# Post-Quantum Asynchronous Remote Key Generation for FIDO2



Jacqueline Brendel, **Sebastian Clermont**, Marc Fischlin

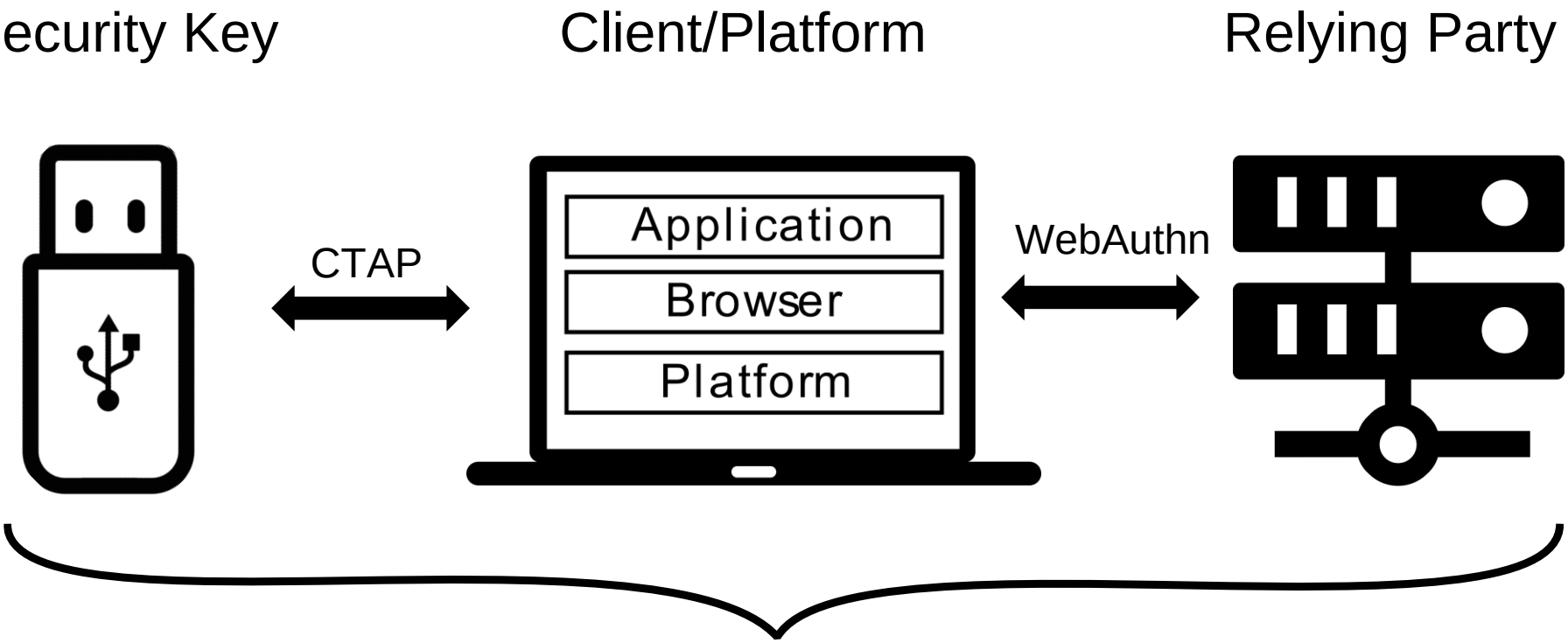
Cryptoplexity, Technische Universität Darmstadt, Germany
{sebastian.clermont, marc.fischlin}@tu-darmstadt.de
mail@jbrendel-info.de

# What is FIDO?

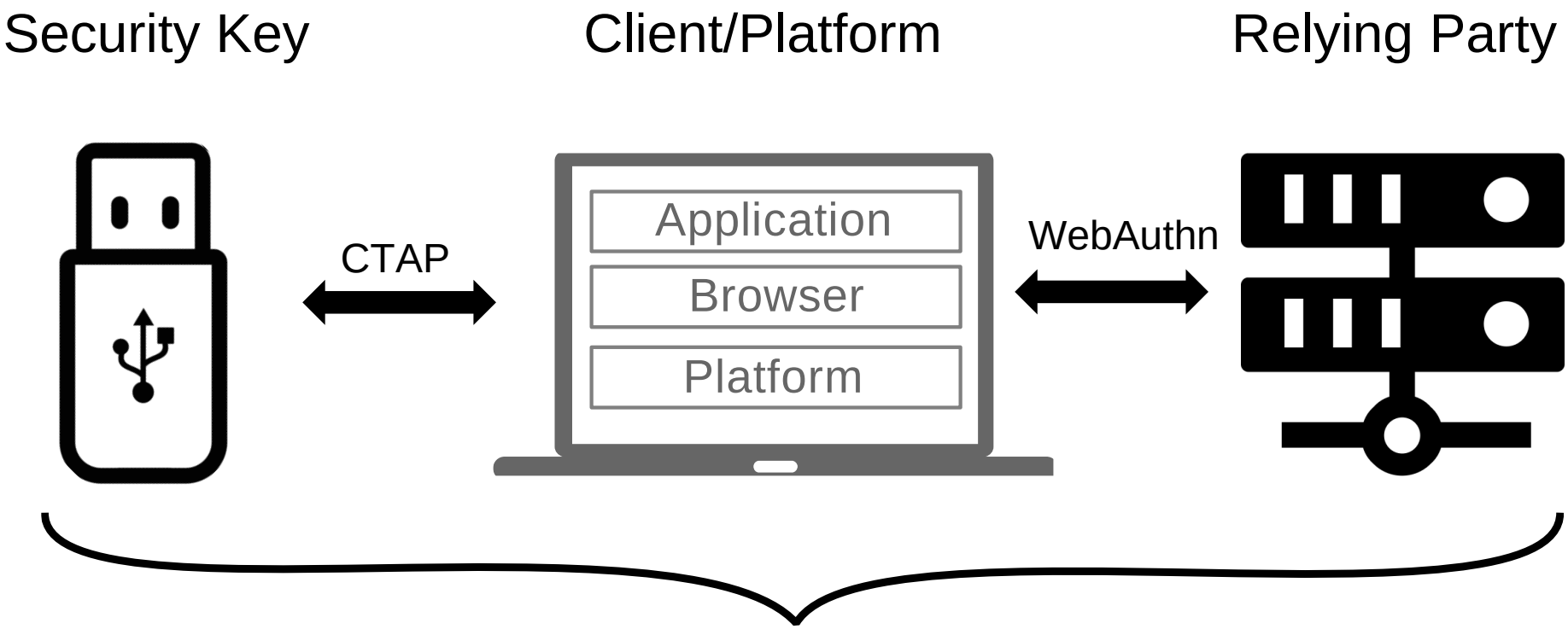

fido™ ALLIANCE

Images from https://fidoalliance.org/fido2/

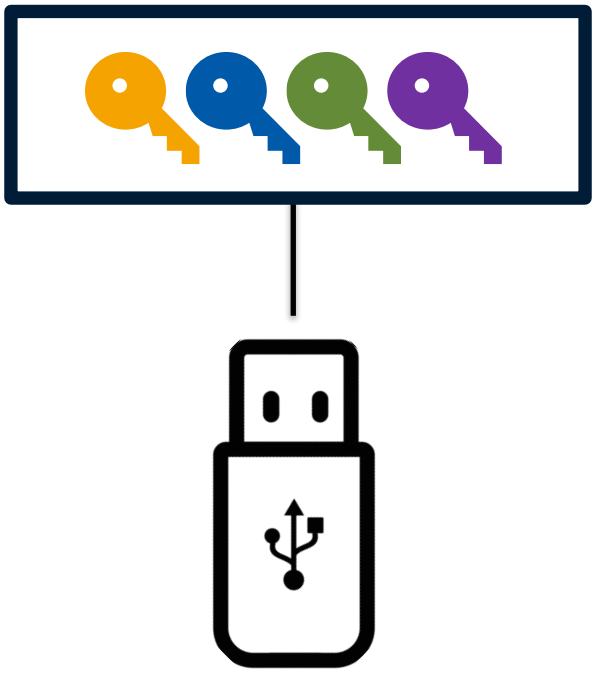
# What is FIDO2?



