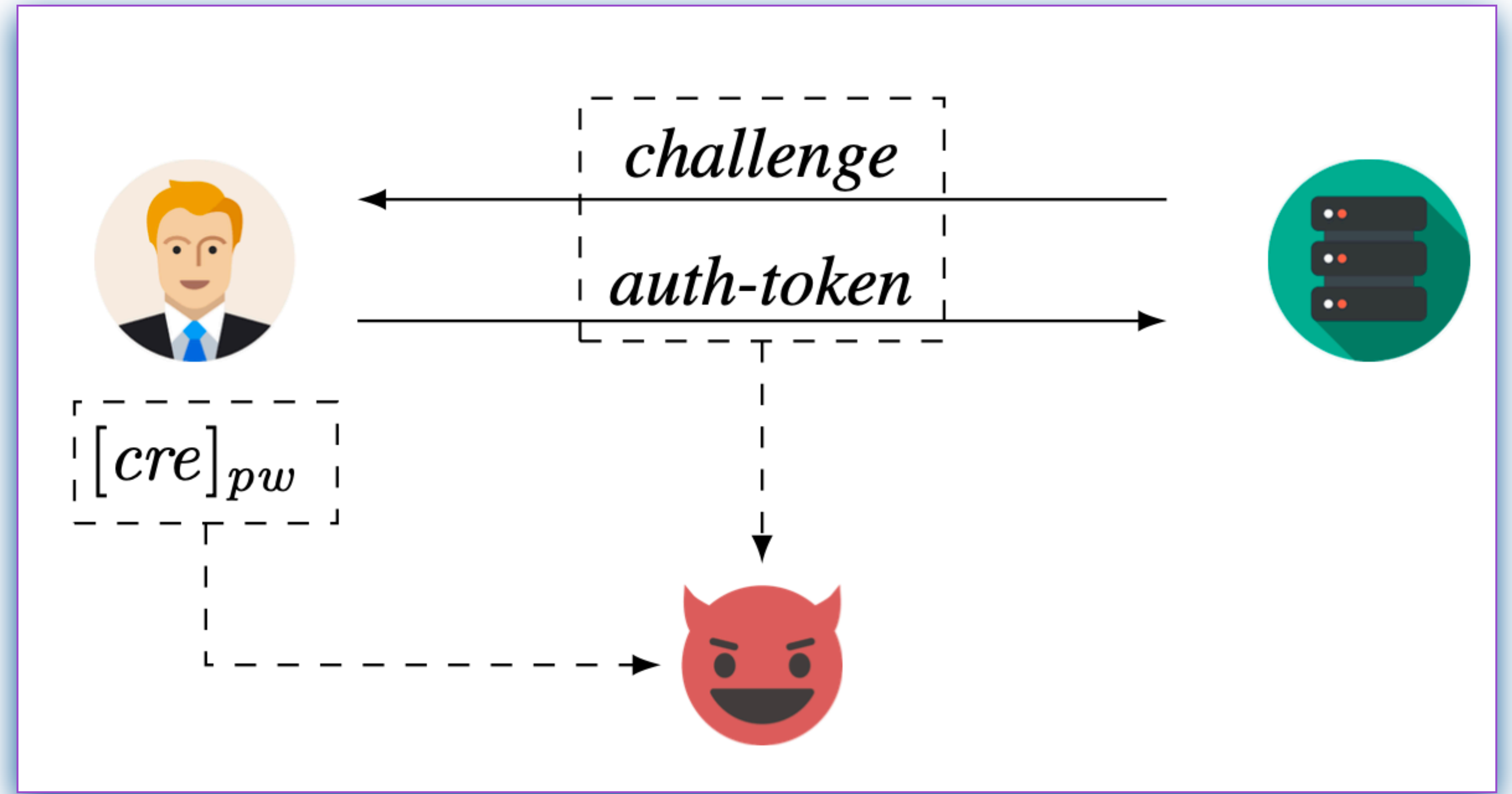
- Tamper-resistant hardware modules
  - Highly recommended by FIDO and W3C
  - FIDO Universal Authentication Framework
  - W3C Web Authentication Specification

- The issues with a tamper-resistant hardware module
  - The module becomes another thing to be remembered to carry
  - The secret would lost if the module/device is broken or lost
  - Decrease usability of the strong authentication scheme

How to Store Secret-keys for Strong Authentication?

- Model for strong-auth without tamper-resistant hardware modules

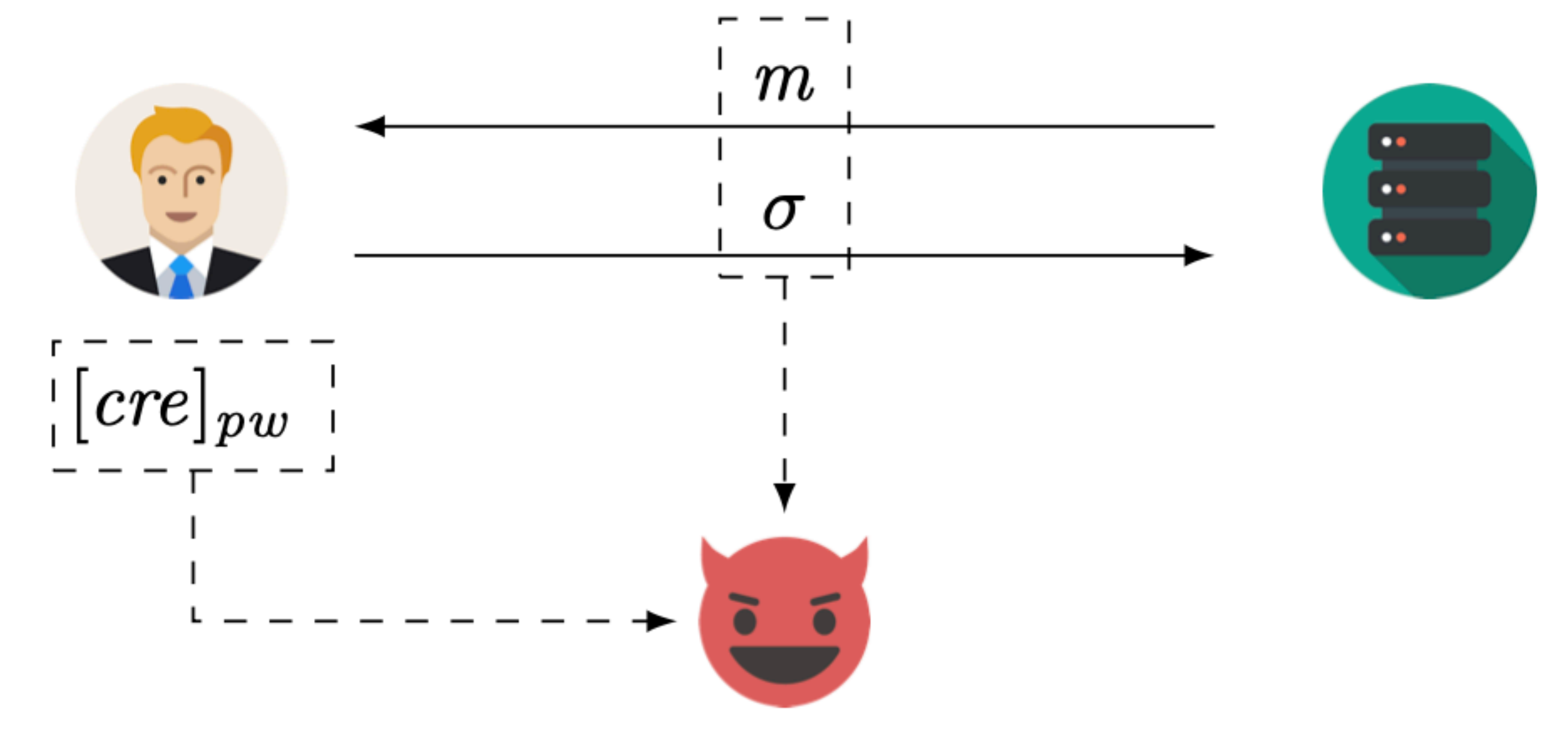
- The adversary's capabilities
  - Obtain PW-wrapped credentials
  - Capture authentication tokens



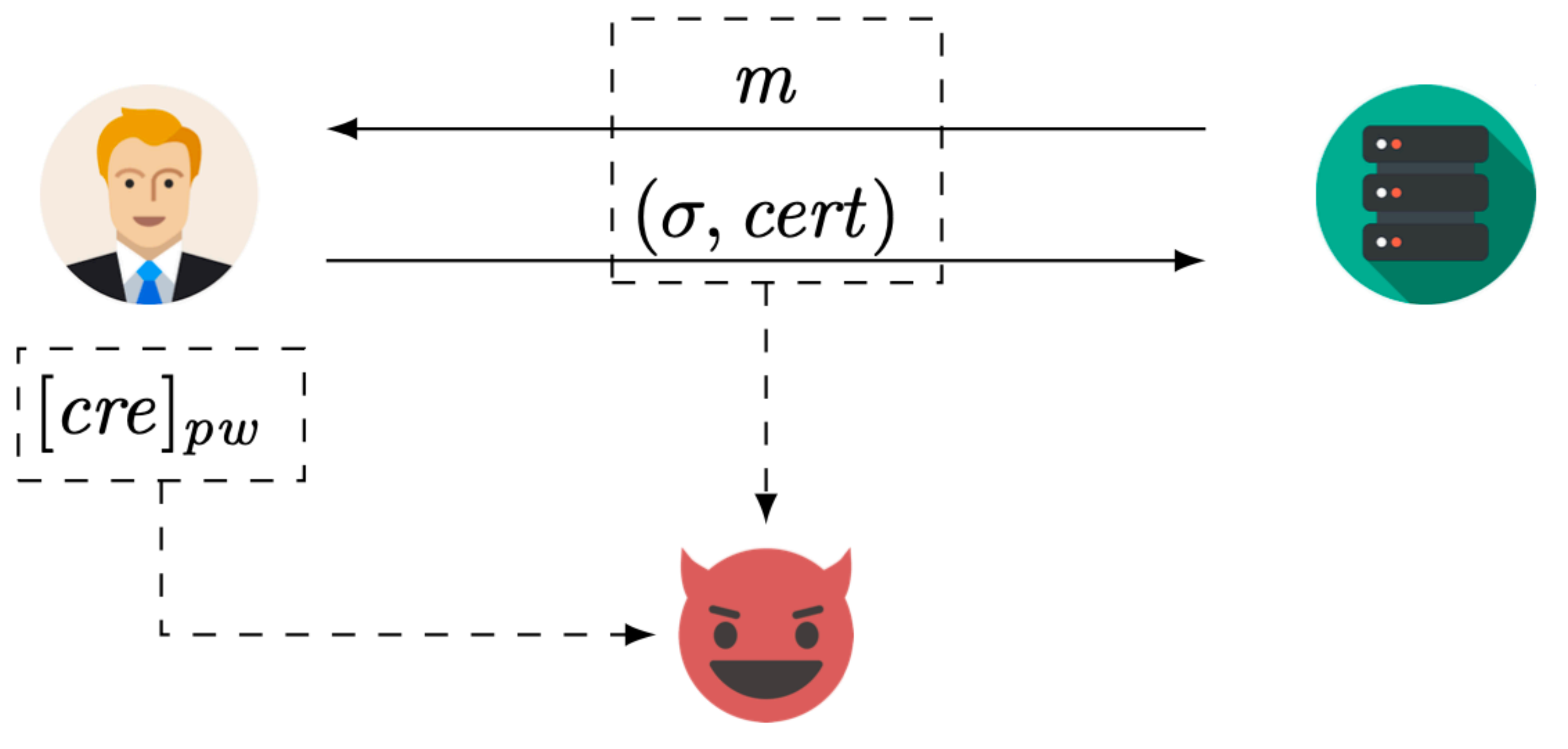
- The security goals
  - Off-line dictionary attacks are infeasible
  - Existential forgery of an authentication token is infeasible

