Robust Password-Protected Secret Sharing

Michel Abdalla, Mario Cornejo,
Anca Niţulescu, David Pointcheval

École Normale Supérieure, CNRS and INRIA, Paris, France

ENS

cnrs dépasser les frontières

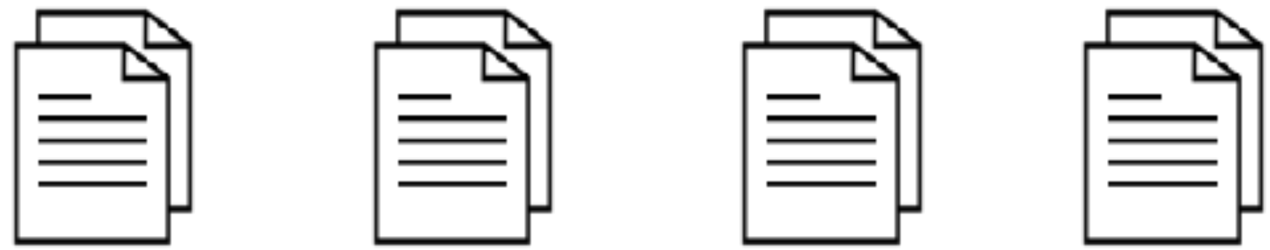
Inria INVENTORS FOR THE DIGITAL WORLD

PSL RESEARCH UNIVERSITY

PPSS: Motivation

PPSS: Motivation

Cloud provider



taxes

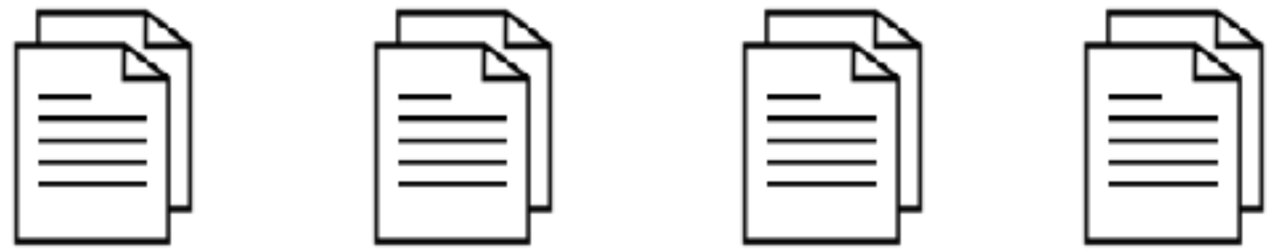