# What is FIDO2?



# Passkeys and Security Keys



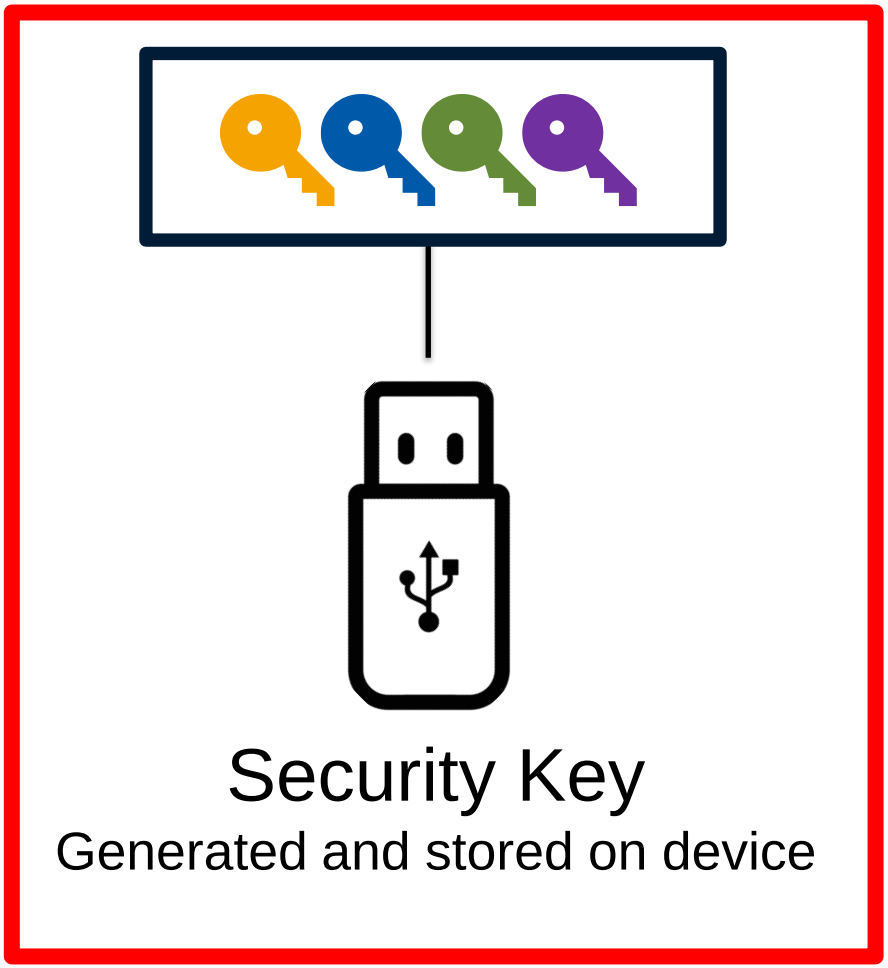
Security Key

Generated and stored on device

Passkey

Synchronized with cloud

# Passkeys and Security Keys



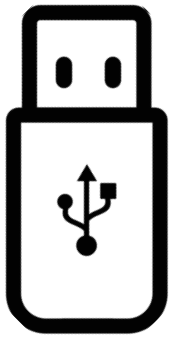
Security Key

Generated and stored on device

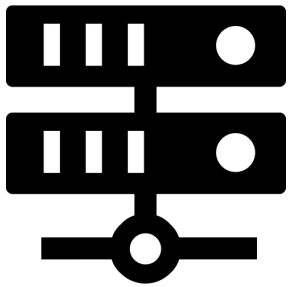
Passkey

Synchronized with cloud

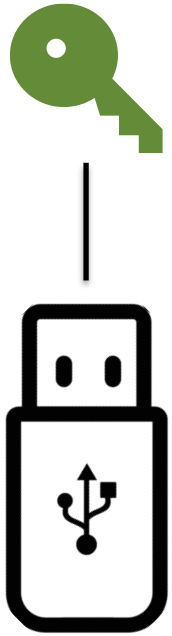
# FIDO2 - Registration



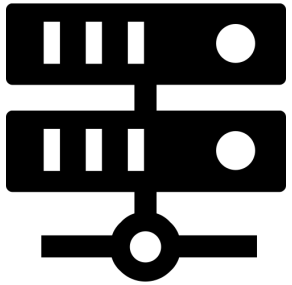
**Registration**



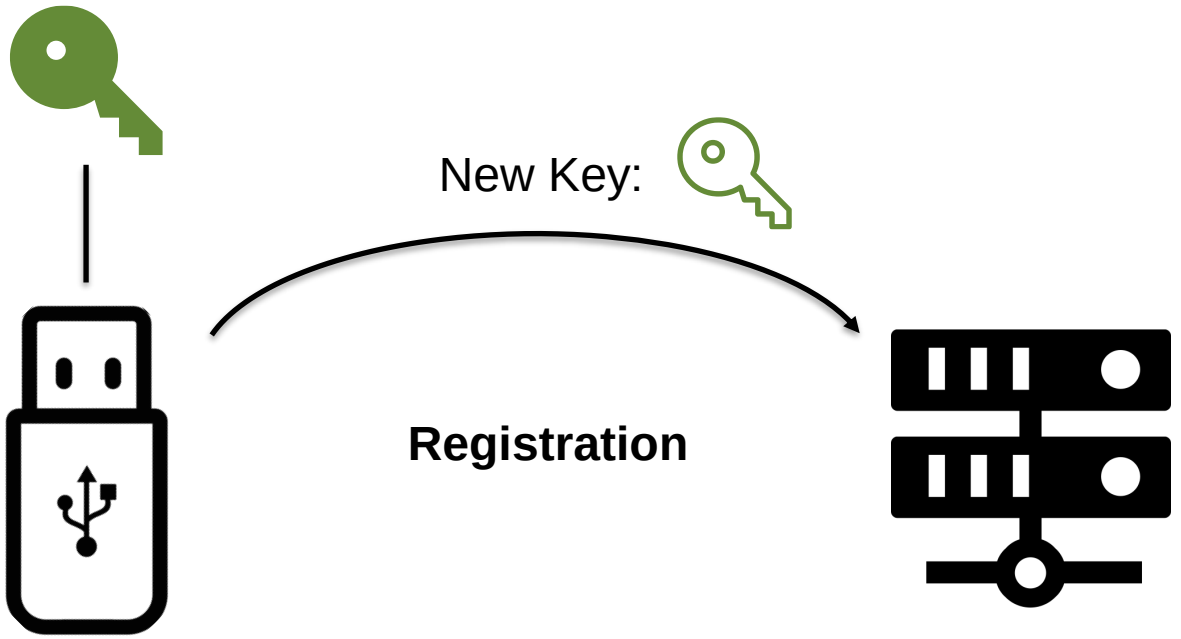
# FIDO2 - Registration



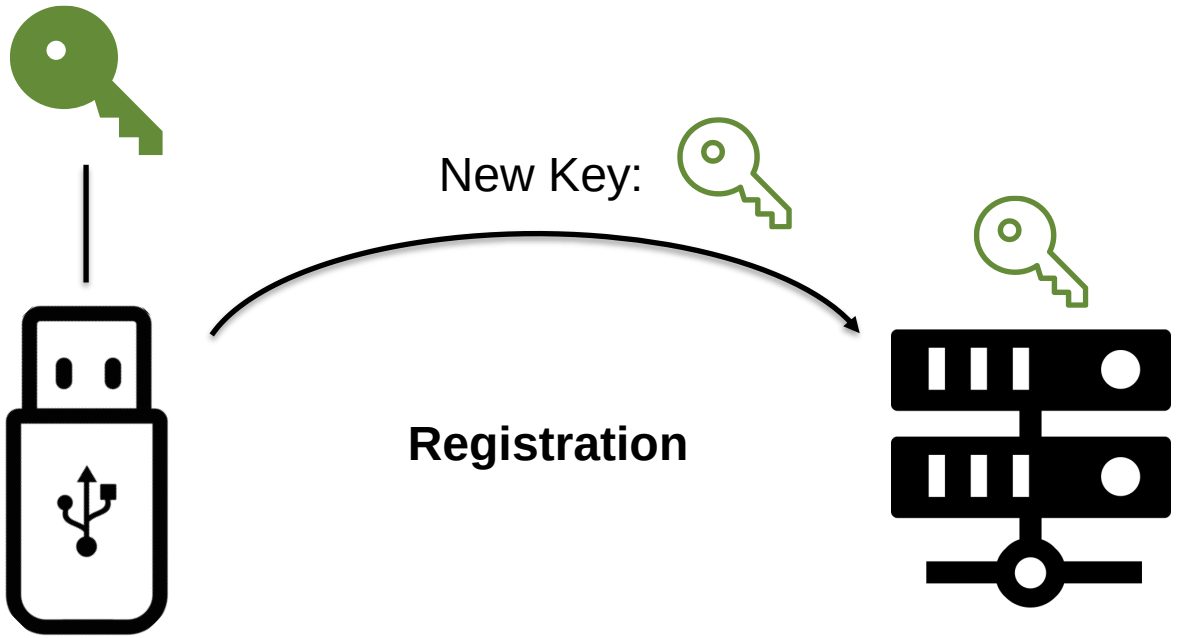
**Registration**



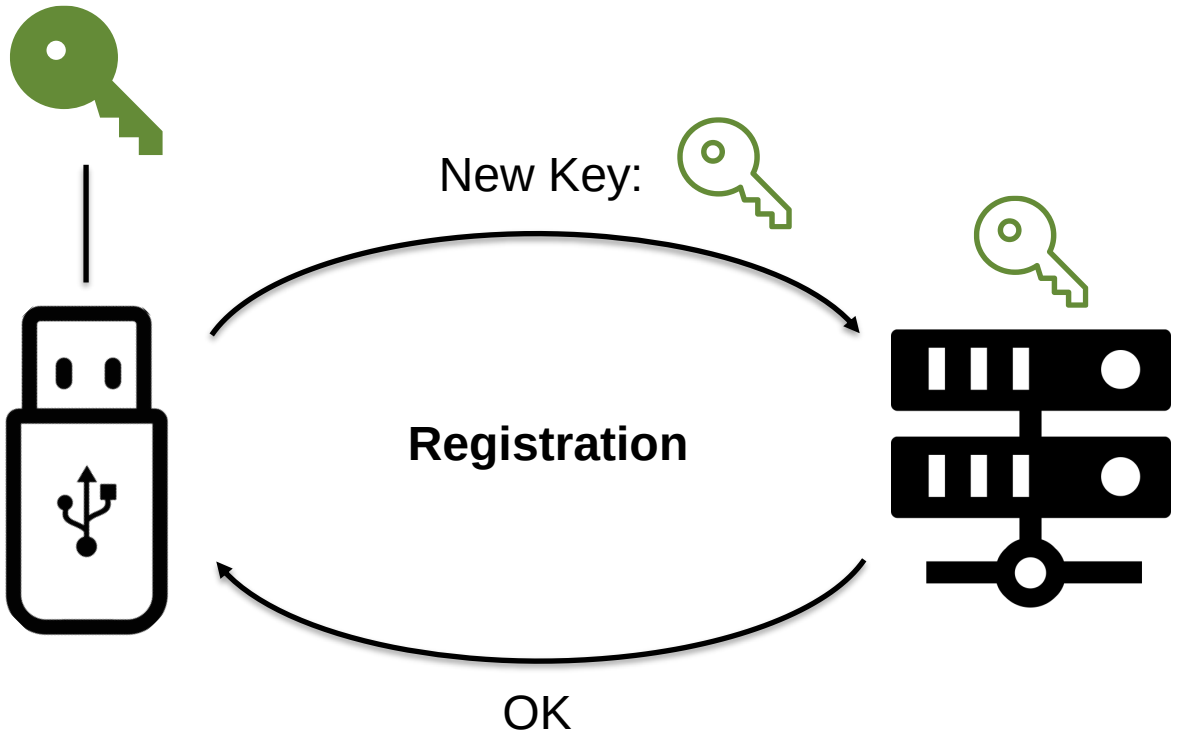
# FIDO2 - Registration



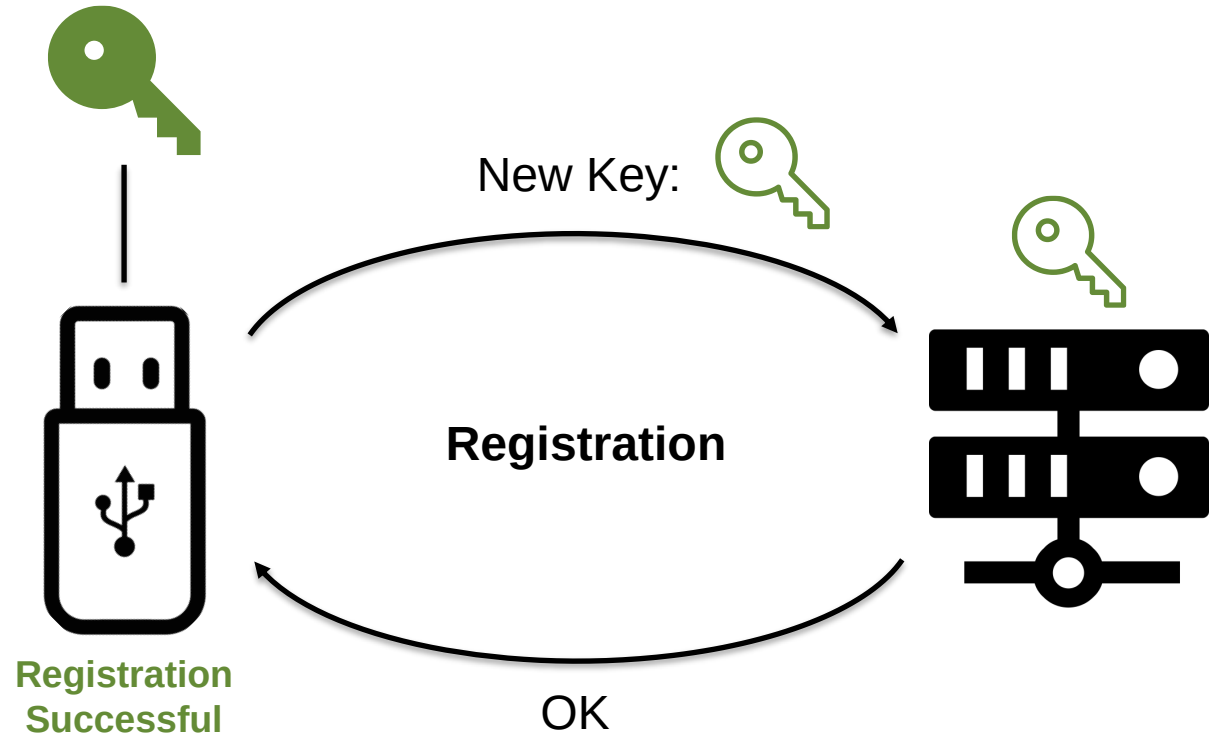