How to Store Secret-keys for Strong Authentication?

- Off-line attacks under the model against strong authentication with symmetric-key crypto (MAC) / public-key crypto (DSA)



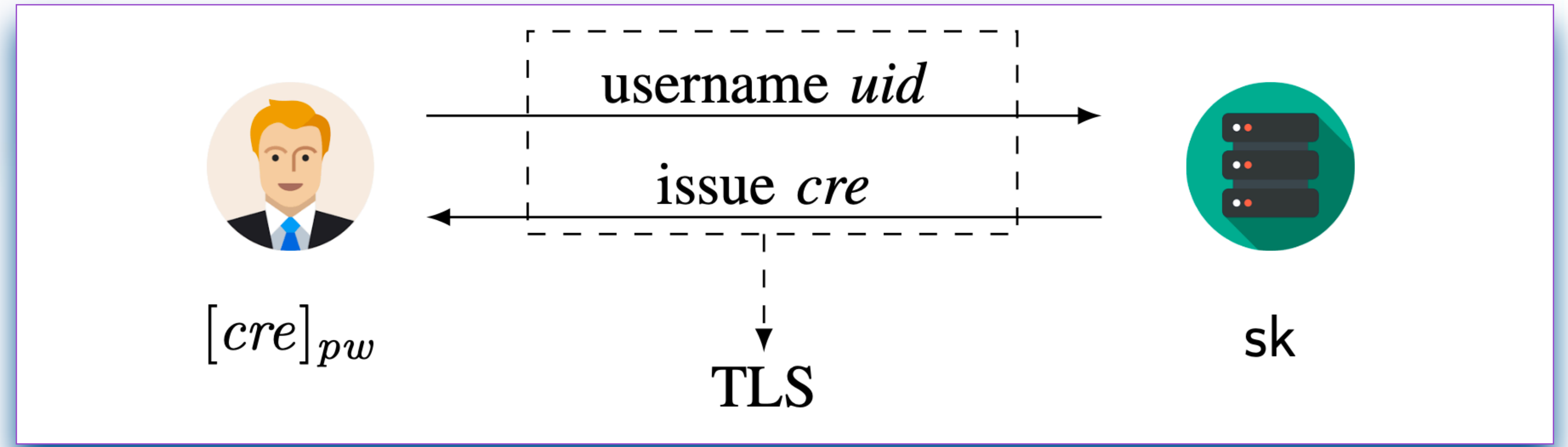
$cre' \leftarrow \mathsf{Dec}(pw', [cre]_{pw})$
check $\mathsf{MAC}(cre', m) = \sigma$



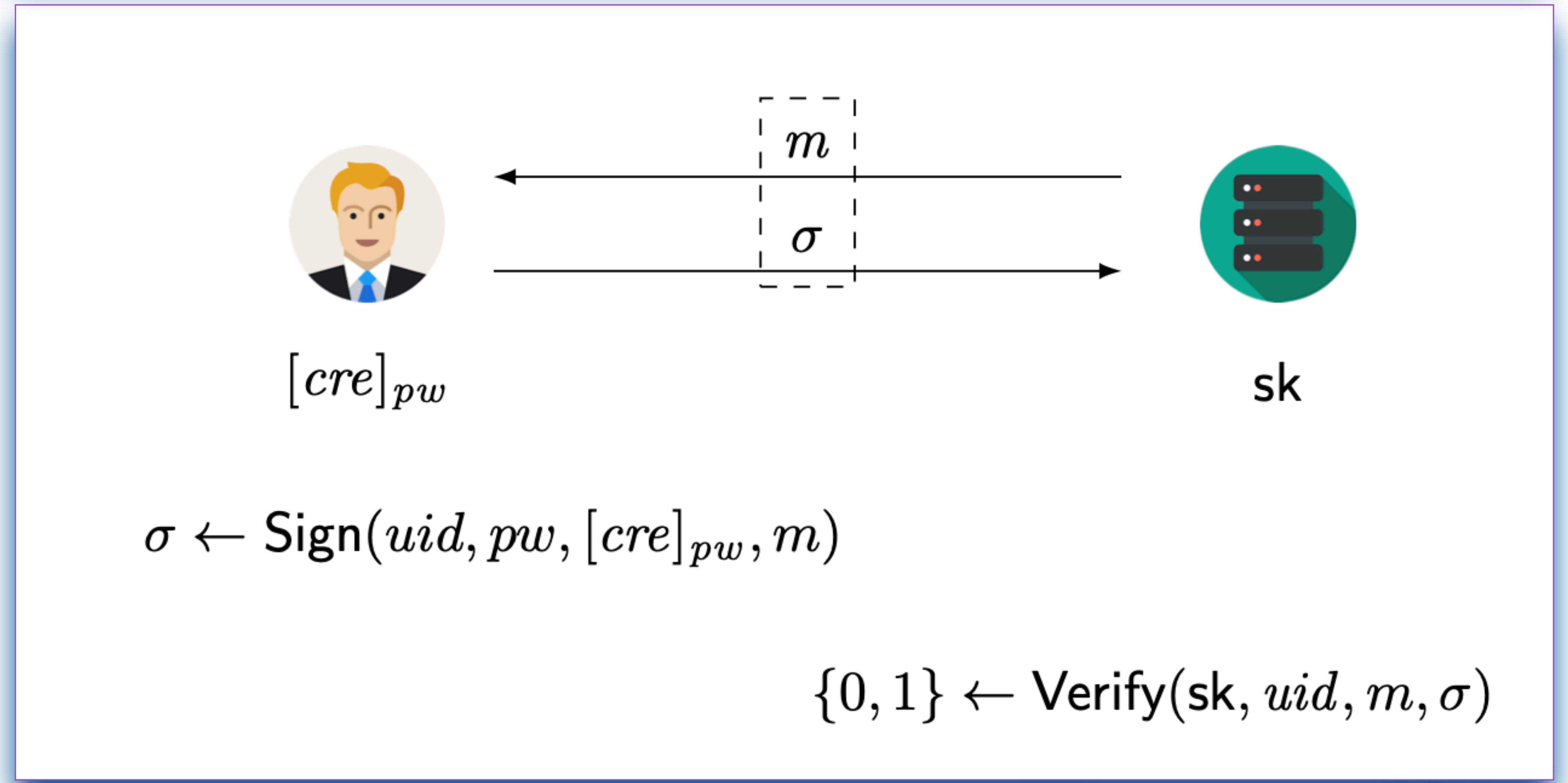
$cre' \leftarrow \mathsf{Dec}(pw', [cre]_{pw})$, $pk \leftarrow cert$,
check $\mathsf{VerifyKey}(pk, cre') = 1$