medical records

paychecks

top secret documents

PPSS: Motivation

Cloud provider



taxes

medical records

paychecks

top secret documents

Everyone might have access to the data

PPSS: Motivation

Cloud provider

taxes

medical records

paychecks

top secret documents

PPSS: Motivation

Cloud provider

taxes

medical records

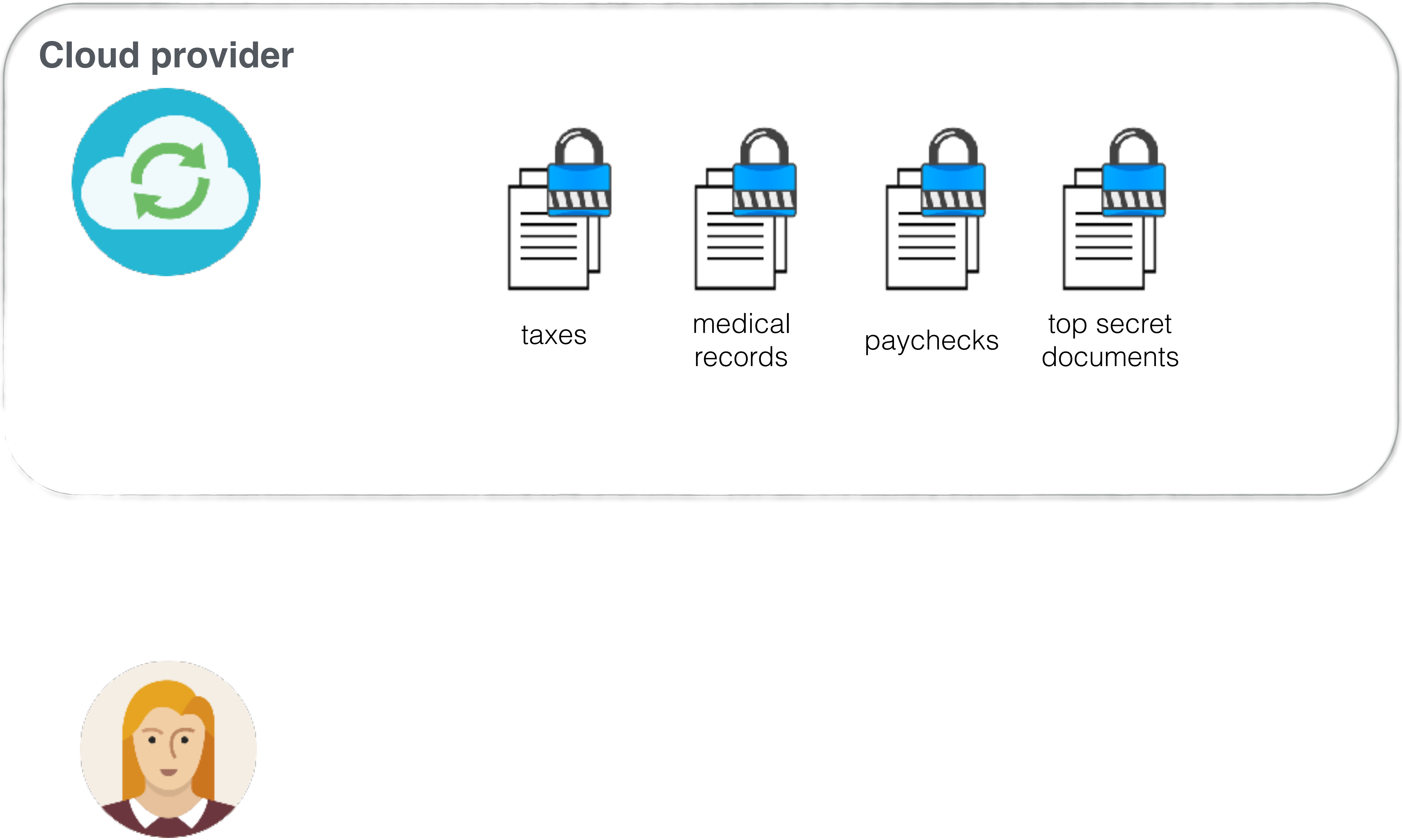