# FIDO2 - Registration



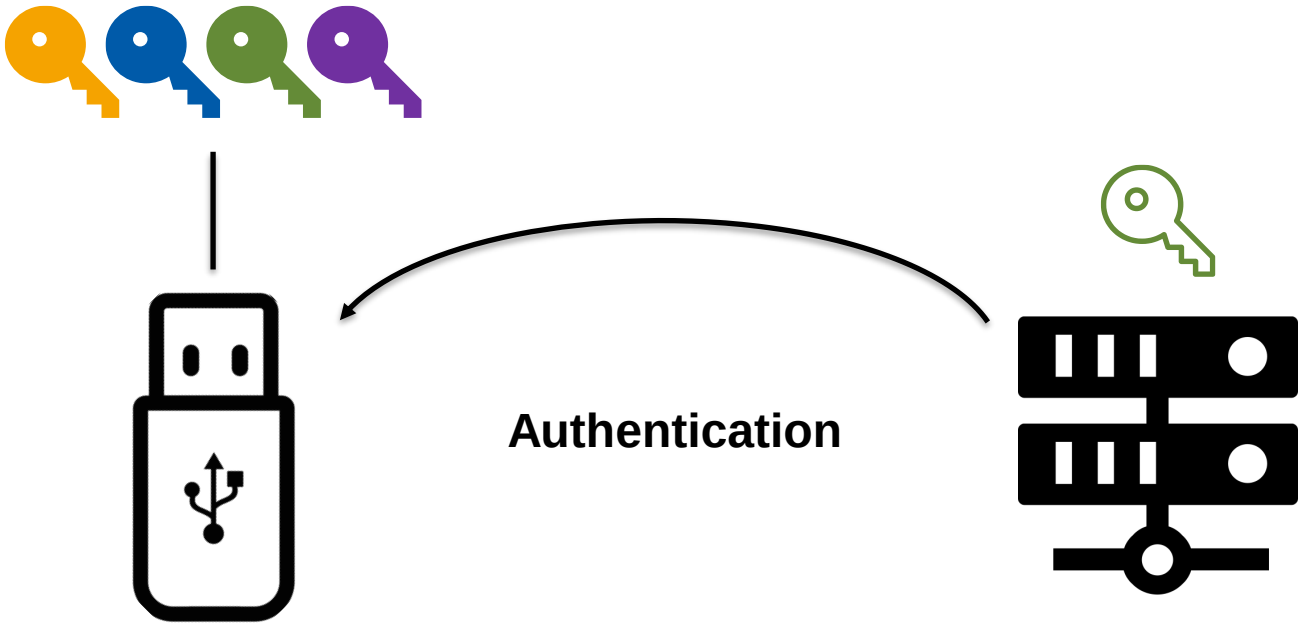
# FIDO2 - Registration



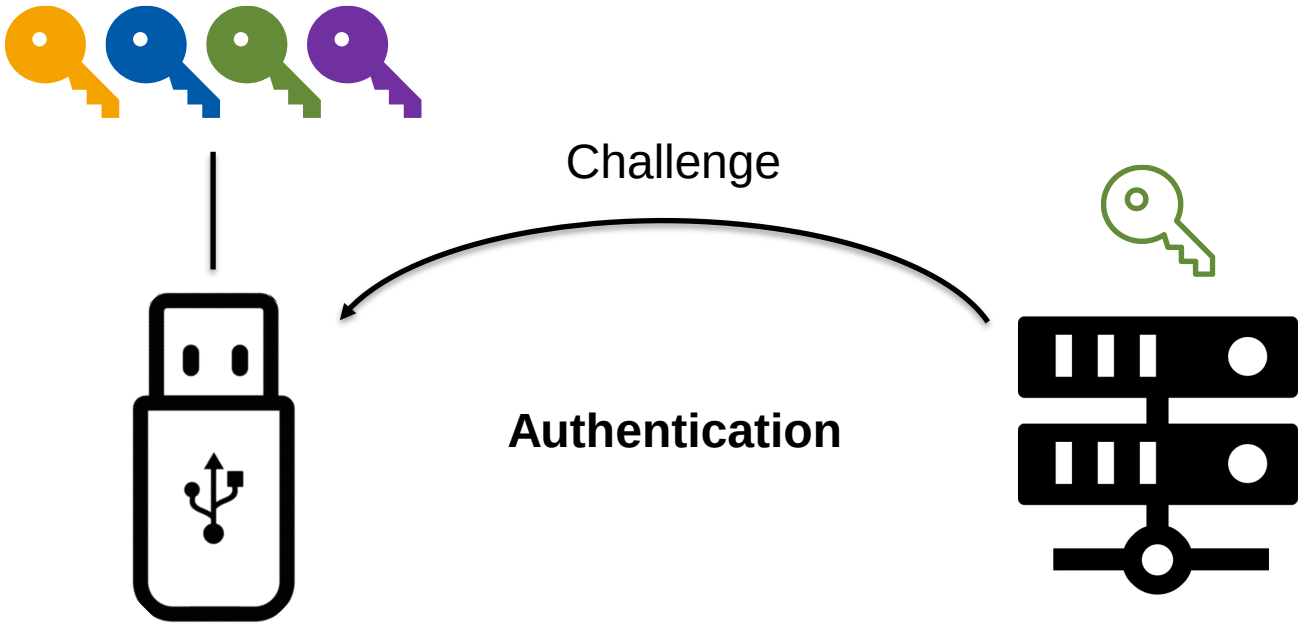
# FIDO2 - Registration



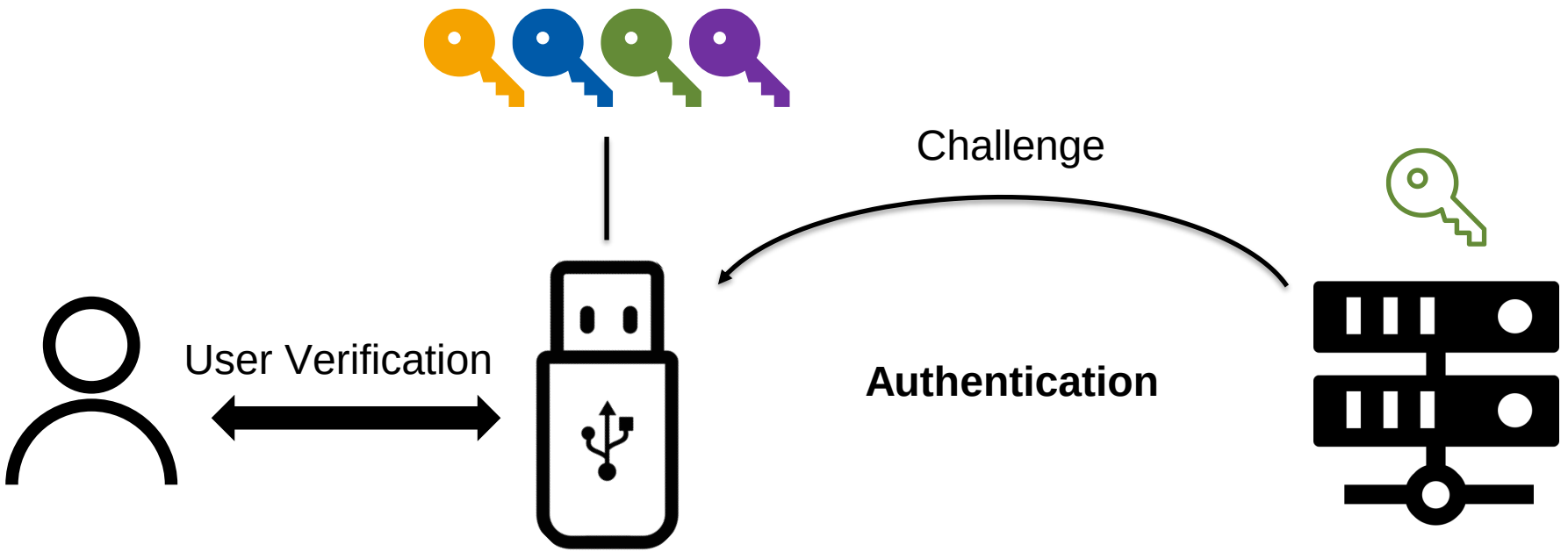
# FIDO2 - Authentication



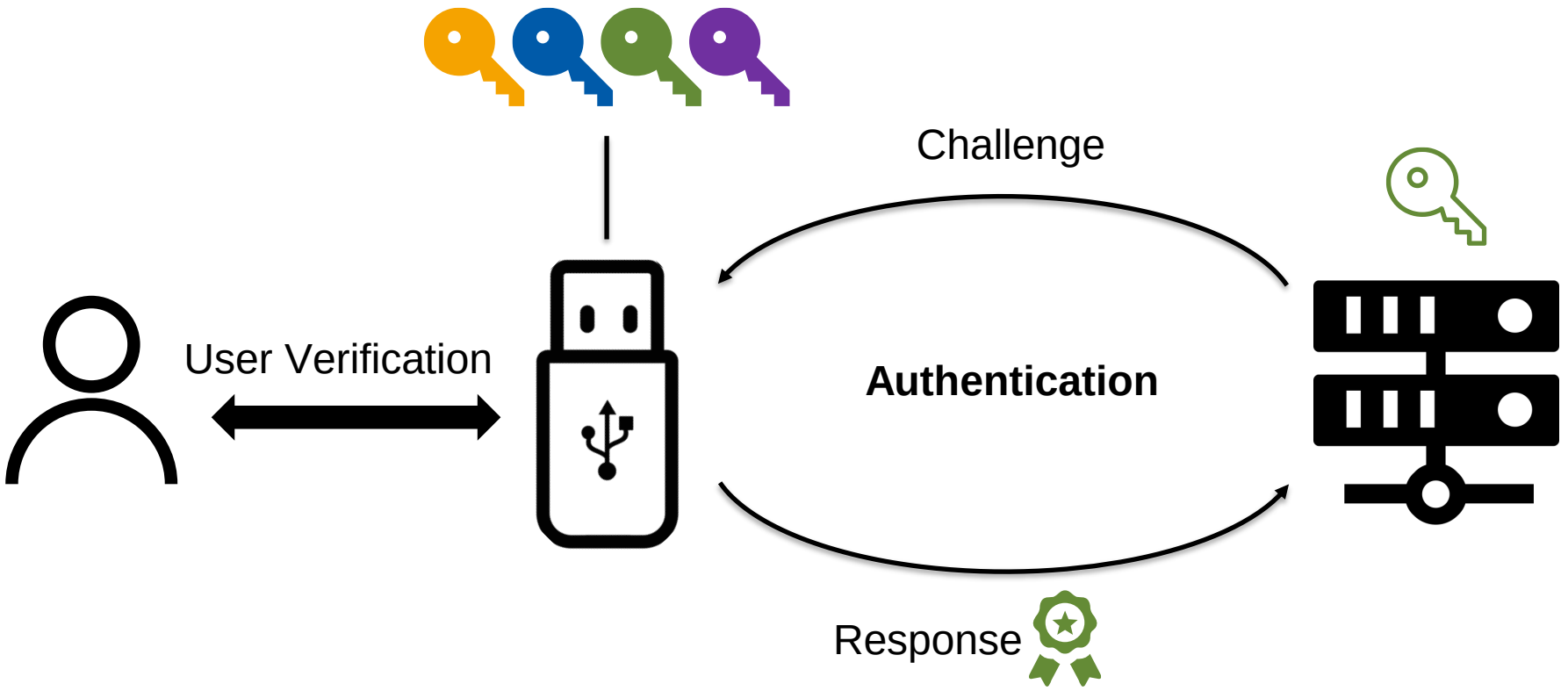
# FIDO2 - Authentication



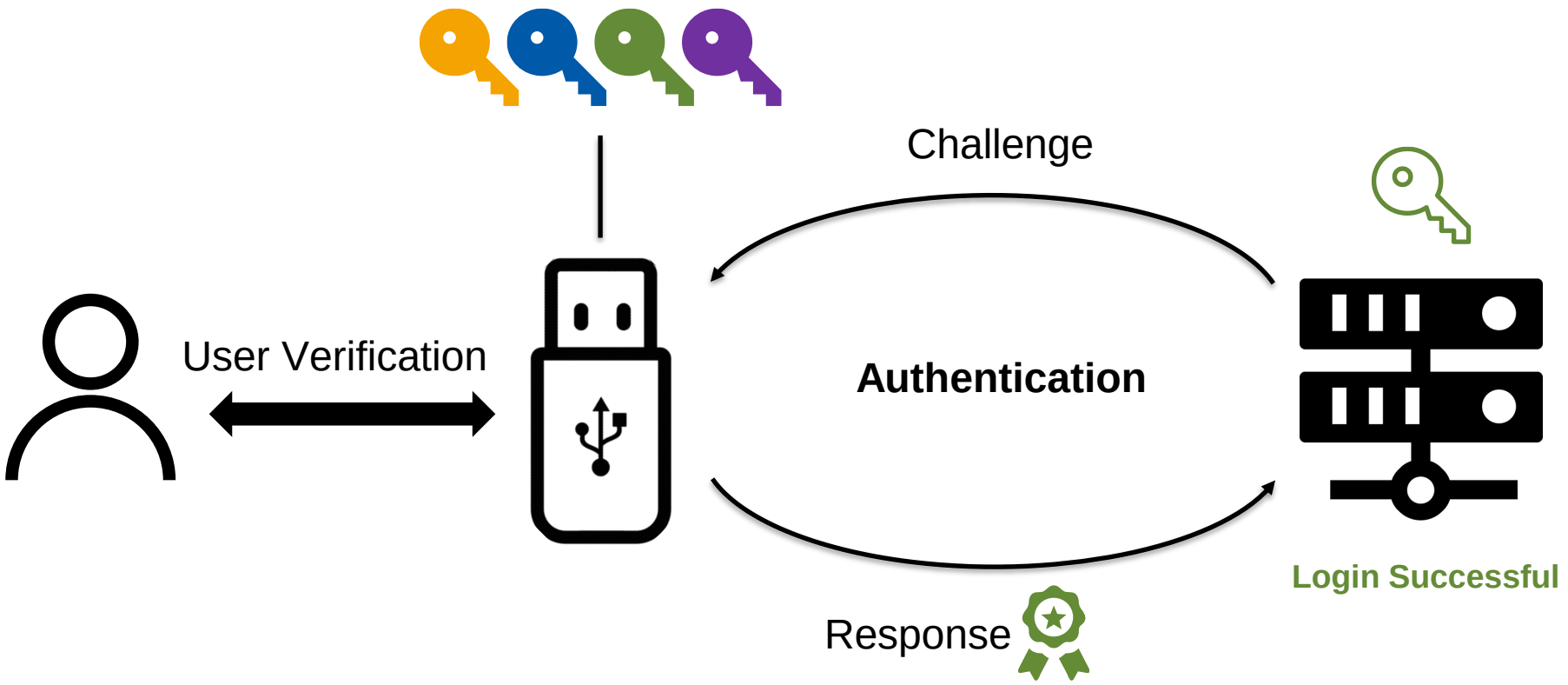
# FIDO2 - Authentication



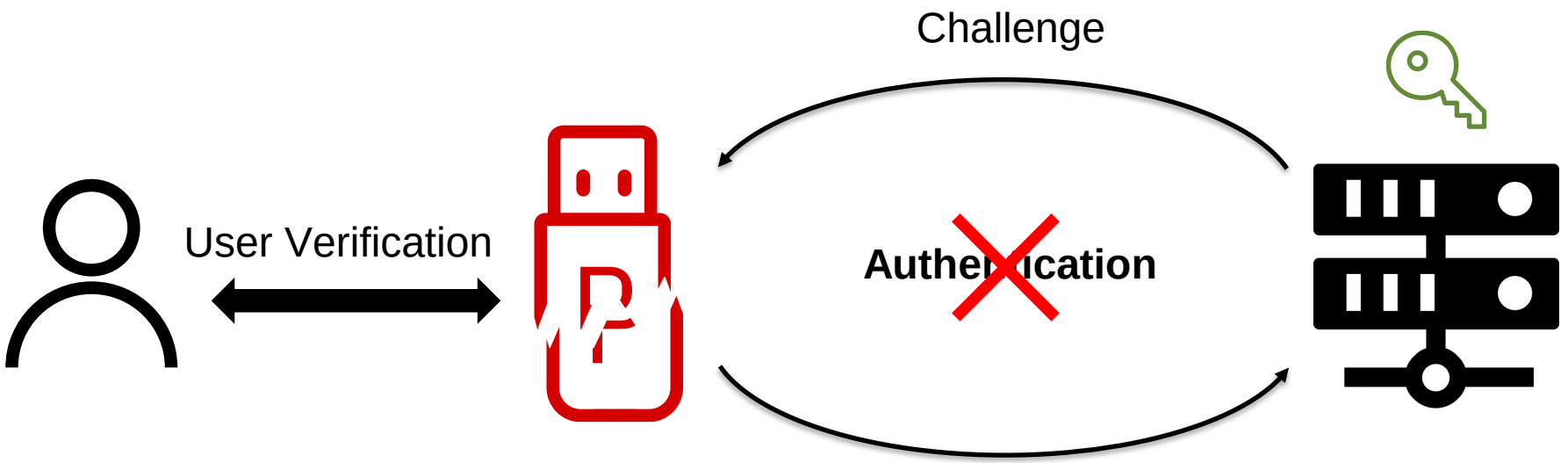
# FIDO2 - Authentication



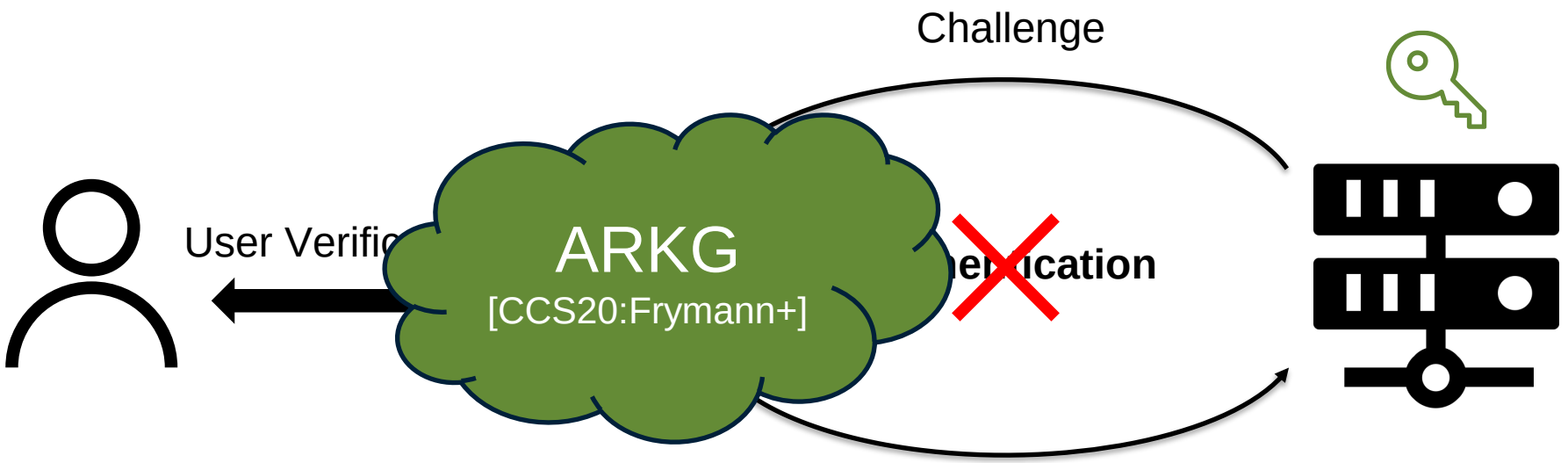
# FIDO2 - Authentication



# Authenticator Loss



# Authenticator Loss

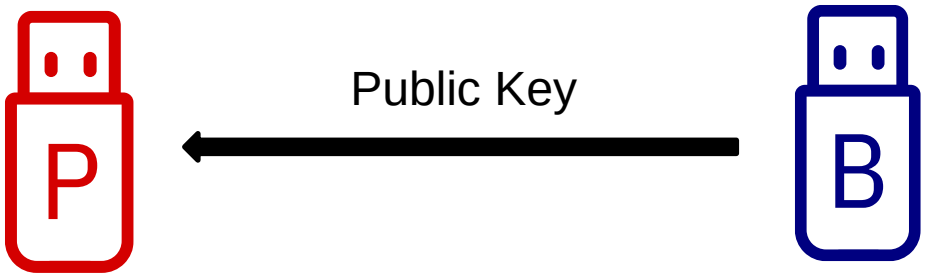




# Asynchronous Remote Key Generation

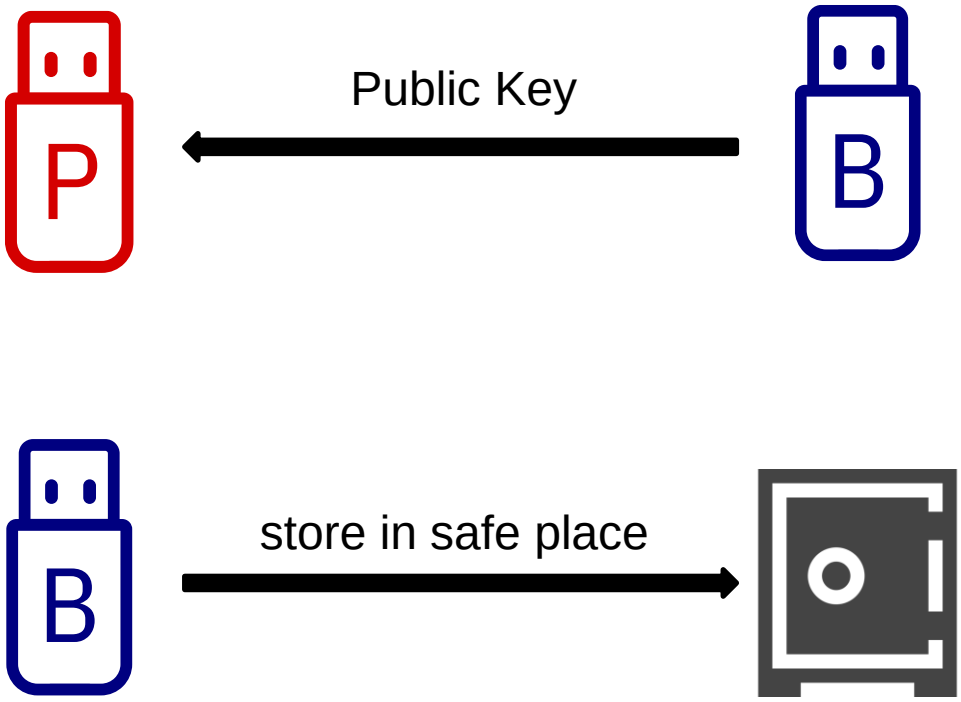