Strong Authentication with Password-based Credentials

- The Registration Phase



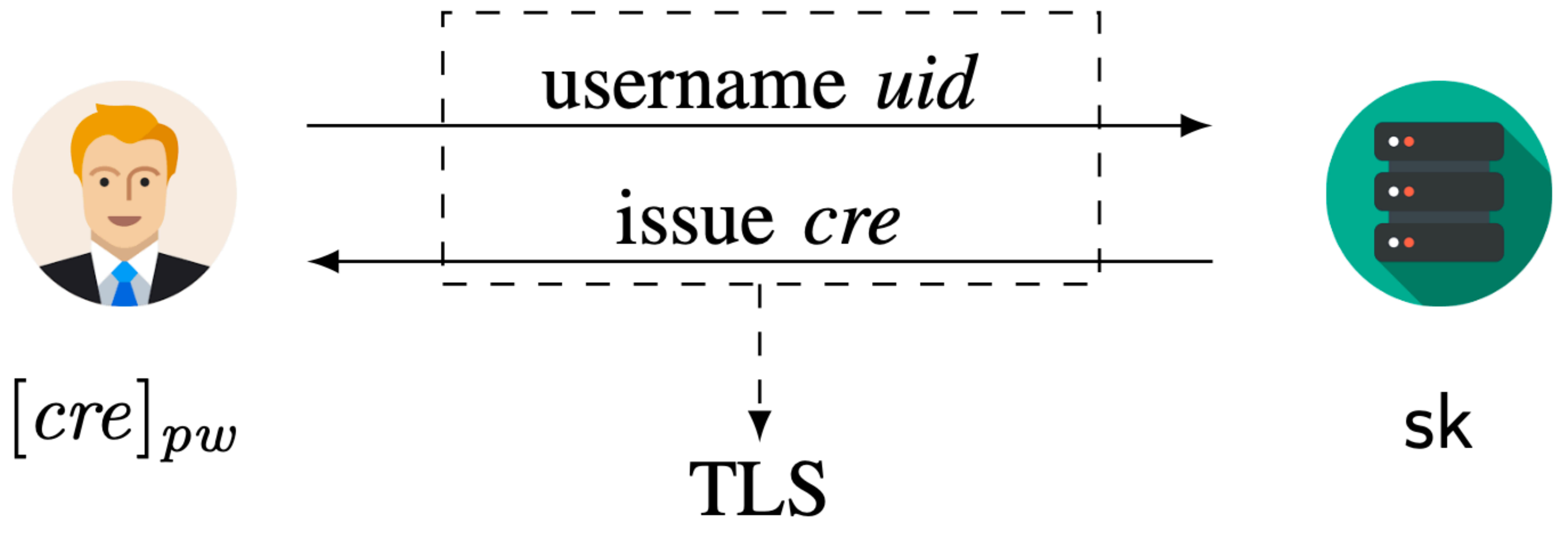
- The Authentication Phase



- The Secure Construction of Password-based Credential

Password-based Credentials

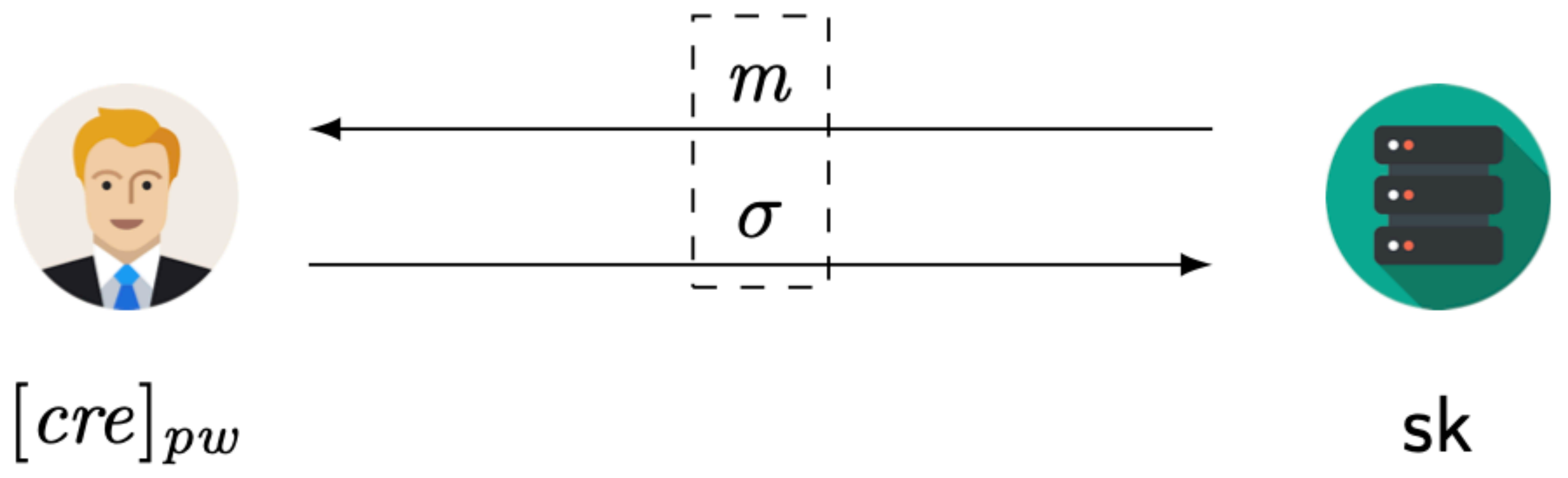
- Setup algorithm
- Key Generation algorithm
- Issue algorithm
- The Sign Algorithm
- The Verify Algorithm



- Setup(1^λ): The algorithm chooses a set of group parameters $(\mathbb{G}, p, g)$ with p a 2λ-bit prime, outputs $\mathsf{pp} = (\mathbb{G}, p, g)$.
- KeyGen(pp): It picks $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $w \leftarrow g^\gamma$. The server sets $\mathsf{sk} \leftarrow \gamma$ and $Reg \leftarrow \emptyset$, publishes $\mathsf{isp} \leftarrow w$.
- Issue$(\gamma, Reg) \rightleftharpoons (uid, pw)$ is executed between the server and a user over TLS.
  1) The server aborts if $uid \in Reg$. Otherwise it computes $cre \leftarrow A = g^{1/(\gamma+uid)}$ and sends cre to the user.
  2) The user encrypts its credential $cre = A$ by computing $[A]_{pw} \leftarrow A \cdot H_{\mathbb{G}}(pw)$, and then stores $[A]_{pw}$.

Password-based Credentials

- Setup algorithm
- Key Generation algorithm
- Issue algorithm
- The Sign Algorithm
  - randomize-then-prove
  - SPK can be standardized signature algorithms [ISO/IEC 14888-3:2018]
- The Verify Algorithm



$$\sigma \leftarrow \mathsf{Sign}(uid, pw, [cre]_{pw}, m)$$

$$\{0,1\} \leftarrow \mathsf{Verify}(\mathsf{sk}, uid, m, \sigma)$$

- $\mathsf{Sign}(uid, pw, [A]_{pw}, m)$: the algorithm decrypts $[A]_{pw}$ by computing $A \leftarrow [A]_{pw}/H_{\mathbb{G}}(pw)$. Then, it chooses $a \xleftarrow{\$} \mathbb{Z}_p^*$ and randomizes A as $T \leftarrow A^a$, and generates a signature proof of knowledge w.r.t T as

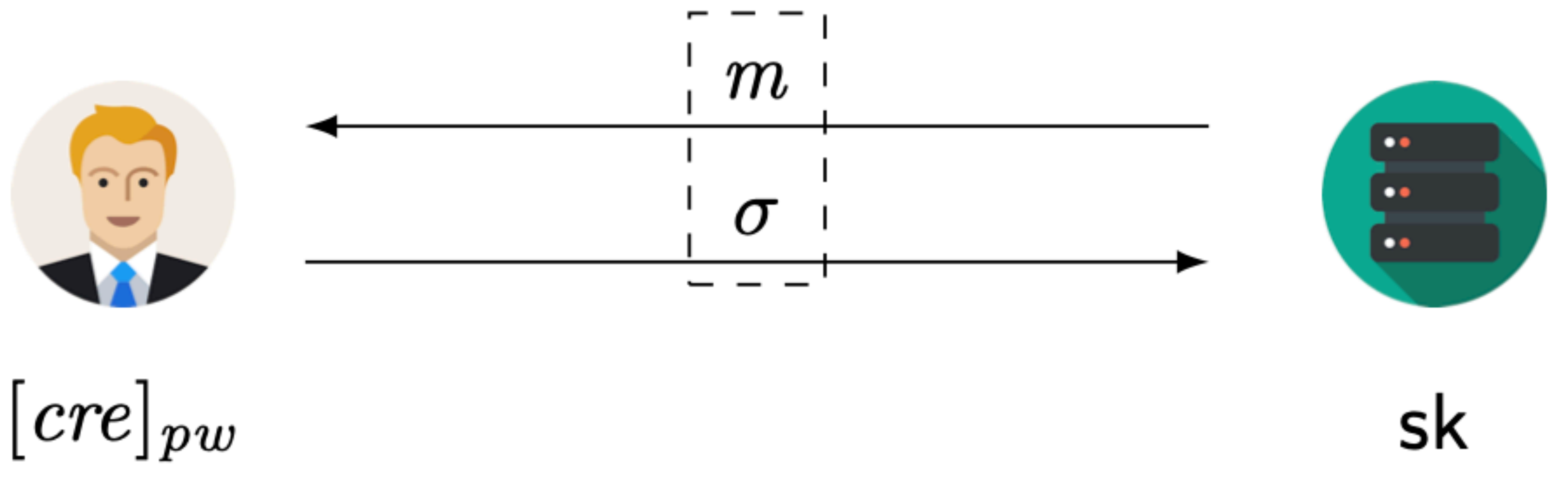
$$\pi_T \leftarrow \mathsf{SPK}\{(a) : g^a = PK\}(m).$$

Finally, it outputs an authentication token $\sigma \leftarrow (T, \pi_T)$.

Password-based Credentials

- Setup algorithm
- Key Generation algorithm
- Issue algorithm

- The Sign Algorithm
  - randomize-then-prove
  - SPK can be standardized signature algorithms [ISO/IEC 14888-3:2018]
- The Verify Algorithm



$\sigma \leftarrow \mathsf{Sign}(uid, pw, [cre]_{pw}, m)$

$\{0,1\} \leftarrow \mathsf{Verify}(\mathsf{sk}, uid, m, \sigma)$

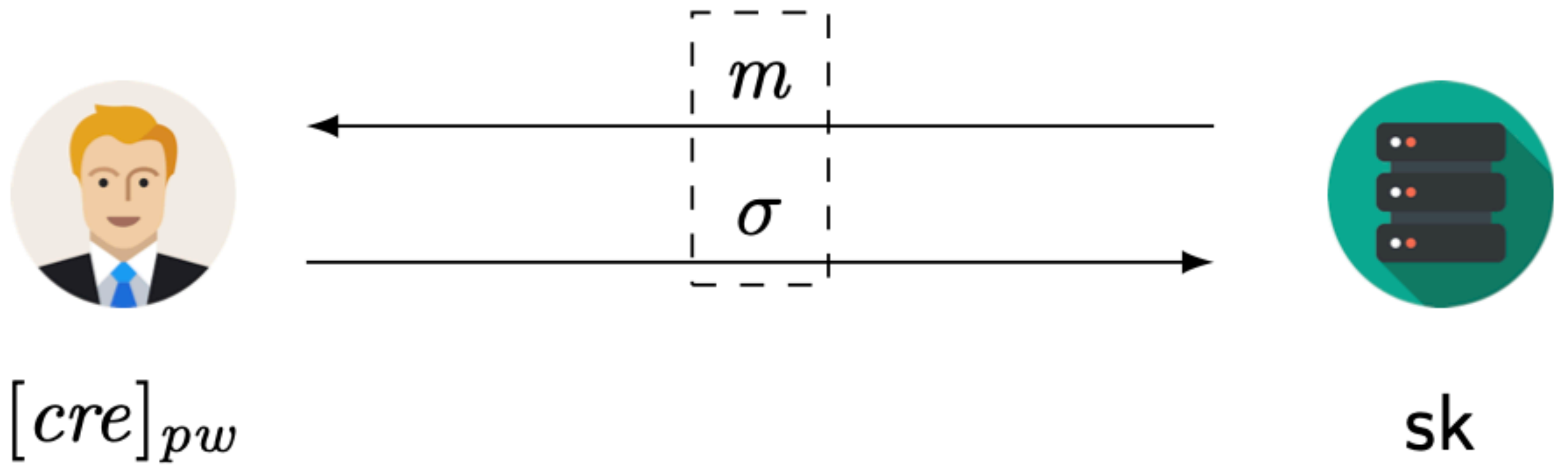
- $\mathsf{Sign}(uid, pw, [A]_{pw}, m)$: the algorithm decrypts $[A]_{pw}$ by computing $A \leftarrow [A]_{pw}/H_{\mathbb{G}}(pw)$. Then, it chooses $a \xleftarrow{\$} \mathbb{Z}_p^*$ and randomizes A as $T \leftarrow A^a$, and generates a signature proof of knowledge w.r.t T as