# ARKG Functionality: Pairing

**Initial Pairing**
**1** time



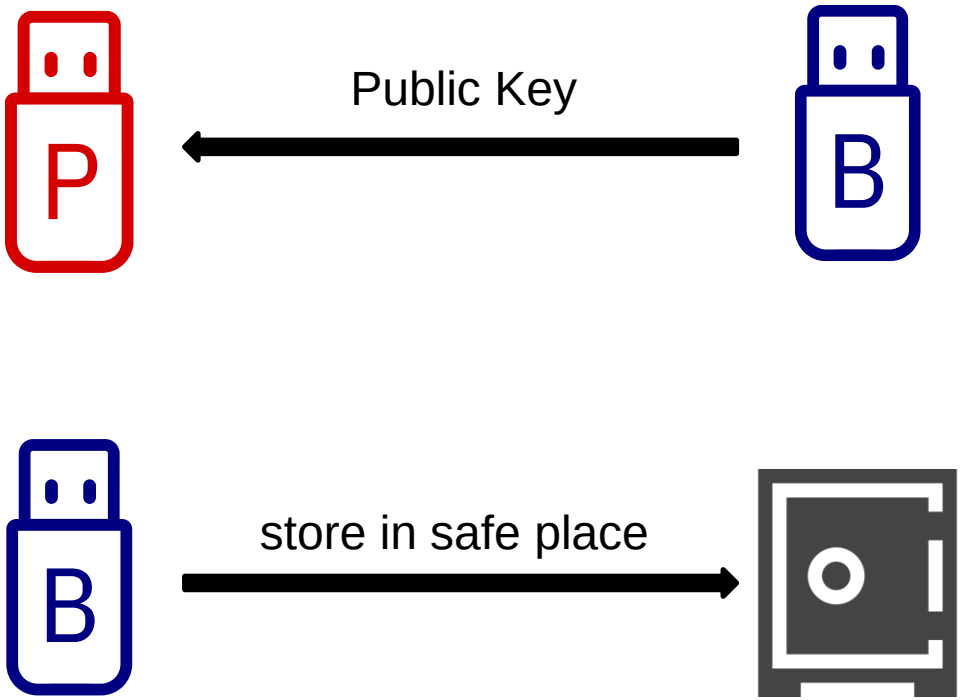
# ARKG Functionality: Pairing

**Initial Pairing**
**1** time



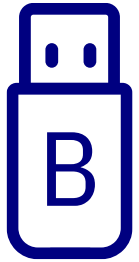
# ARKG Functionality: Pairing

**Initial Pairing**
**1** time



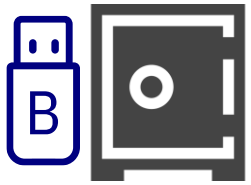

Key Generation

{ }

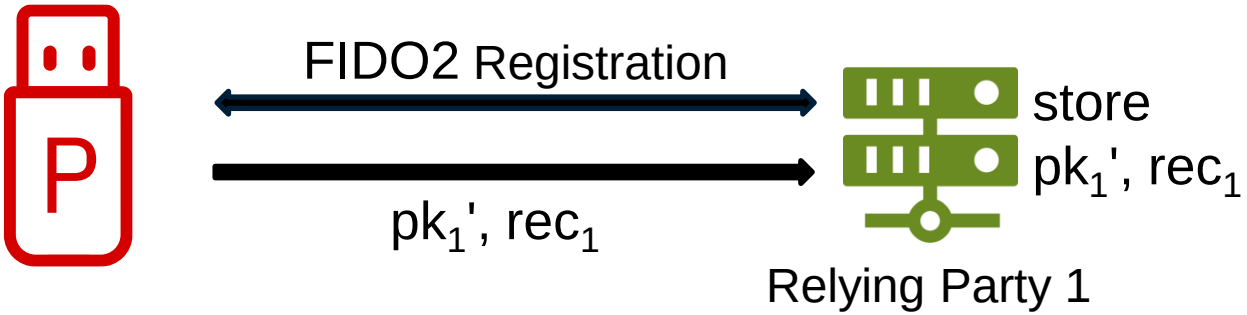



$pk_0$, $sk_0$

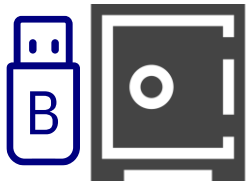
# ARKG Functionality: Registration



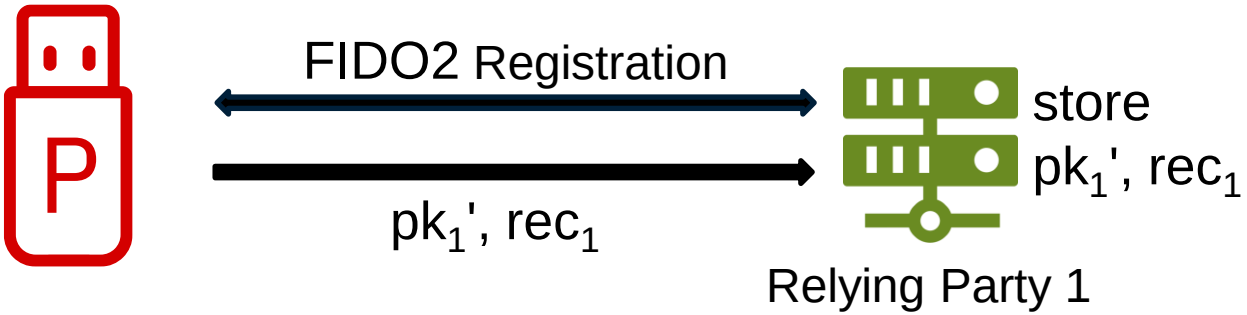

**Registration**
**n** times



# ARKG Functionality: Registration

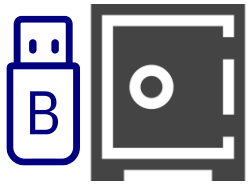



**Registration**
**n** times

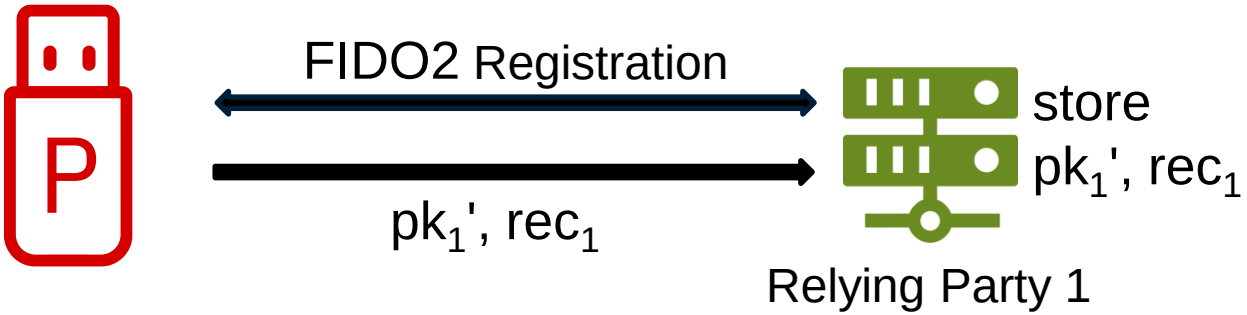



...

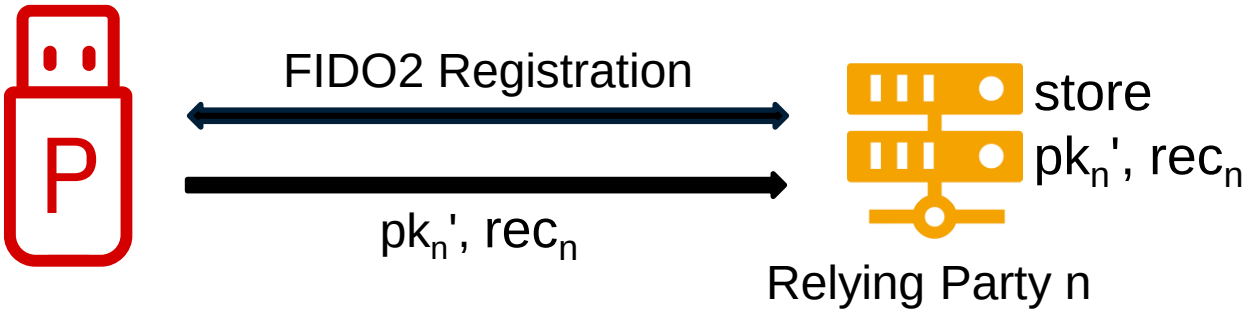
# ARKG Functionality: Registration



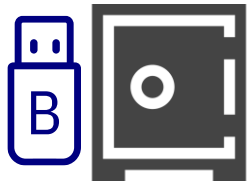

**Registration**
**n** times



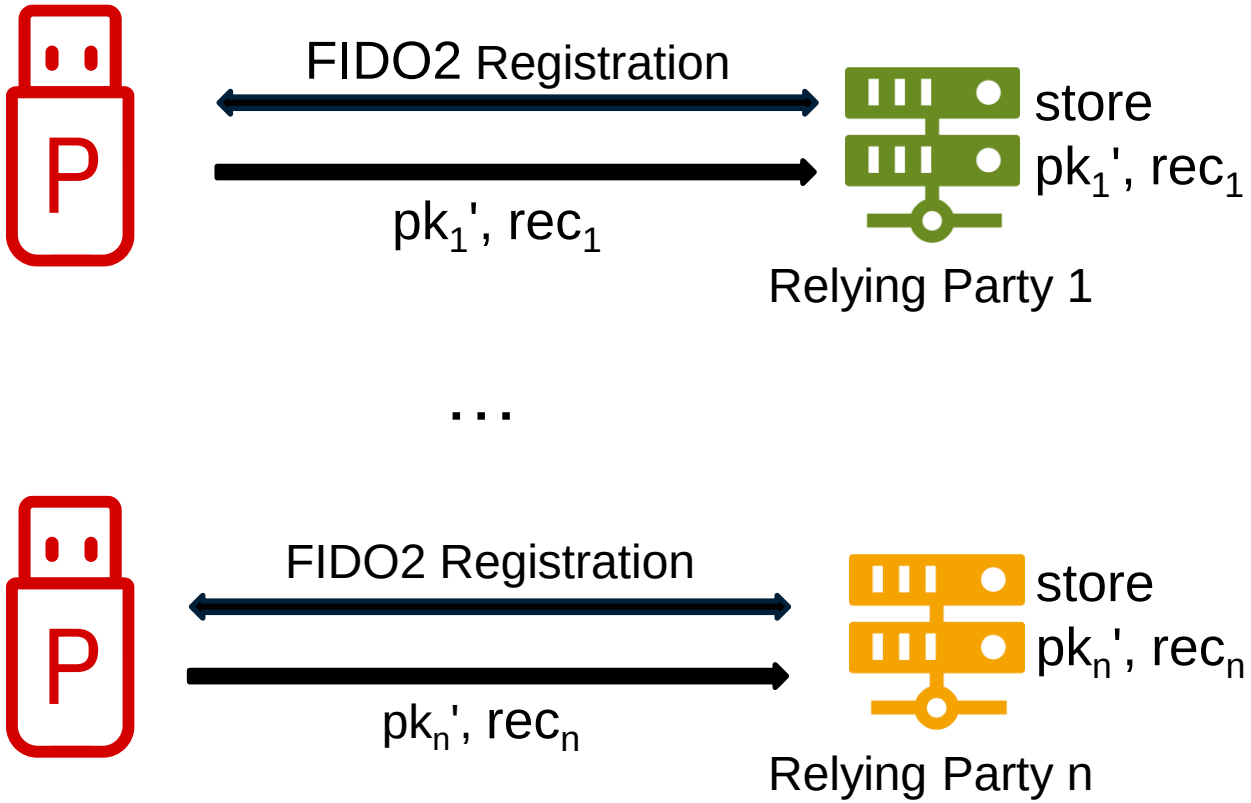

...



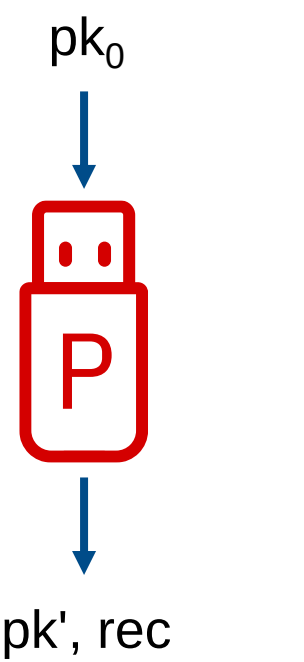
# ARKG Functionality: Registration




**Registration**
**n** times



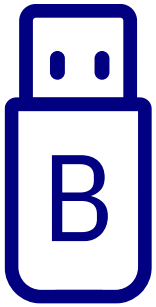

Derive Public Key

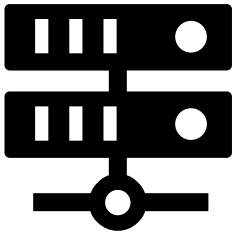


# ARKG Functionality: Recovery

**Recovery**
**n** times

„Recover for user X“

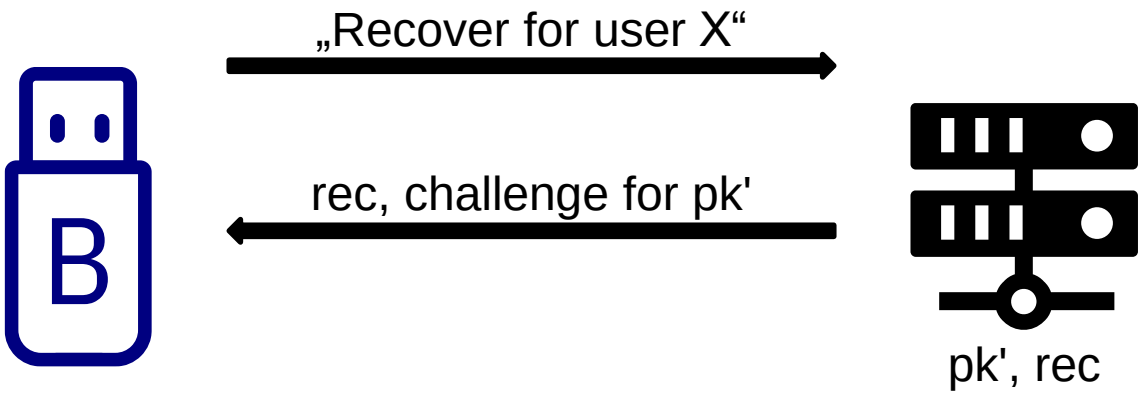


pk', rec

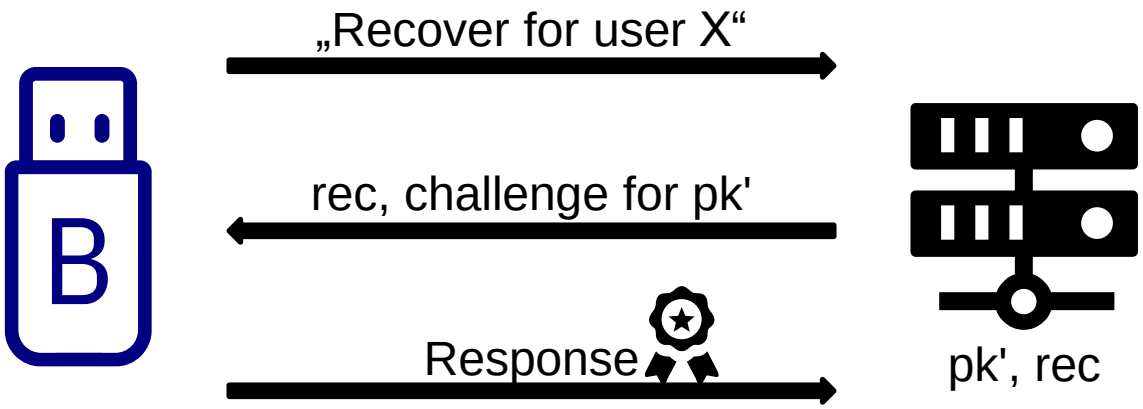
# ARKG Functionality: Recovery

**Recovery**
**n** times



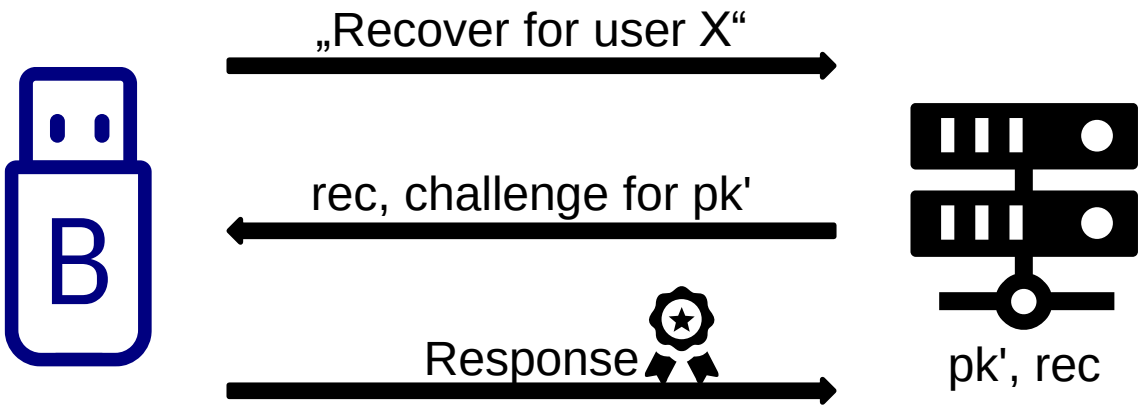
# ARKG Functionality: Recovery

**Recovery**
**n** times



# ARKG Functionality: Recovery

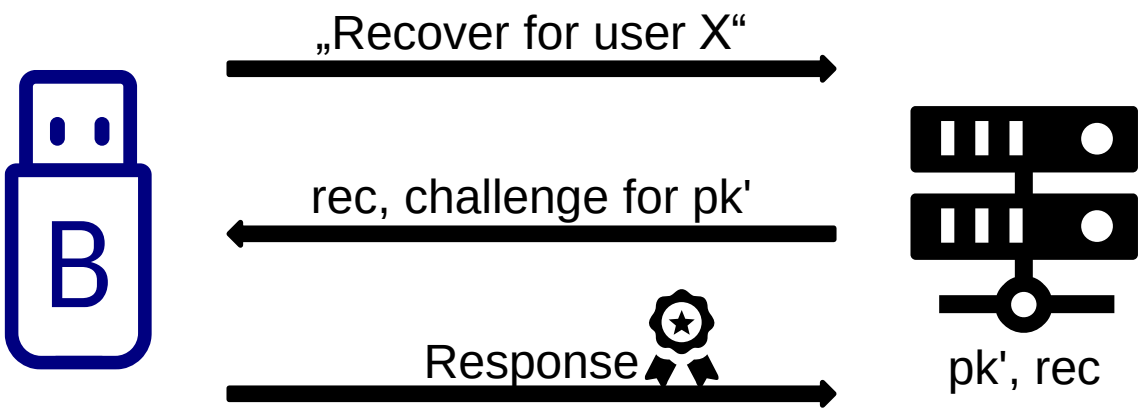
**Recovery**
**n** times




**recovery is successful, if valid signature under pk' is returned**

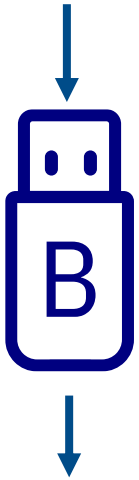
# ARKG Functionality: Recovery

**Recovery**
**n** times




Derive Secret Key

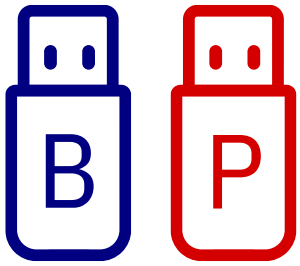
$sk_0$, rec




**recovery is successful, if valid signature under pk' is returned**

# Interactions Summarized

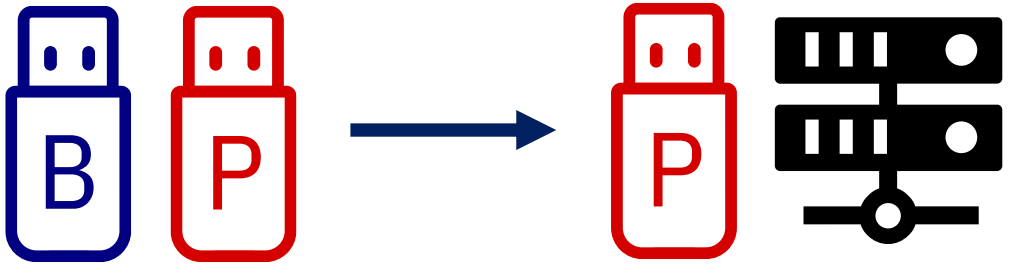
# Interactions Summarized




**Pair**

Key Generation

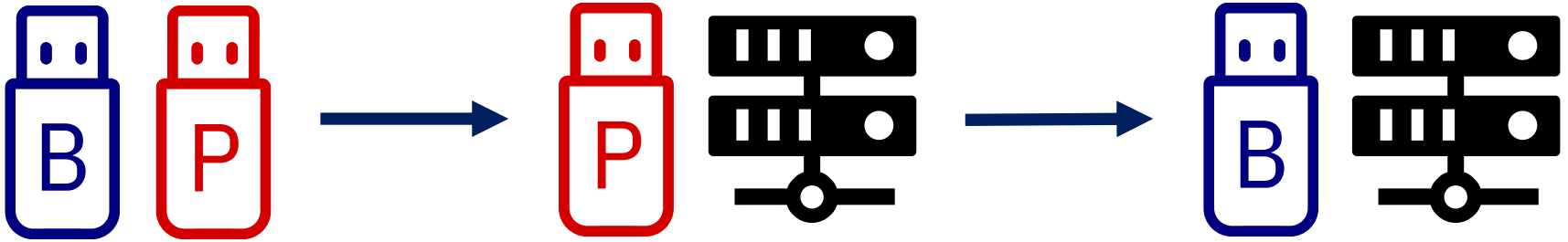
# Interactions Summarized




**Pair**
Key Generation

**Register**
Derive Public Key

# Interactions Summarized



# Key Contributions

1. Update security notions to match FIDO2 setting
2. Generic construction from standardized primitives
3. Instantiation in post-quantum setting