$$\pi_T \leftarrow \mathsf{SPK}\left\{(a) : g^a = PK\right\}(m).$$

Finally, it outputs an authentication token $\sigma \leftarrow (T, \pi_T)$.

Password-based Credentials

- Setup algorithm
- Key Generation algorithm
- Issue algorithm
- The Sign Algorithm
- The Verify Algorithm



$\sigma \leftarrow \mathsf{Sign}(uid, pw, [cre]_{pw}, m)$

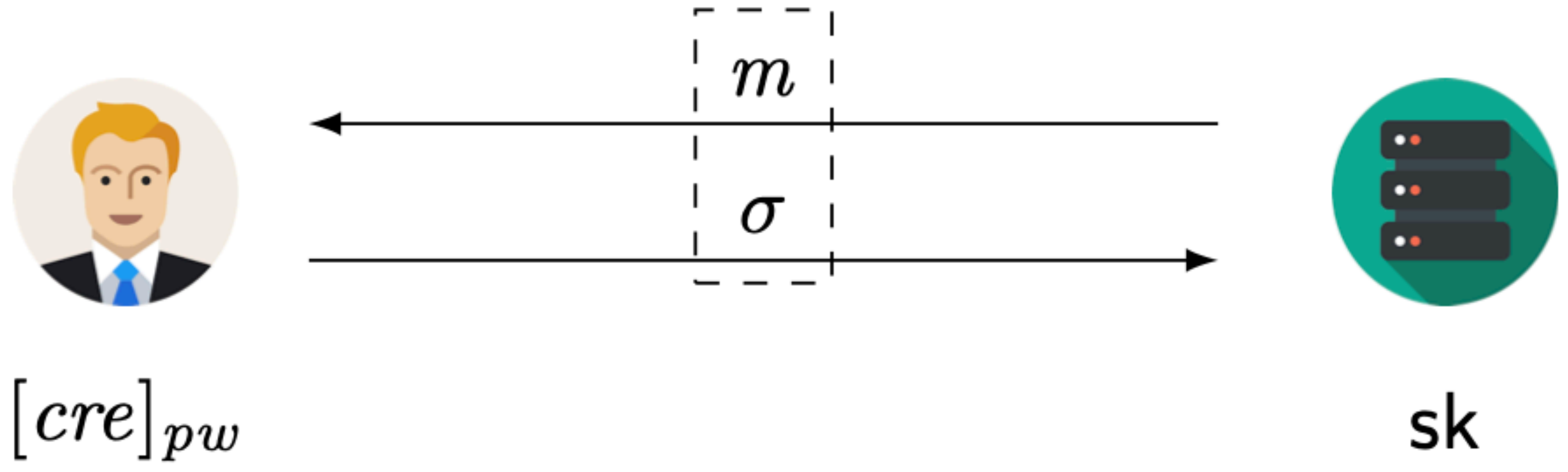
$\{0,1\} \leftarrow \mathsf{Verify}(\mathsf{sk}, uid, m, \sigma)$

- $\mathsf{Verify}(\gamma, uid, m, \sigma)$: the algorithm parses σ as (T, π_T) and computes $PK = T^{\gamma+uid}$, if $T \neq 1$. It then returns the outputs of $\mathsf{Verify}_{\mathsf{SPK}}\ ((g, PK), m, \pi_T)$.

Note that $T^{\gamma+uid} = g^a$, hence the claimer who has the secret a also holds T^{-a}, which has the form $g^{1/(\gamma+uid)}$.

Password-based Credentials

- Setup algorithm
- Key Generation algorithm
- Issue algorithm
- The Sign Algorithm
- The Verify Algorithm



$\sigma \leftarrow \mathsf{Sign}(uid, pw, [cre]_{pw}, m)$

$\{0,1\} \leftarrow \mathsf{Verify}(\mathsf{sk}, uid, m, \sigma)$

- $\mathsf{Verify}(\gamma, uid, m, \sigma)$: the algorithm parses σ as (T, π_T) and computes $PK = T^{\gamma+uid}$, if $T \neq 1$. It then returns the outputs of $\mathsf{Verify}_{\mathsf{SPK}}\ ((g, PK), m, \pi_T)$.

  Note that $T^{\gamma+uid} = g^a$, hence the claimer who has the secret a also holds T^{-a}, which has the form $g^{1/(\gamma+uid)}$.

Password-based Credentials

Security Model of PBC and Provable Security

Experiment $\mathsf{Exp}^{\text{EUF-CMVA}}_{\mathsf{PBC}}(\mathcal{A})$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{sk}, \mathsf{isp}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$.

$RUpw, RUcred, Q \leftarrow \emptyset$.

For each $i \in [n]$, $pw_i \xleftarrow{\$} \mathcal{D}$, and

$[cre_i]_{pw_i} \leftarrow \mathsf{Issue}(\mathsf{sk}, Reg) \rightleftharpoons (uid_i, pw_i)$.

$(uid^*, m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{isp}, \{uid_i\}_{i=1}^n, \textsc{Sign}, \textsc{Verify}, \textsc{RevealPW}, \textsc{RevealCred})$.

If $\mathsf{Verify}(\mathsf{sk}, uid^*, m^*, \sigma^*) = 0$, return 0.

If $uid^* \notin Reg$, return 1.

If $uid^* = uid_{i^*} \in Reg$, then

- If $(i^*, m^*) \in Q$, return 0.
- If $i^* \in RUpw \cap RUcred$, return 0.
- If $i^* \notin RUcred$, return 1.
- If $i^* \in RUcred \wedge i^* \notin RUpw$, return 2.

Theorem 1: Let $\mathcal{A}$ be an adversary against the sEUF-CMVA security of PBC scheme Π_{PBC} who runs in time t, and makes q_s queries to the $\textsc{Sign}$ oracle and q_v queries to the $\textsc{Verify}$ oracle. Then, we have:

$$\mathsf{Adv}^{sEUF\text{-}CMVA}_{\Pi_{\text{PBC}},case\text{-}1}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{SPK}}(t', q_s, q_v) + (q_v + 1)(\mathsf{Adv}^{\mathsf{SDH}}_{\mathbb{G}}(t'', n+1) + n\mathsf{Adv}^{\mathsf{SDH}}_{\mathbb{G}}(t'', n)),$$

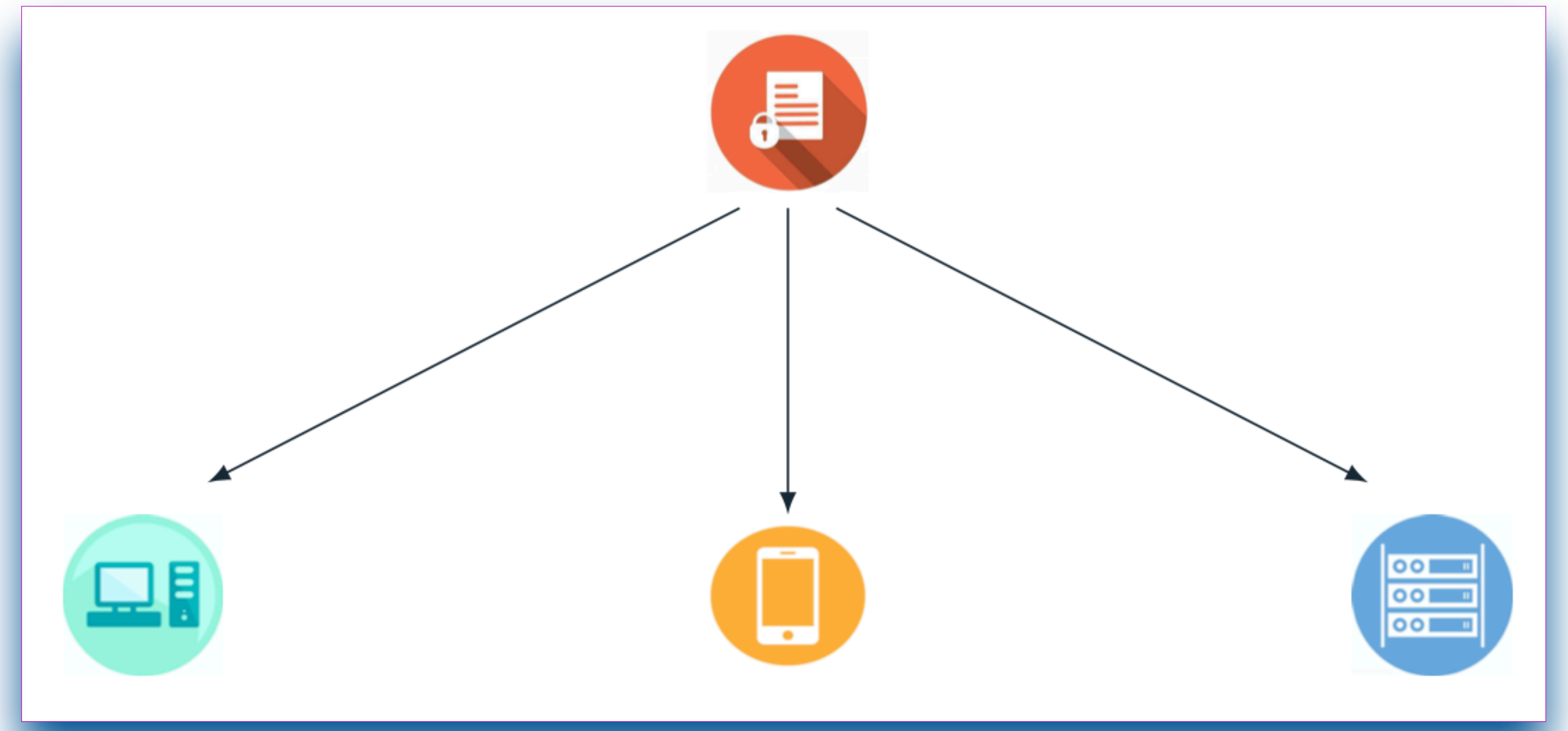
$$\mathsf{Adv}^{sEUF\text{-}CMVA}_{\Pi_{\text{PBC}},case\text{-}2}(\mathcal{A}) \leq \frac{q_v}{|\mathcal{D}|} + \mathsf{Adv}_{\mathsf{SPK}}(t', q_s, q_v) + (q_v + 1)\mathsf{Adv}^{\mathsf{SDH}}_{\mathbb{G}}(t'', n+1) + n\mathsf{Adv}^{\mathsf{DDHI}}_{\mathbb{G}}(t'', n),$$