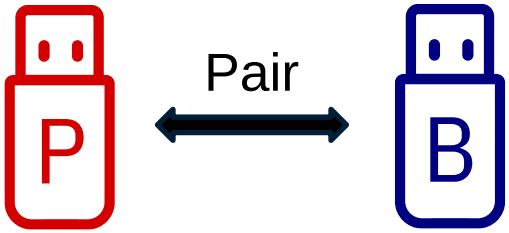
# Post-Quantum Asynchronous Remote Key Generation for FIDO2



## Security of ARKG Schemes

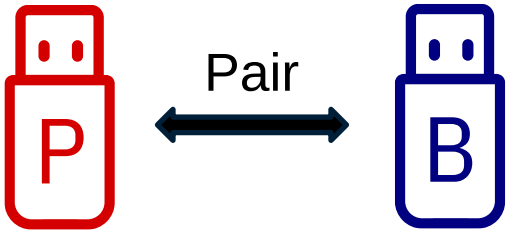
# Authentication Security

## Observe

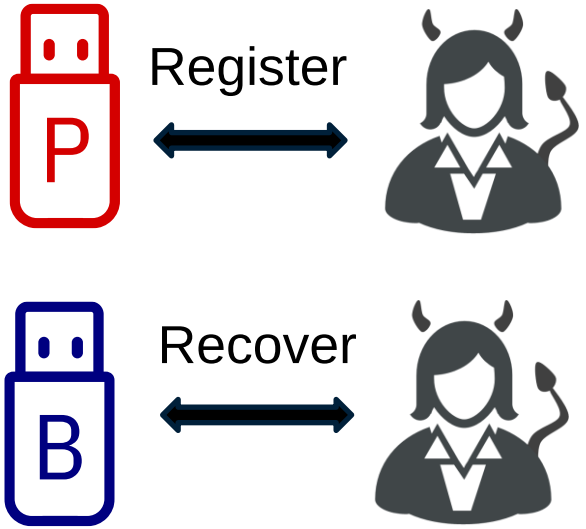


# Authentication Security

## Observe

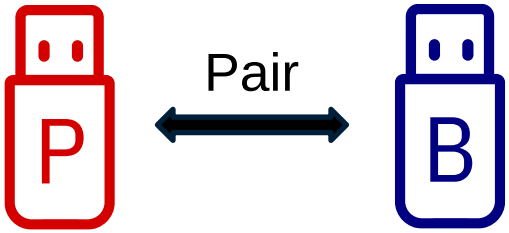



## Participate

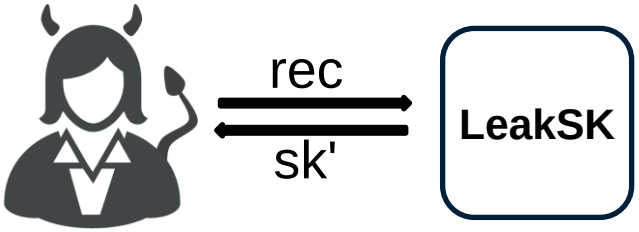


# Authentication Security

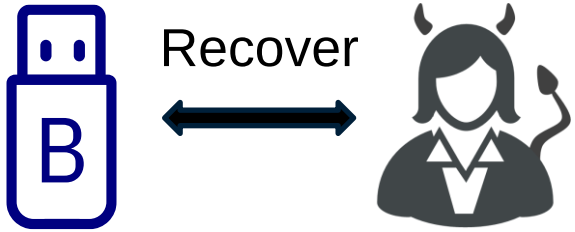
## Observe




## Leak

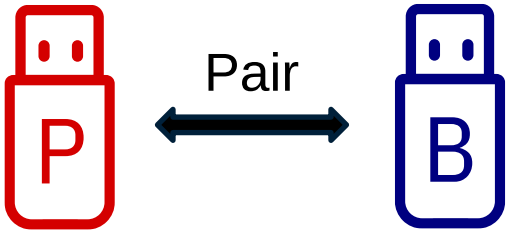



## Participate

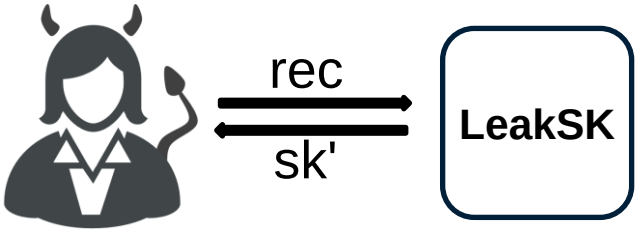
# Authentication Security

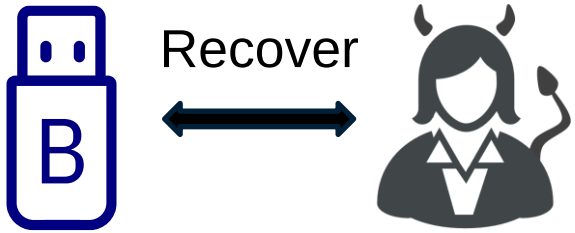
## Observe



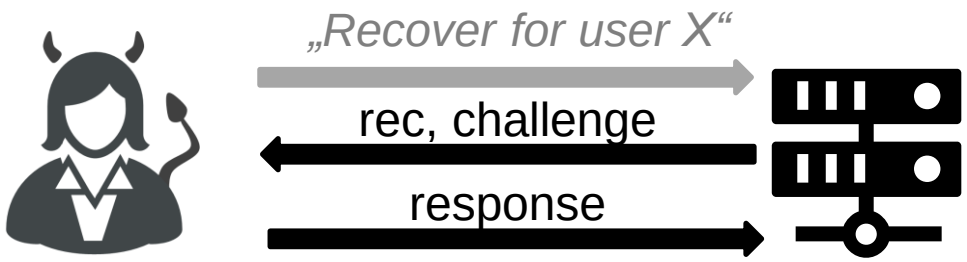

## Participate


## Leak



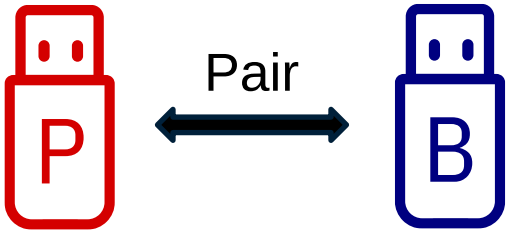




## Attack



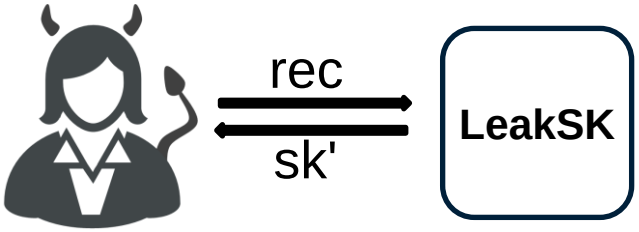
# Authentication Security

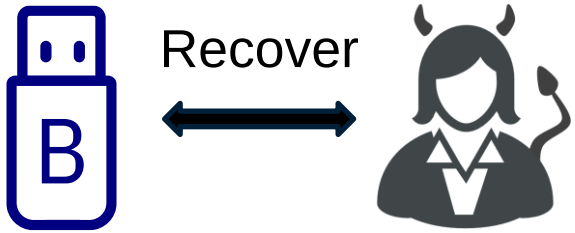
## Observe



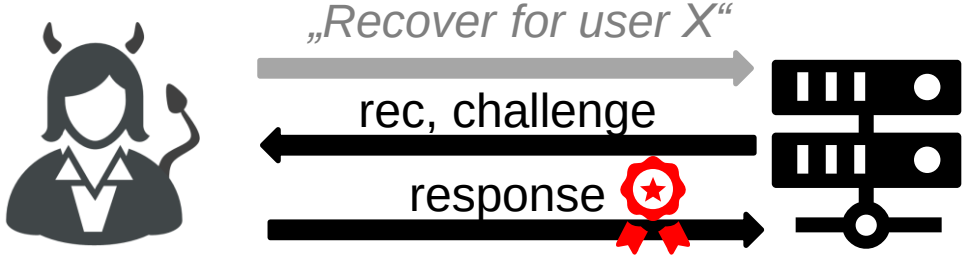

## Participate


## Leak

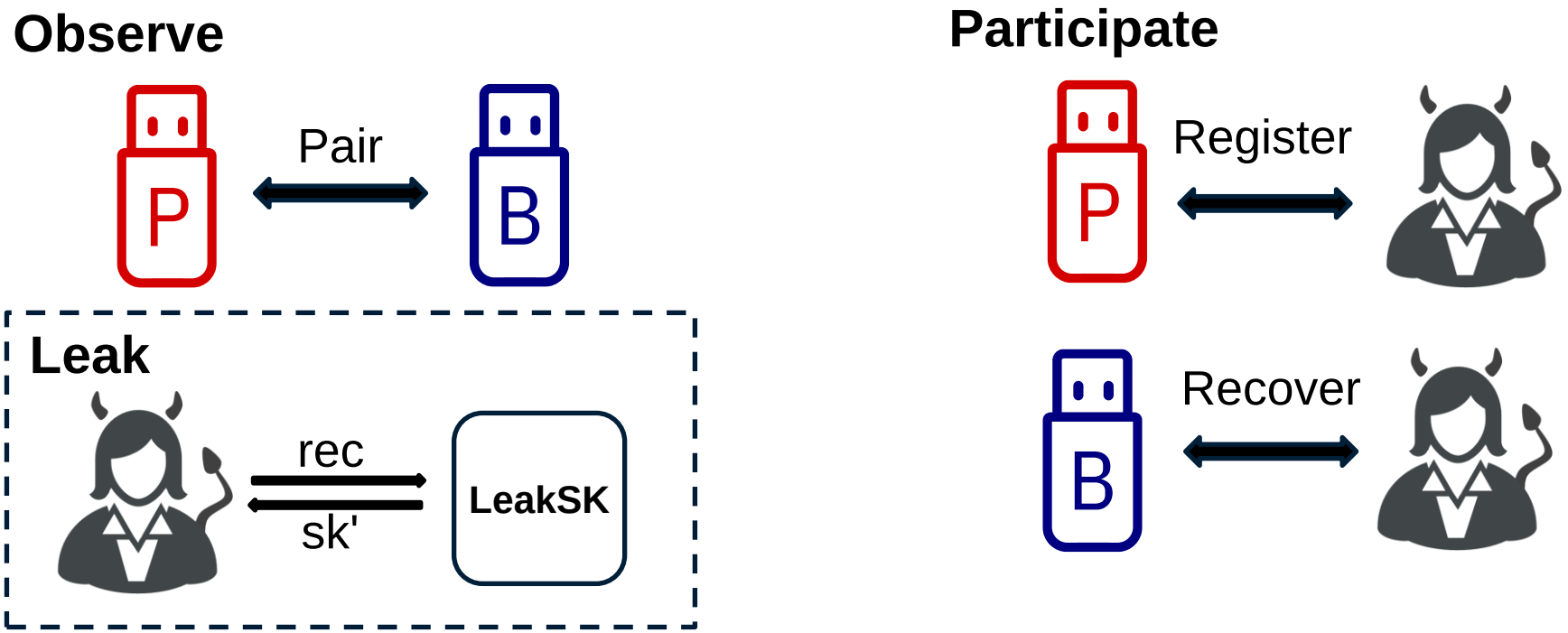


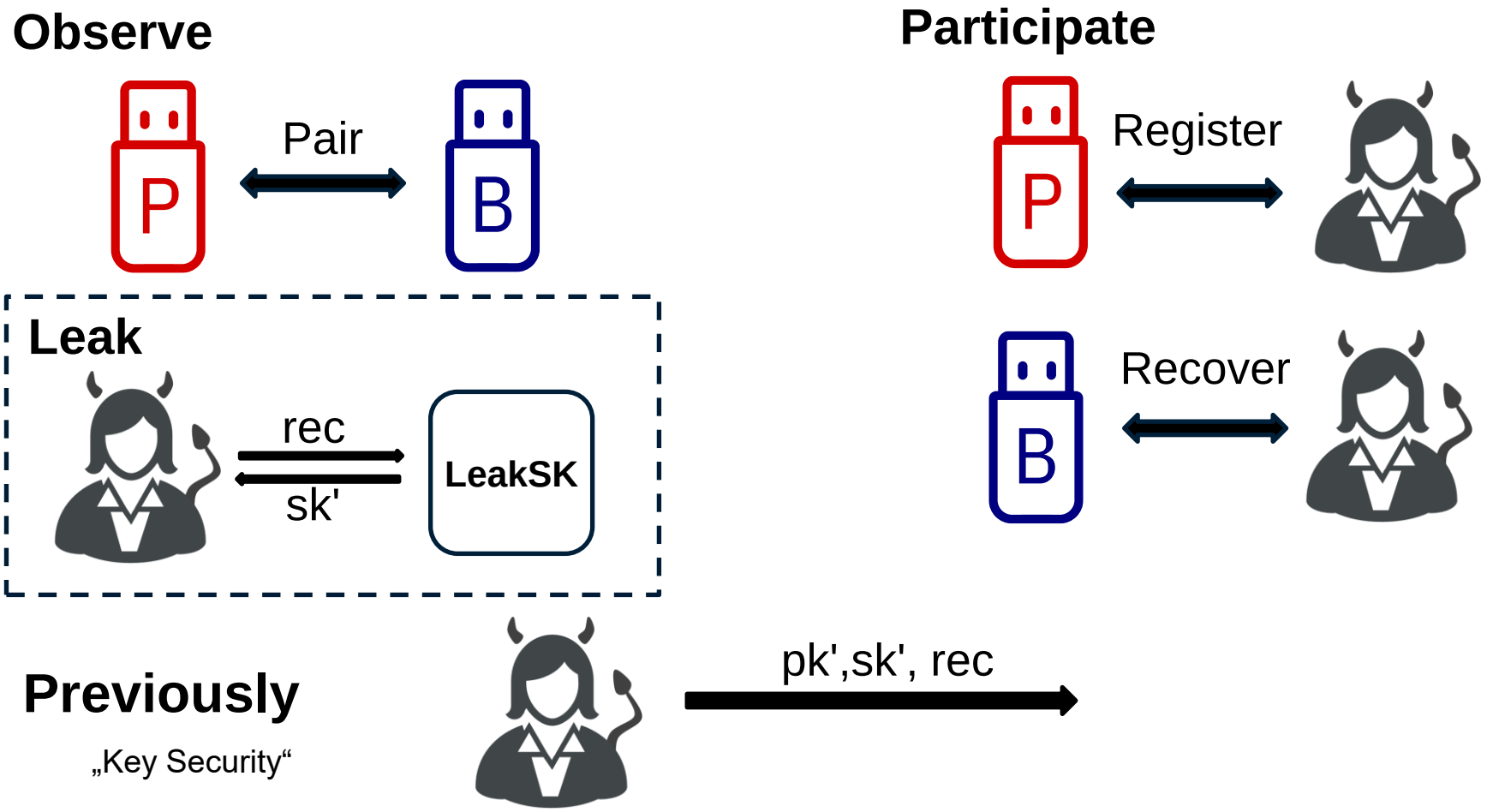

## Attack



# Previous Authentication Security

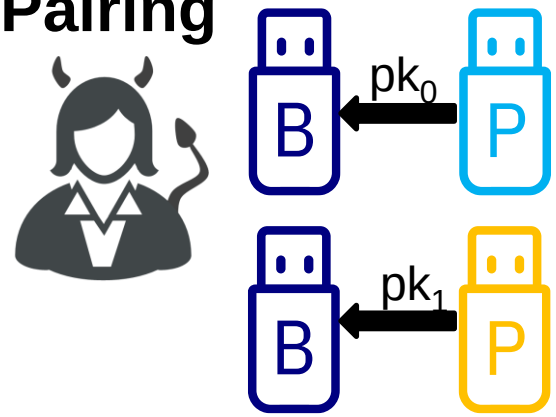


# Previous Authentication Security



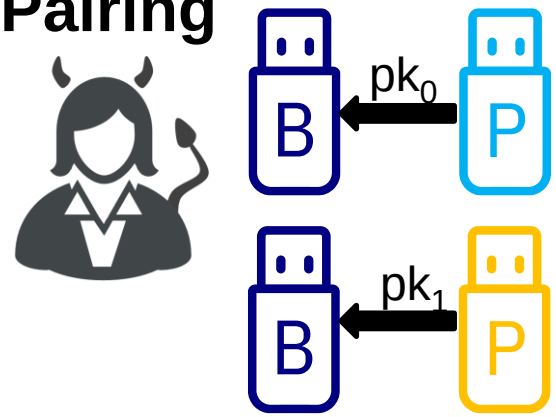
# Unlinkablity

**Observe Pairing**

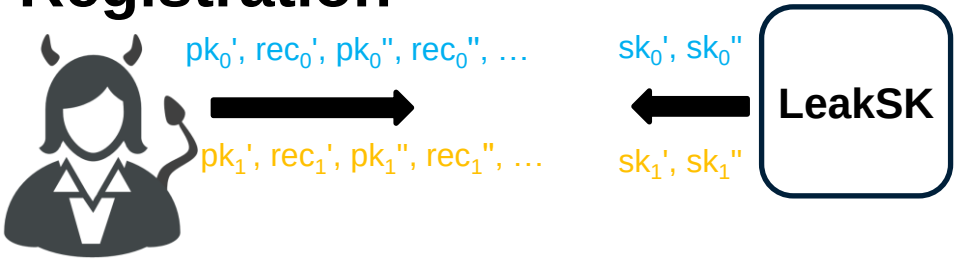


# Unlinkablity

## Observe Pairing

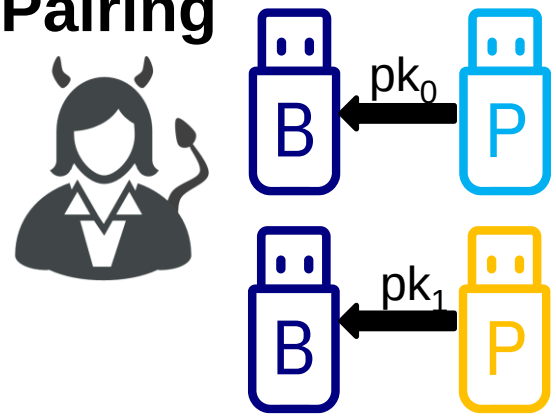



## Simulate Registration

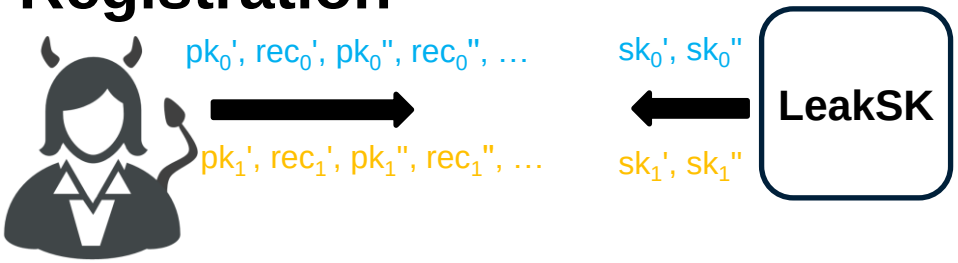


# Unlinkablity

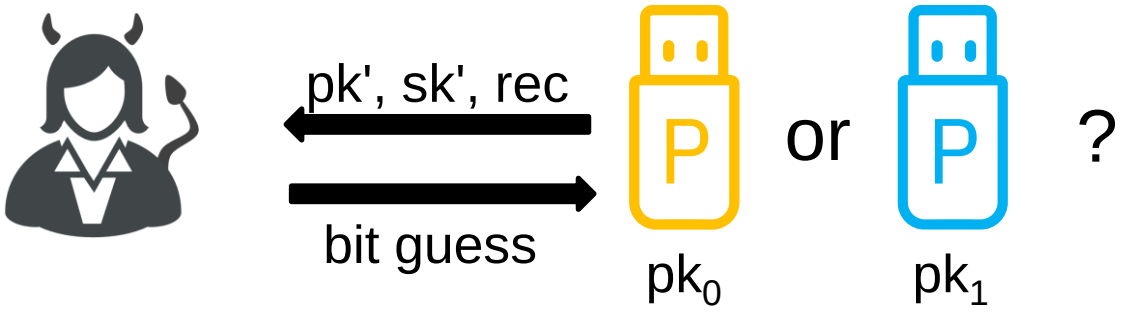
## Observe Pairing




## Simulate Registration

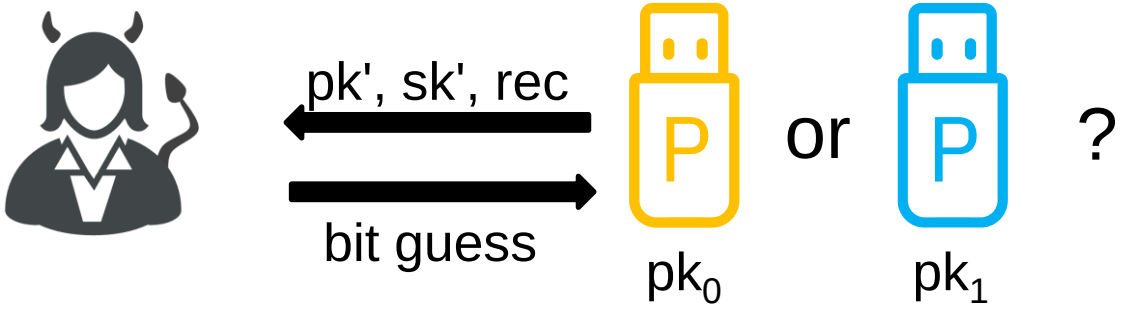



## Challenge Query

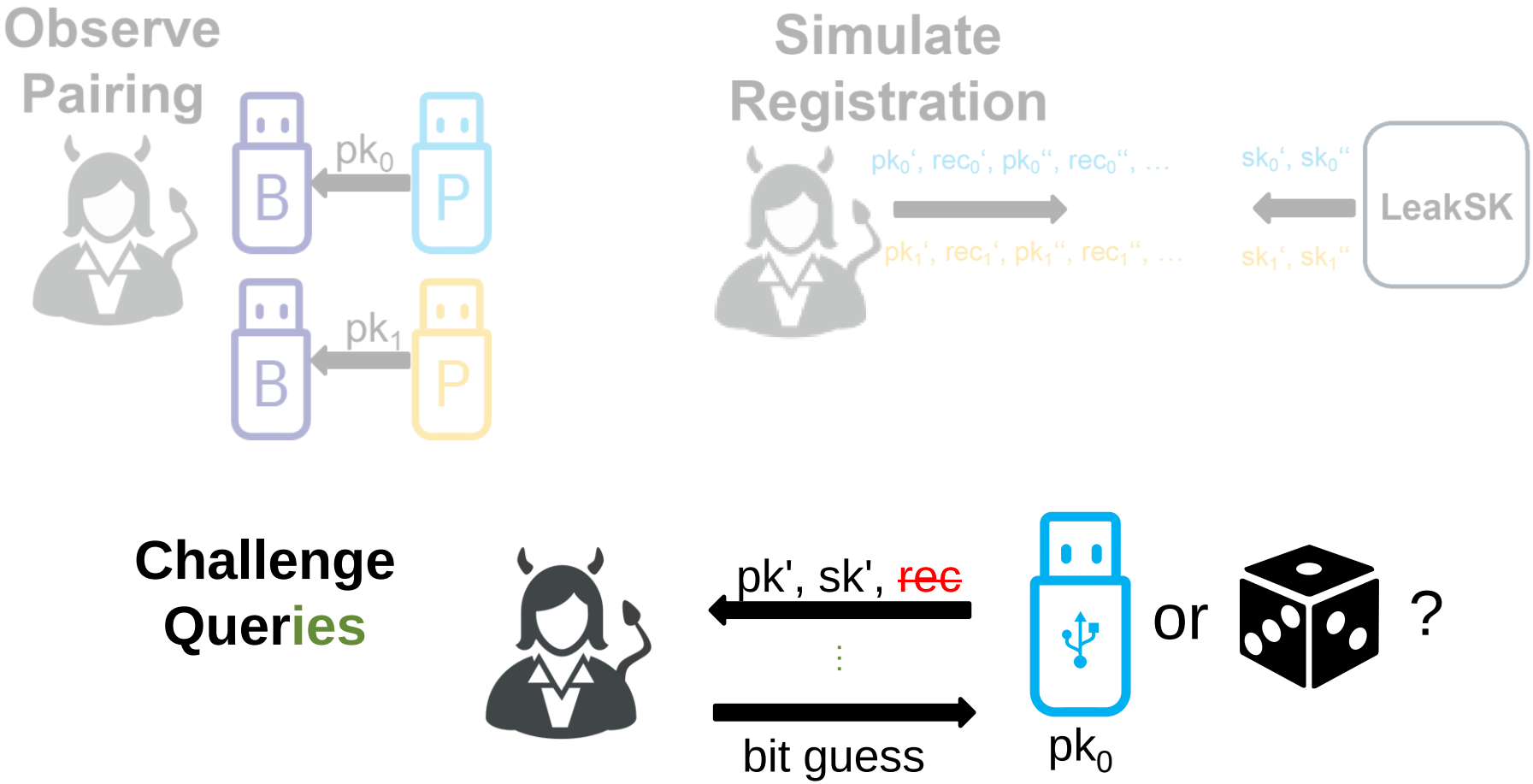


# Previous Unlinkablity


**Challenge Query**



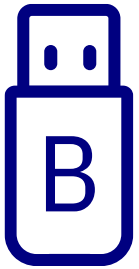
# Previous Unlinkablity





# Our Construction

# Our Construction: Pairing (done once)

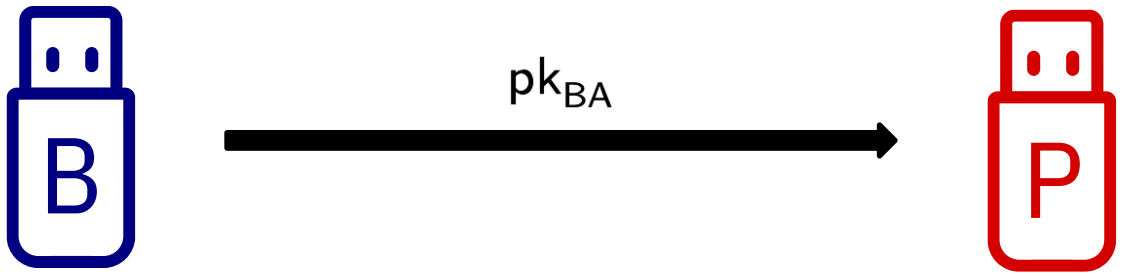




$\underline{\mathsf{KGen}(\mathsf{pp}):}$

1 $(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}}) \leftarrow\!\$\ \mathsf{KEM.KGen}(\mathsf{pp})$