where $\mathsf{Adv}_{\mathsf{SPK}}(t', q_s, q_v) = \mathcal{O}(\mathsf{Adv}^{uzk}_{\mathsf{SPK}}(t', q_s) + \mathsf{Adv}^{ss-ext}_{\mathsf{SPK}}(t', q_s, q_v))$, $t' = t + \mathcal{O}((q_s + q_v)t_{exp})$, $t'' = \mathcal{O}(t' + n^2 t_{exp})$, and t_{exp} denotes the time for one exponentiation.

Strong Authentication with Password-based Credentials

Implementation of PBC-based strong authentication

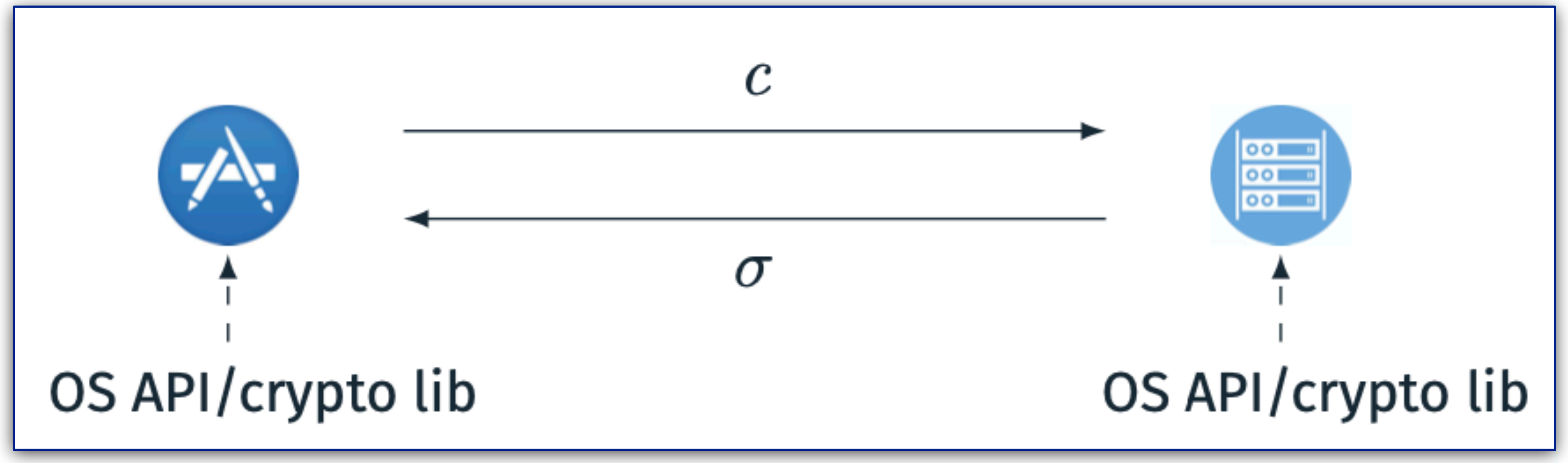
- Common cryptographic libraries
  - Standardized elliptic curves, not require pairing-friendly curves
  - OpenSSL, Bouncy Castle, sjcl,…
- Mainstream programming language, e.g., C/C++, Java, JavaScript,…
- Across devices, e.g., mobile and desktop
- PBC-backup for devices broken or lost
  - Cross device backup
  - Cloud server backup



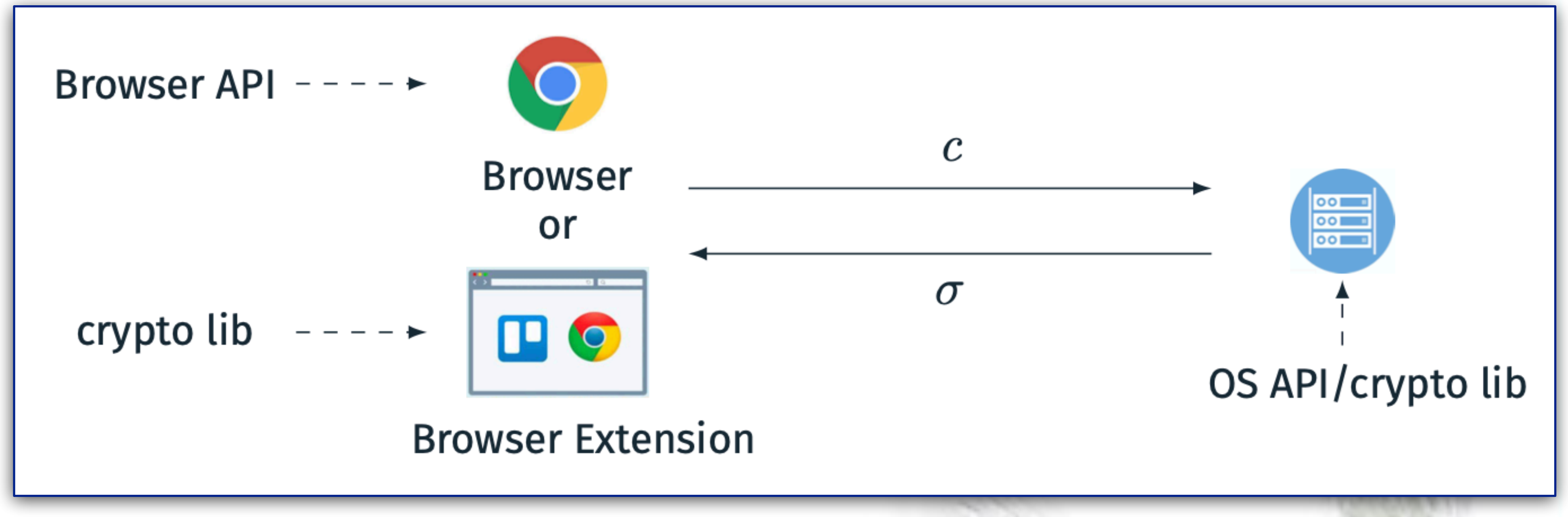
Strong Authentication with Password-based Credentials

Deployment of PBC-based authenticator and AUTH

- PBC authenticators deployed with
  - OS API (e.g., Android's Keystore)
  - Browser API (e.g., W3C's AuthAPIs)



- PBC-AUTH for both C/S and B/S architecture
  - Server (Protect key with hardware)
  - Client (i.e., Application)
  - Browser Extension



Federated Identities with Password-based Credentials

- Identity federation: SAML 2, OAUTH 2.0, OpenID Connect
  - FAL-3: holder-of-key assertion (HoKA), a reference to a key held by a user, RP requires the user to prove possession of the key (PoPK)
- Holder-of-key assertion mechanisms via certificates
  - Require tamper-resistant hardwares to protect the private keys
  - IdP cannot both preserve the privacy of users and support HoKA
- Holder-of-key assertion mechanisms via PBCs
  - Without requirement of tamper-resistant hardware for users
  - Support privacy-preserving HoKA and PoPK

Federated Identities with Password-based Credentials

Holder-of-Key Assertion & Proof-of-Possession of Key with PBCs

- *User-IdP Authentication.* The user authenticates to IdP with a valid authentication token $\sigma = (T, \pi_T)$ s.t.

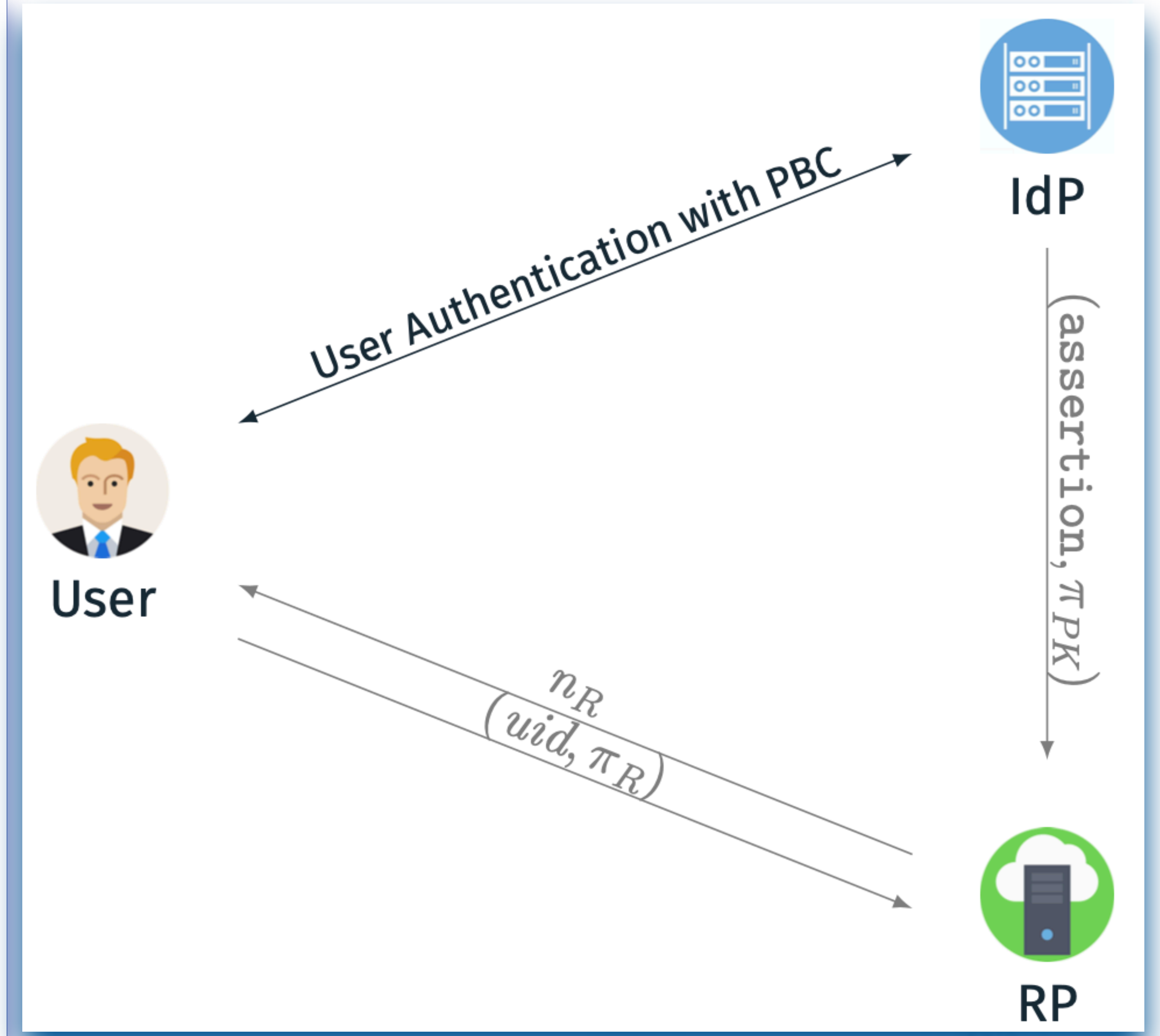
$$\pi_T \leftarrow \mathsf{SPK}\{(a) : g^a = PK\}(m) \text{ for } PK = T^{\gamma+uid}$$

- *Holder-of-Key Assertion.* IdP calculates $PK = T^{\gamma+uid}$, and sets assertion $\leftarrow (T, PK)$. Then, it signs assertion by generating a signature proof of knowledge:

$$\pi_{PK} = \mathsf{SPK}'\{(\gamma) : w = g^{\gamma} \wedge T^{-uid} \cdot PK = T^{\gamma}\}(\cdot)$$

- *Proof-of-Possession of Key.* The user generates a proof-of-possession of the private-key a w.r.t to PK, by calculating

$$\pi_R \leftarrow \mathsf{SPK}\{(a) : g^a = PK\}(n_R)$$



Federated Identities with Password-based Credentials

Holder-of-Key Assertion & Proof-of-Possession of Key with PBCs

- *User-IdP Authentication.* The user authenticates to IdP with a valid authentication token $\sigma = (T, \pi_T)$ s.t.

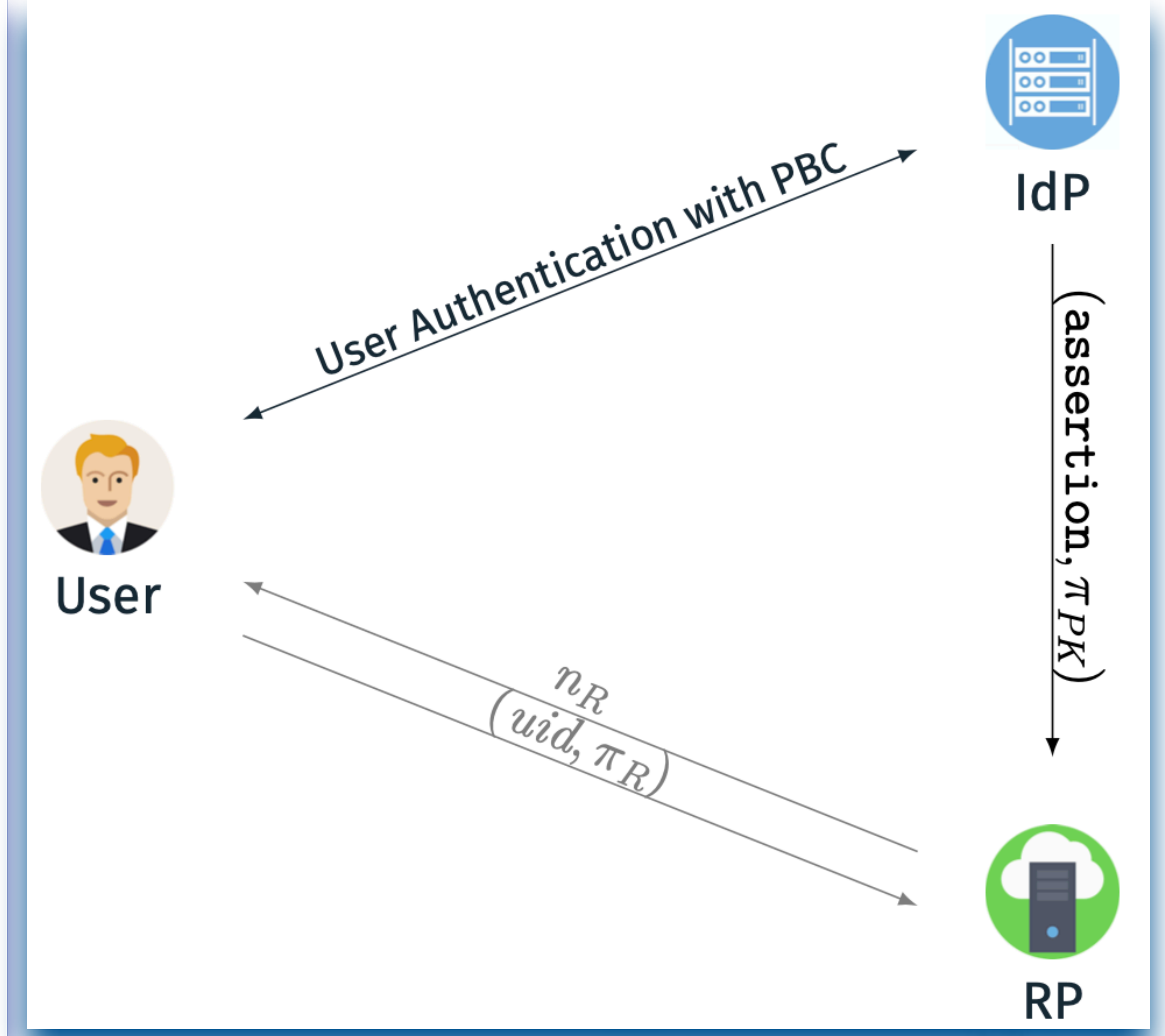
$$\pi_T \leftarrow \mathsf{SPK}\{(a) : g^a = PK\}(m) \text{ for } PK = T^{\gamma+uid}$$

- *Holder-of-Key Assertion.* IdP calculates $PK = T^{\gamma+uid}$, and sets assertion $\leftarrow (T, PK)$. Then, it signs assertion by generating a signature proof of knowledge:

$$\pi_{PK} = \mathsf{SPK}'\{(\gamma) : w = g^\gamma \wedge T^{-uid} \cdot PK = T^\gamma\}(\cdot)$$

- *Proof-of-Possession of Key.* The user generates a proof-of-possession of the private-key a w.r.t to PK, by calculating

$$\pi_R \leftarrow \mathsf{SPK}\{(a) : g^a = PK\}(n_R)$$



Federated Identities with Password-based Credentials

Holder-of-Key Assertion & Proof-of-Possession of Key with PBCs

- *User-IdP Authentication*. The user authenticates to IdP with a valid authentication token $\sigma = (T, \pi_T)$ s.t.