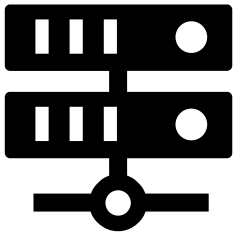
# Our Construction: Pairing (done once)




KGen(pp):

1 $(pk_{BA}, sk_{BA}) \leftarrow\$ \text{KEM.KGen}(pp)$

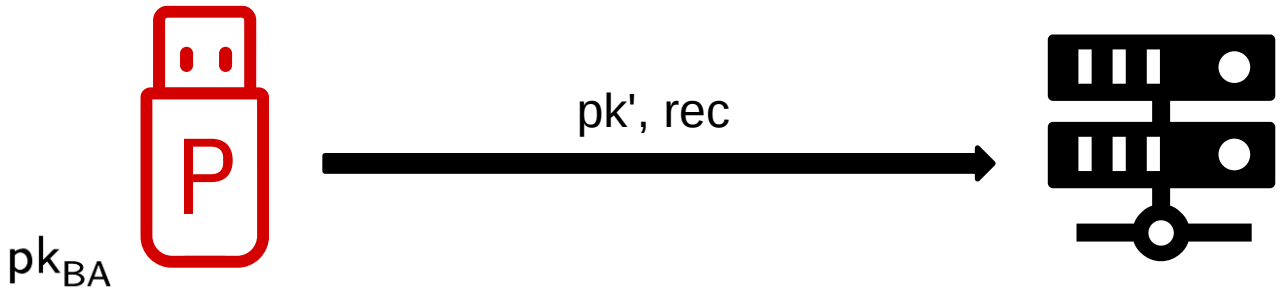
# Our Construction: Registration

$\underline{\mathsf{DerivePK}(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}}, aux):}$

1 $(c, K) \leftarrow\$ \mathsf{KEM.Encaps}(\mathsf{pk}_{\mathsf{BA}})$
2 $r \leftarrow \mathsf{KDF}(K, aux)$
3 $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Sig.KGen}(\mathsf{pp}; r)$
4 $\mathsf{rec} \leftarrow (c, aux)$

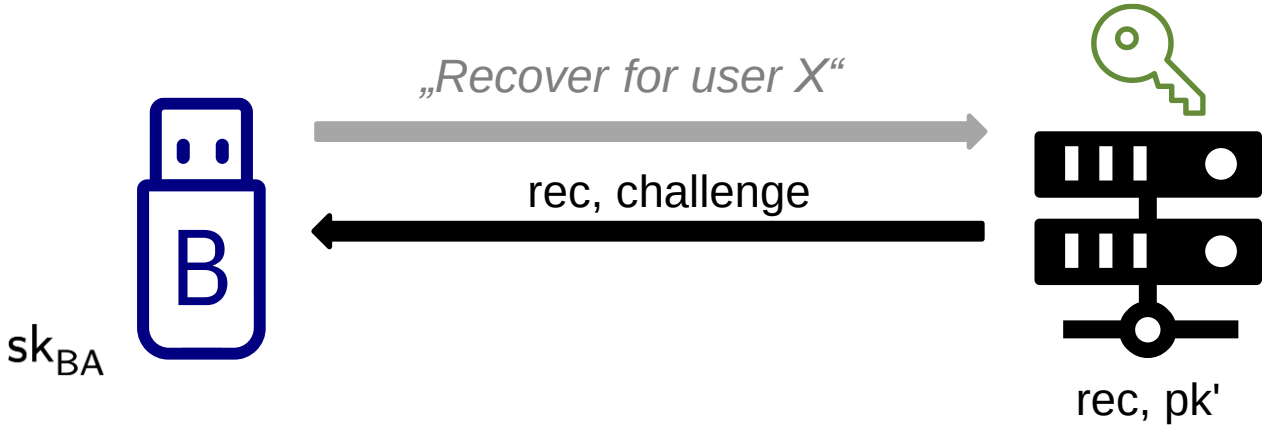
# Our Construction: Registration



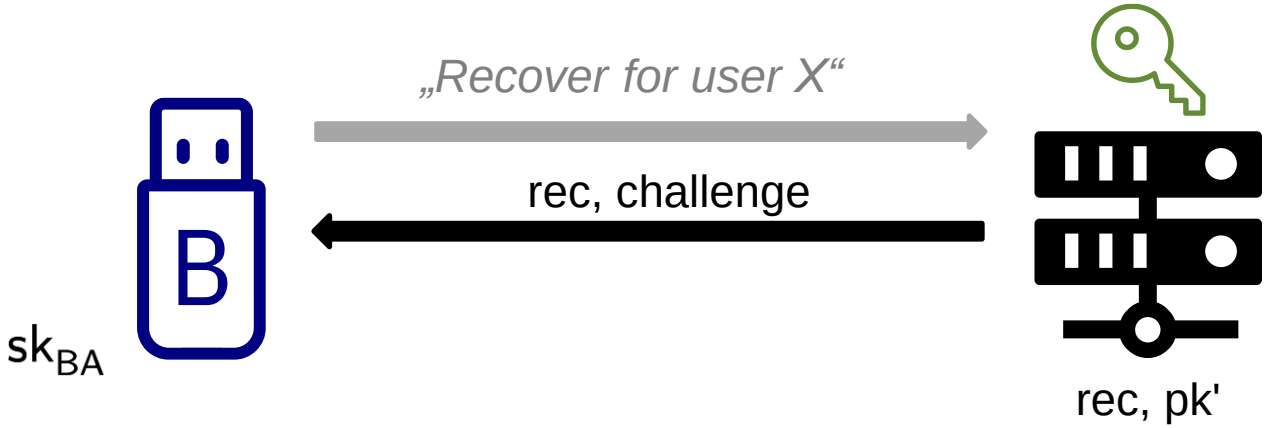

DerivePK$(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}}, aux)$:

1 $(c, K) \leftarrow\$ \mathsf{KEM.Encaps}(\mathsf{pk}_{\mathsf{BA}})$
2 $r \leftarrow \mathsf{KDF}(K, aux)$
3 $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Sig.KGen}(\mathsf{pp}; r)$
4 $\mathsf{rec} \leftarrow (c, aux)$

# Our Construction: Recovery



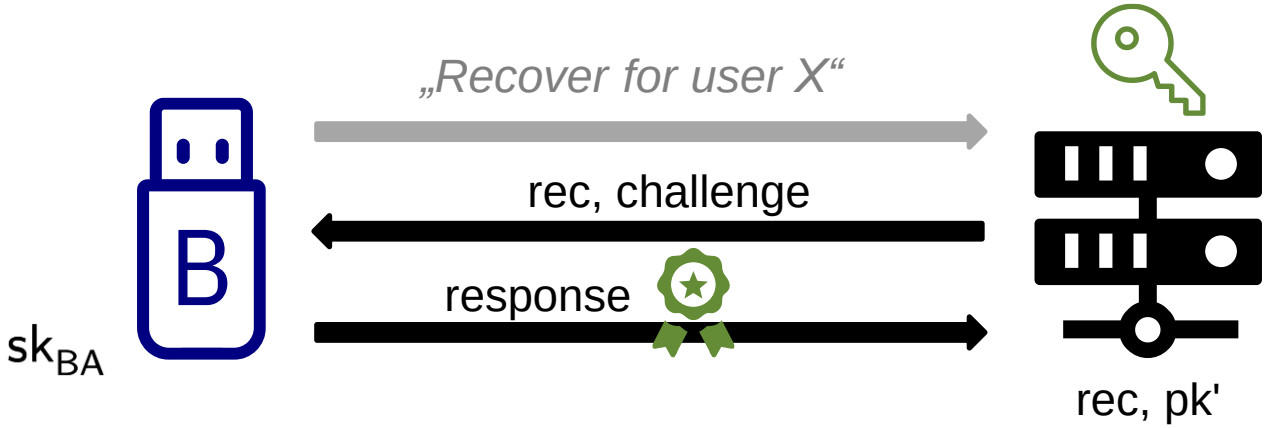
# Our Construction: Recovery




$\underline{\mathsf{DeriveSK}(\mathsf{pp}, \mathsf{sk}_{\mathsf{BA}}, \mathsf{rec}):}$

1 $(c, aux) \leftarrow \mathsf{rec}$
2 $K \leftarrow \mathsf{KEM.Decaps}(\mathsf{sk}_{\mathsf{BA}}, c)$
3 $r \leftarrow \mathsf{KDF}(K, aux)$
4 $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Sig.KGen}(\mathsf{pp}; r)$

# Our Construction: Recovery




DeriveSK$(pp, sk_{BA}, rec)$:

1 $(c, aux) \leftarrow$ rec
2 $K \leftarrow$ KEM.Decaps$(sk_{BA}, c)$
3 $r \leftarrow$ KDF$(K, aux)$
4 $(pk', sk') \leftarrow$ Sig.KGen$(pp; r)$

# Post-Quantum Asynchronous Remote Key Generation for FIDO2



## Security and PQ Instantiation

# Authentication Security

**Theorem 1** *Let* ARKG *be the generic instantiation of* ARKG *as given in Figure 4,* KEM *be an* IND-CCA *secure and* $\epsilon$*-correct KEM scheme,* Sig *be an* EUF-CMA *secure signature scheme and* KDF *a secure key derivation function modeled as a* PRF. *Then* ARKG *provides* $\epsilon$*-correctness and authentication security as defined in Definition 2. More precisely, for any QPT adversary* $\mathcal{A}$ *against* AUTH, *there exist QPT algorithms* $\mathcal{B}_1, \mathcal{B}_2$, *and* $\mathcal{B}_3$ *with approximately the same running time as* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\text{auth}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \;\leq\; q \cdot \Big(\epsilon + \mathsf{Adv}^{\text{ind-cca}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{prf}}_{\mathsf{KDF},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig},\mathcal{B}_3}(\lambda)\Big)$$

*where* $q$ *is the maximum number of calls to the* DERIVEPK *oracle.*

# Unlinkability

**Theorem 2** *Let* ARKG *be the instantiation of* ARKG *as shown in Figure 4 and* KEM *be an* ANON-CCA *secure* KEM *scheme. Then* ARKG *provides unlinkability security as described in Definition 3. More precisely, for any QPT adversary* $\mathcal{A}$ *against* UNL *there exists a QPT algorithm* $\mathcal{B}$ *with approximately the same running time as* $\mathcal{A}$*, such that*

$$\mathsf{Adv}^{\mathrm{unl}}_{\mathsf{ARKG},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{anon-cca}}_{\mathsf{KEM},\mathcal{B}}(\lambda).$$

**ANON-CCA / (strong) anonymity)**
the ciphertext obtained during encapsulation does not leak any information on the public key used during the encapsulation operation, even in the presence of a decapsulation oracle.

# Post-Quantum Instantiation

**Setup** $(1^\lambda)$:

1 **return** $\mathsf{pp} = (\mathsf{KEM}, \mathsf{KDF}, \mathsf{Sig})$

**KGen**$(\mathsf{pp})$:

1 $(\mathsf{pk}_{\mathsf{BA}}, \mathsf{sk}_{\mathsf{BA}}) \leftarrow\!\!\$\ \mathsf{KEM.KGen}(\mathsf{pp})$

**DerivePK**$(\mathsf{pp}, \mathsf{pk}_{\mathsf{BA}}, aux)$:

1 $(c, K) \leftarrow\!\!\$\ \mathsf{KEM.Encaps}(\mathsf{pk}_{\mathsf{BA}})$
2 $r \leftarrow \mathsf{KDF}(K, aux)$
3 $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Sig.KGen}(\mathsf{pp}; r)$
4 $\mathsf{rec} \leftarrow (c, aux)$

**DeriveSK**$(\mathsf{pp}, \mathsf{sk}_{\mathsf{BA}}, \mathsf{rec})$:

1 $(c, aux) \leftarrow \mathsf{rec}$
2 $K \leftarrow \mathsf{KEM.Decaps}(\mathsf{sk}_{\mathsf{BA}}, c)$
3 $r \leftarrow \mathsf{KDF}(K, aux)$
4 $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Sig.KGen}(\mathsf{pp}; r)$

- ML-KEM provides IND-CCA and ANON-CCA security
- All (secure) PQ signatures provide EUF-CMA security
- KDF is symmetric primitive, use sufficient key length

# Conclusion

- Security games tailored to FIDO2 setting
- Generic construction from standard primitives
- Instantiable in PQ setting


**Thank you!**

https://ia.cr/2023/1275