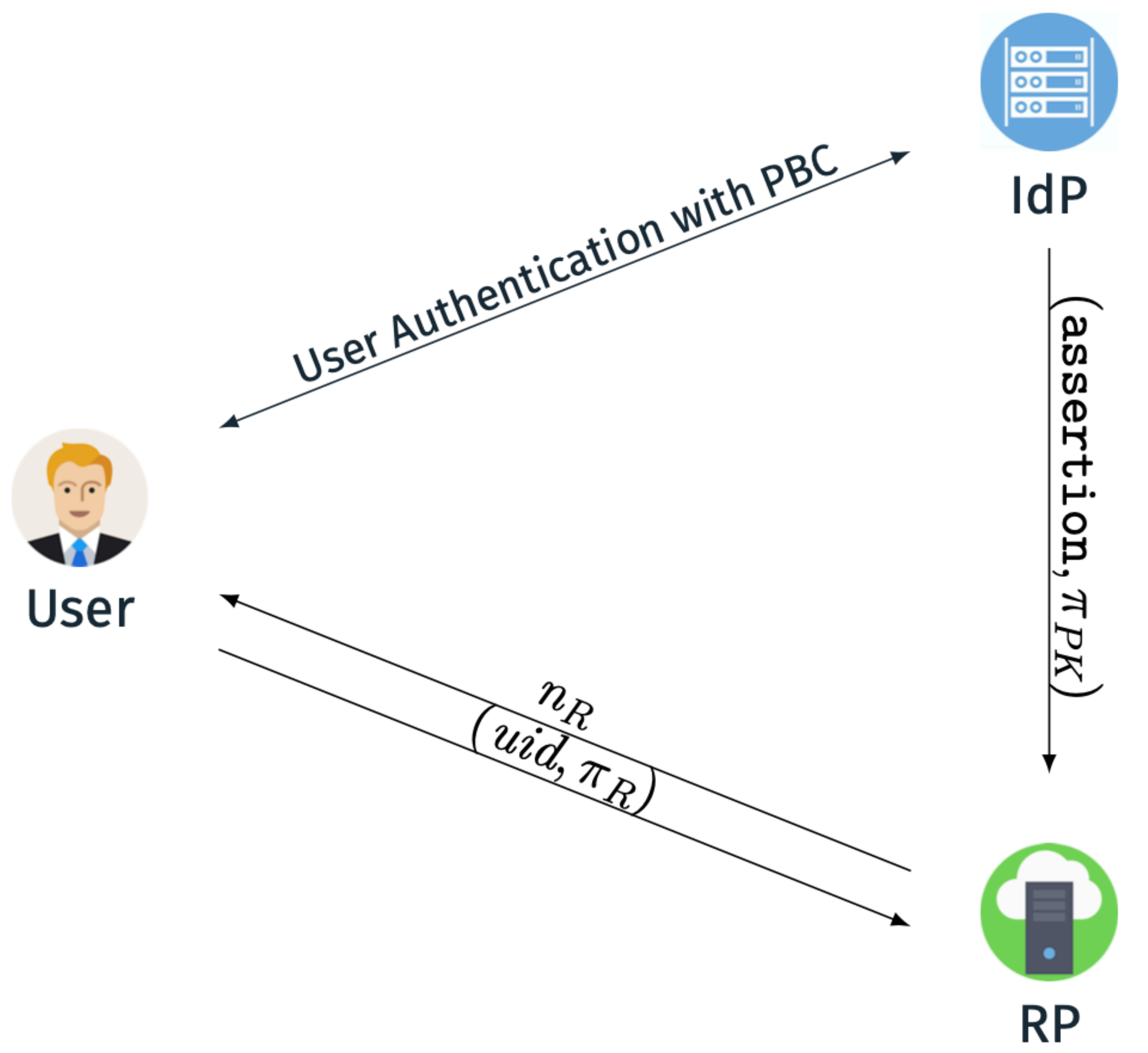
$$\pi_T \leftarrow \mathsf{SPK}\left\{(a) : g^a = PK\right\}(m) \text{ for } PK = T^{\gamma+uid}$$

- *Holder-of-Key Assertion*. IdP calculates $PK = T^{\gamma+uid}$, and sets assertion $\leftarrow (T, PK)$. Then, it signs assertion by generating a signature proof of knowledge:

$$\pi_{PK} = \mathsf{SPK}'\left\{(\gamma) : w = g^{\gamma} \wedge T^{-uid} \cdot PK = T^{\gamma}\right\}(\cdot)$$

- *Proof-of-Possession of Key*. The user generates a proof-of-possession of the private-key a w.r.t to PK, by calculating

$$\pi_R \leftarrow \mathsf{SPK}\{(a) : g^a = PK\}(n_R)$$



Federated Identities with Password-based Credentials

Privacy-Preserving Holder-of-Key Assertion & PoPK with PBCs

- *User-IdP Authentication.* The user authenticates to IdP with a valid authentication token $\sigma = (T, \pi_T)$ s.t.

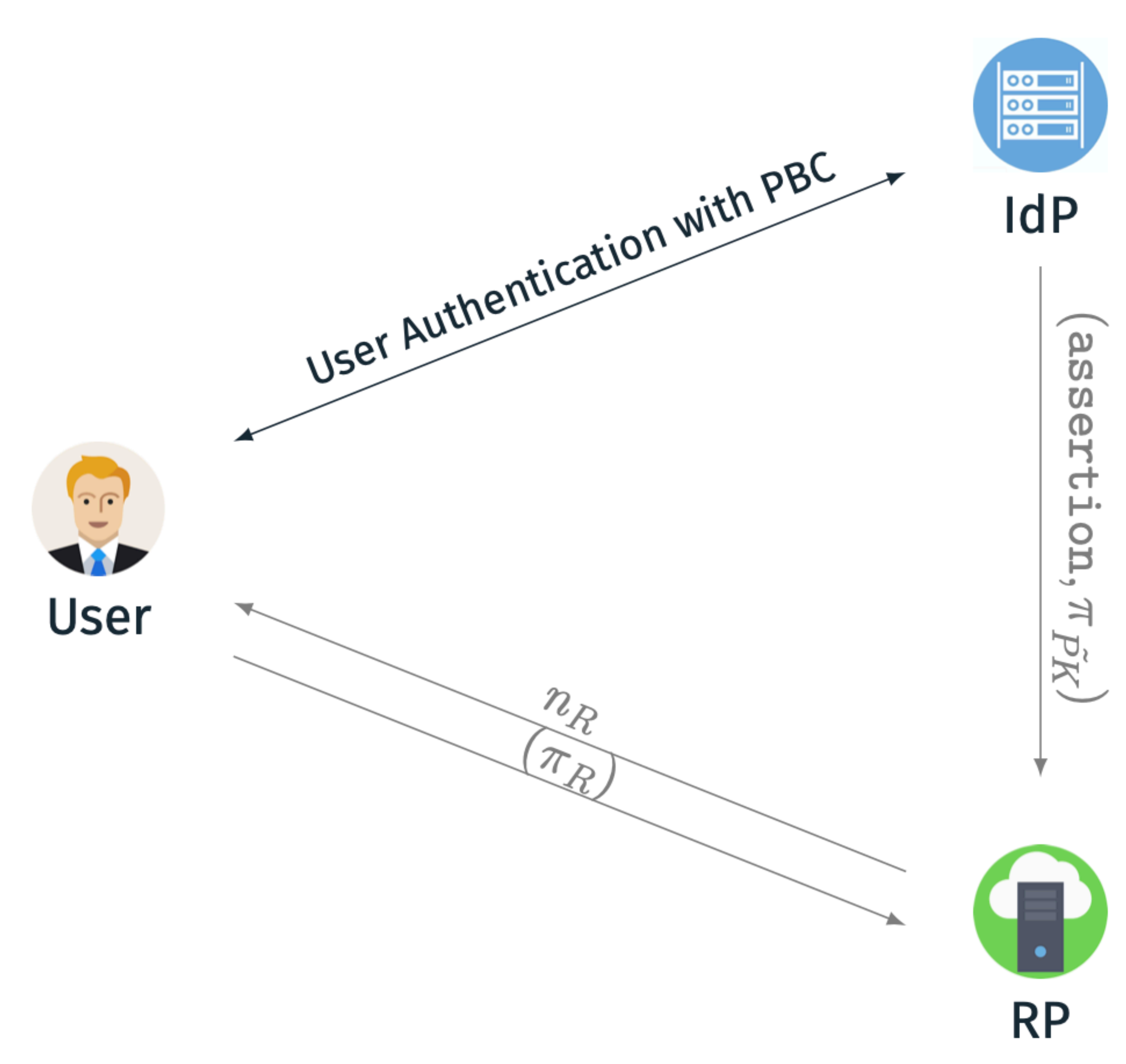
$$\pi_T \leftarrow \mathsf{SPK}\{(a) : g^a = PK\}(m) \text{ for } PK = T^{\gamma + uid}$$

- *Holder-of-Key Assertion.* The IdP calculates $\tilde{PK} = T^{\gamma}$ and sets assertion $\leftarrow (T, \tilde{PK})$, then signs assertion by generating a signature proof of knowledge:

$$\pi_{\tilde{PK}} = \mathsf{SPK}'\{(\gamma) : w = g^{\gamma} \wedge \tilde{PK} = T^{\gamma}\}(\cdot)$$

- *Proof-of-Possession of Key.* The user generates a proof-of-possession of the private-key w.r.t to $\tilde{PK}$, with a privacy-preserving authentication token:

$$\pi_R \leftarrow \mathsf{SPK}''\{(a, uid) : T^{-uid} \cdot g^a = \tilde{PK}\}(n_R)$$



Federated Identities with Password-based Credentials

Privacy-Preserving Holder-of-Key Assertion & PoPK with PBCs

- *User-IdP Authentication.* The user authenticates to IdP with a valid authentication token $\sigma = (T, \pi_T)$ s.t.

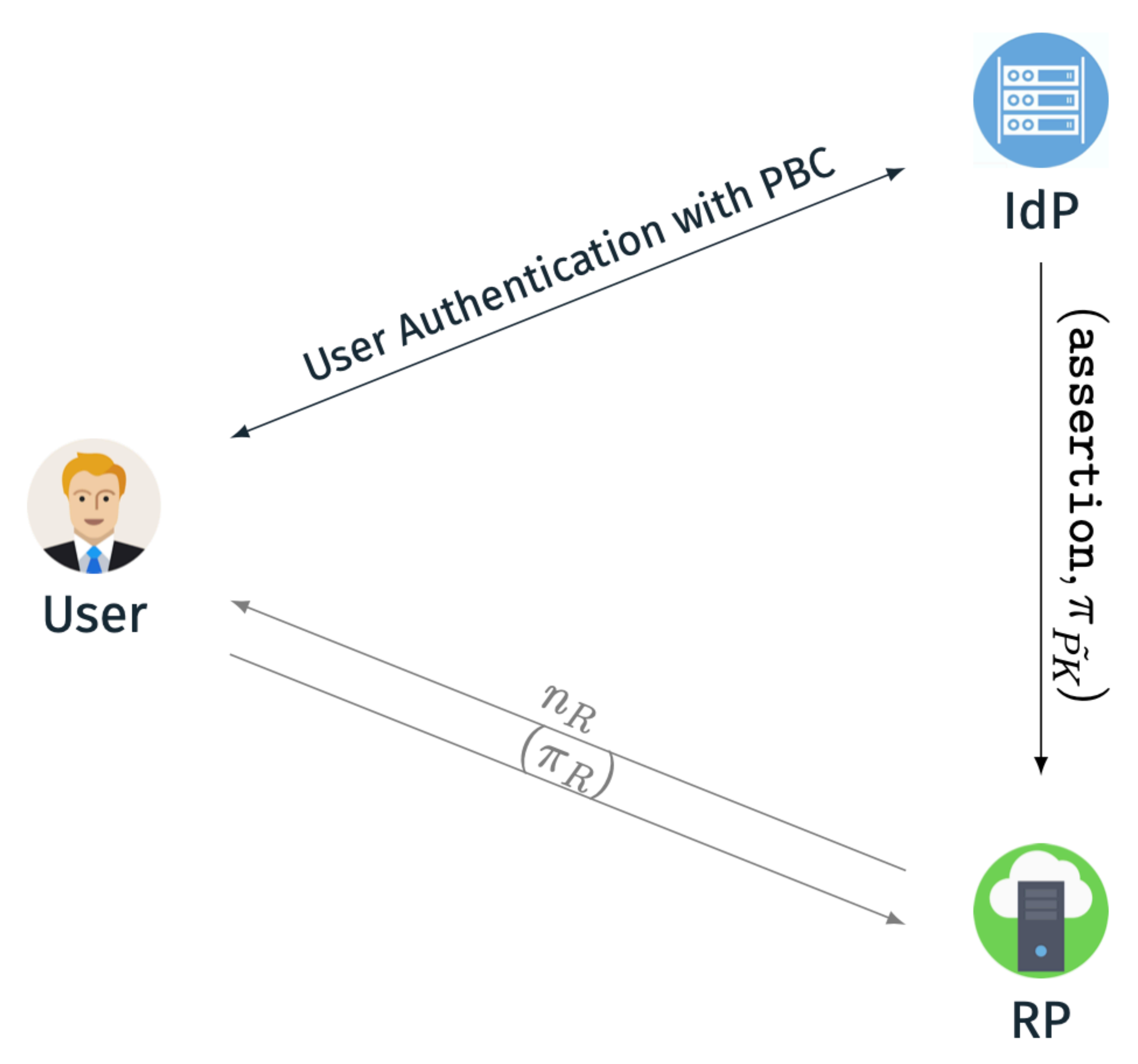
$$\pi_T \leftarrow \mathsf{SPK}\{(a) : g^a = PK\}(m) \text{ for } PK = T^{\gamma + uid}$$

- *Holder-of-Key Assertion.* The IdP calculates $\tilde{PK} = T^{\gamma}$ and sets assertion $\leftarrow (T, \tilde{PK})$, then signs assertion by generating a signature proof of knowledge:

$$\pi_{\tilde{PK}} = \mathsf{SPK}'\{(\gamma) : w = g^{\gamma} \wedge \tilde{PK} = T^{\gamma}\}(\cdot)$$

- *Proof-of-Possession of Key.* The user generates a proof-of-possession of the private-key w.r.t to $\tilde{PK}$, with a privacy-preserving authentication token:

$$\pi_R \leftarrow \mathsf{SPK}''\{(a, uid) : T^{-uid} \cdot g^a = \tilde{PK}\}(n_R)$$



Federated Identities with Password-based Credentials

Privacy-Preserving Holder-of-Key Assertion & PoPK with PBCs

- *User-IdP Authentication*. The user authenticates to IdP with a valid authentication token $\sigma = (T, \pi_T)$ s.t.

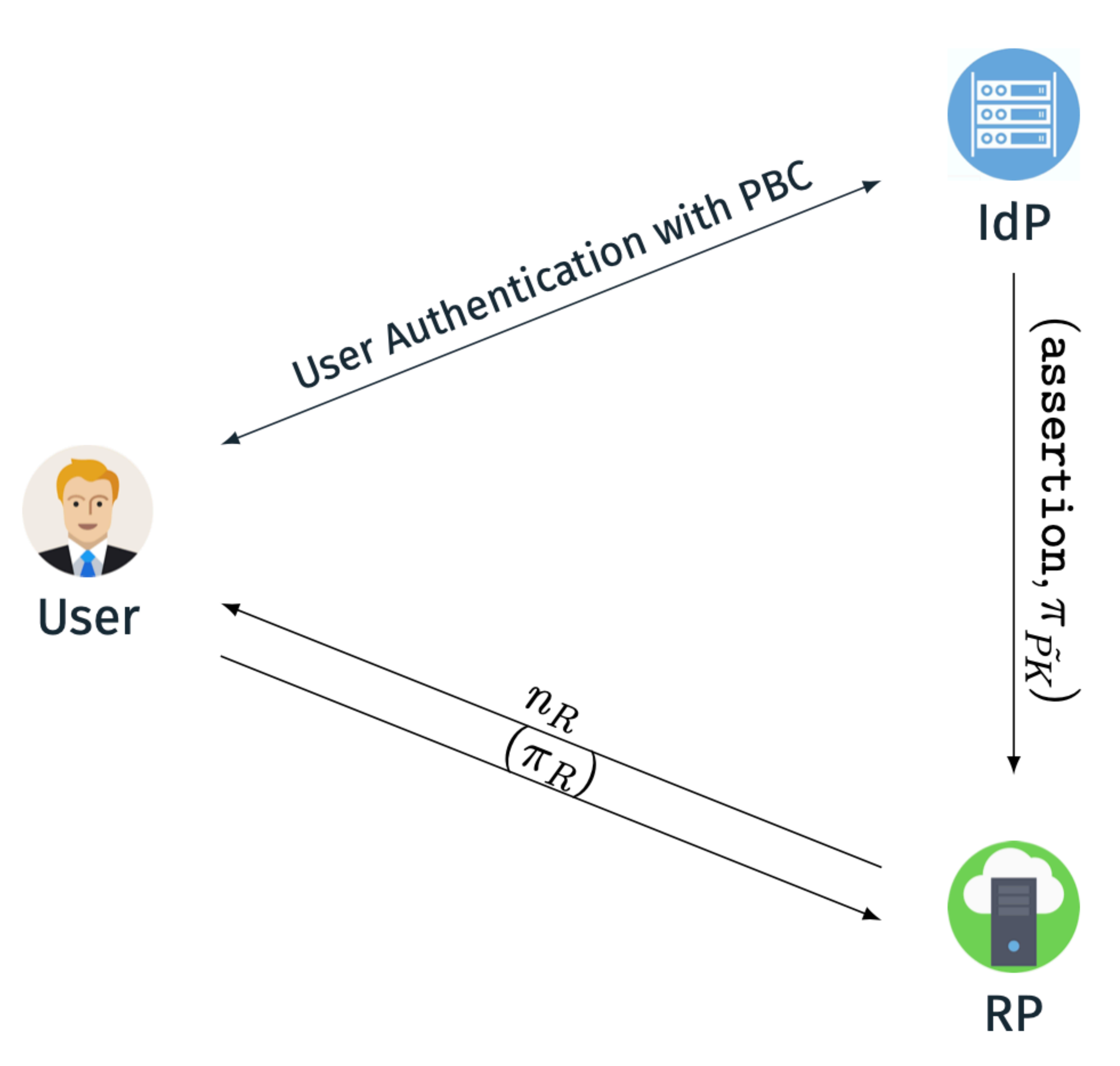
$$\pi_T \leftarrow \mathsf{SPK}\{(a) : g^a = PK\}(m) \text{ for } PK = T^{\gamma + uid}$$

- *Holder-of-Key Assertion*. The IdP calculates $\tilde{PK} = T^\gamma$ and sets assertion $\leftarrow (T, \tilde{PK})$, then signs assertion by generating a signature proof of knowledge:

$$\pi_{\tilde{PK}} = \mathsf{SPK}'\{(\gamma) : w = g^\gamma \wedge \tilde{PK} = T^\gamma\}(\cdot)$$

- *Proof-of-Possession of Key*. The user generates a proof-of-possession of the private-key w.r.t to $\tilde{PK}$, with a privacy-preserving authentication token:

$$\pi_R \leftarrow \mathsf{SPK}''\{(a, uid) : T^{-uid} \cdot g^a = \tilde{PK}\}(n_R).$$



Performance Evaluation

- AUTH-x strong authentication, x-ECDSA/PBC with/without tamper-resistant hardware at user-end

	token/assertion generation	token/assertion verification	LAN	WAN			
				30ms	60ms	90ms	120ms
AUTH-ECDSA	$272.4^{\dagger *}$	1.1	$300.1^{\dagger *}$	$342.4^{\dagger *}$	$376.2^{\dagger *}$	$390.1^{\dagger *}$	$432.3^{\dagger *}$
AUTH-PBC	$187.5^{\dagger}$	1.0	$192.4^{\dagger}$	$224.9^{\dagger}$	$250.6^{\dagger}$	$284.3^{\dagger}$	$319.5^{\dagger}$
PoPK-ECDSA	$271.1^{\dagger *}$	1.1	$305.4^{\dagger *}$	$334.3^{\dagger *}$	$370.8^{\dagger *}$	$400.6^{\dagger *}$	$425.3^{\dagger *}$
PoPK-PBC	$100.6^{\dagger}$	1.0	$125.0^{\dagger}$	$149.7^{\dagger}$	$188.8^{\dagger}$	$219.0^{\dagger}$	$250.2^{\dagger}$
PoPK-PBC′	$167.3^{\dagger}$	1.0	$190.5^{\dagger}$	$223.7^{\dagger}$	$245.2^{\dagger}$	$281.1^{\dagger}$	$314.2^{\dagger}$
HoKA-ECDSA	0.7	1.0	3.3	34.7	65.2	93.9	124.5
HoKA-PBC	2.1	2.4	5.1	38.3	69.4	98.7	129.0
HoKA-PBC′	2.0	1.9	5.0	37.2	68.8	98.4	127.1

Conclusions and Take-aways

- Strong authentication without tamper-resistant hardware modules
  - Highly practical construction from PBCs
  - Resistant against offline attacks & token-forgery attacks
- Federated identity system from PBCs
  - User-IdP strong authentication
  - (Privacy-preserving) holder-of-key assertion
- User-friendly and easy-to-implement
  - On general-purpose devices, via common programming languages
  - Authenticator backup in case of devices broken/lost

Thanks for the attention ！

Strong Authentication without Tamper-Resistant Hardware and Application to Federated Identities

Zhenfeng Zhang[✝,⚛], Yuchen Wang[✝] and Kang Yang[✧]

✝ Institute of Software, Chinese Academy of Sciences;
⚛ The Joint Academy of Blockchain Innovation;
✧ State Key Laboratory of Cryptology

Contact: zhenfeng@iscas.ac.cn, yuchenwang@tca.iscas.ac.cn