paychecks

top secret documents

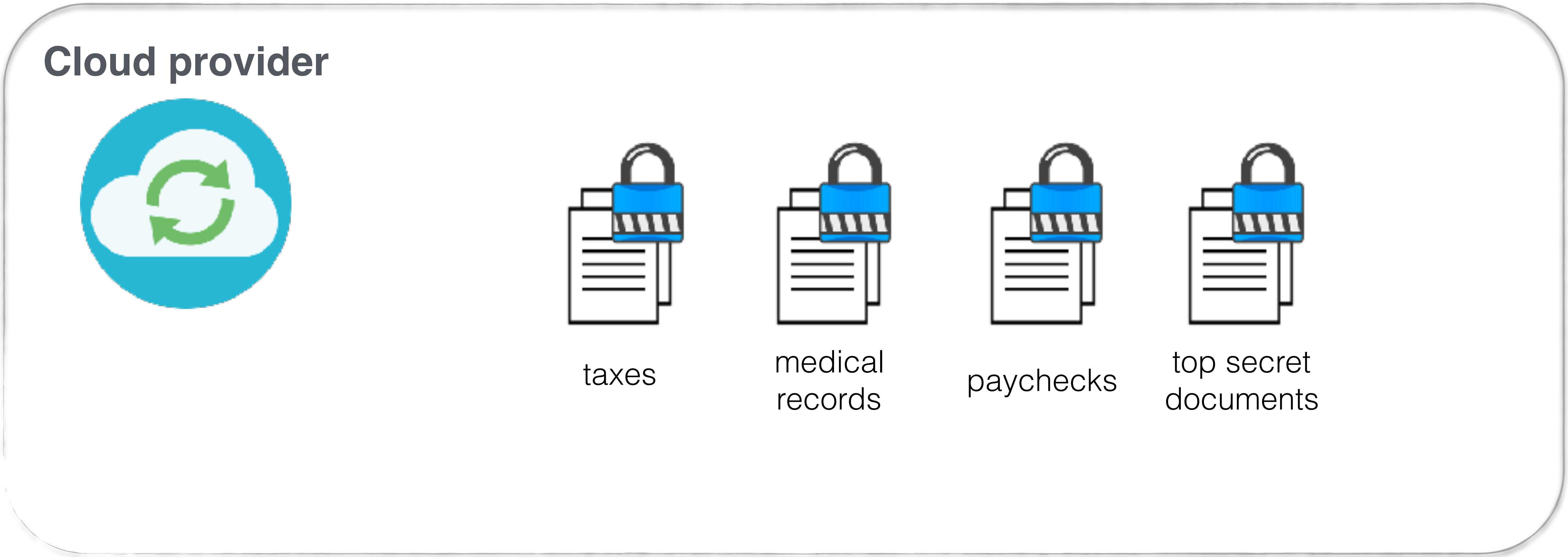
Provider *still* has access to the data

PPSS: Motivation

PPSS: Motivation



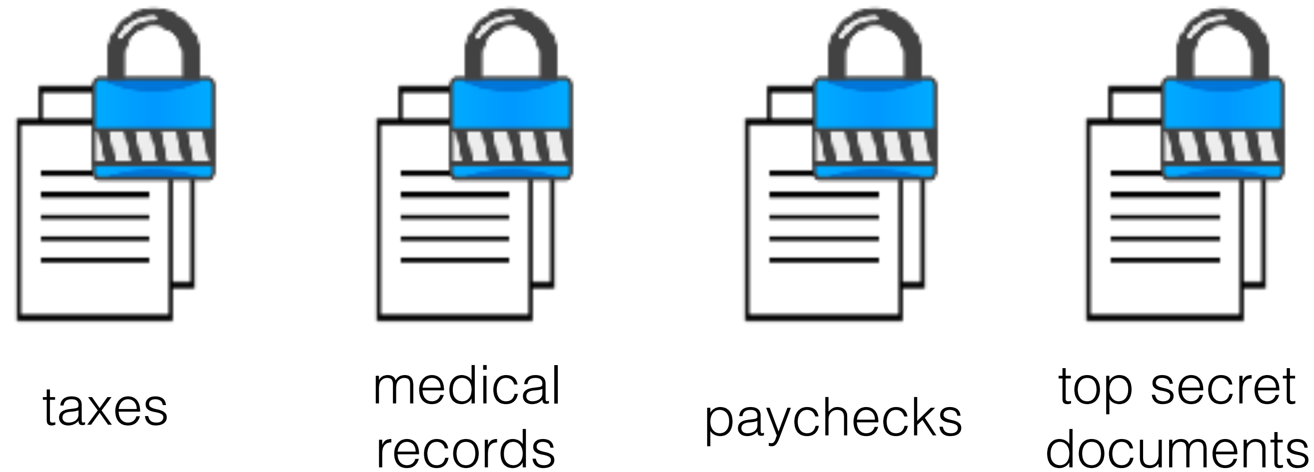
PPSS: Motivation



- We can remember just low-entropy passwords (and not too many).
- Humans cannot remember large secret keys.
- **Provider/authorities might perform an offline dictionary attack.**

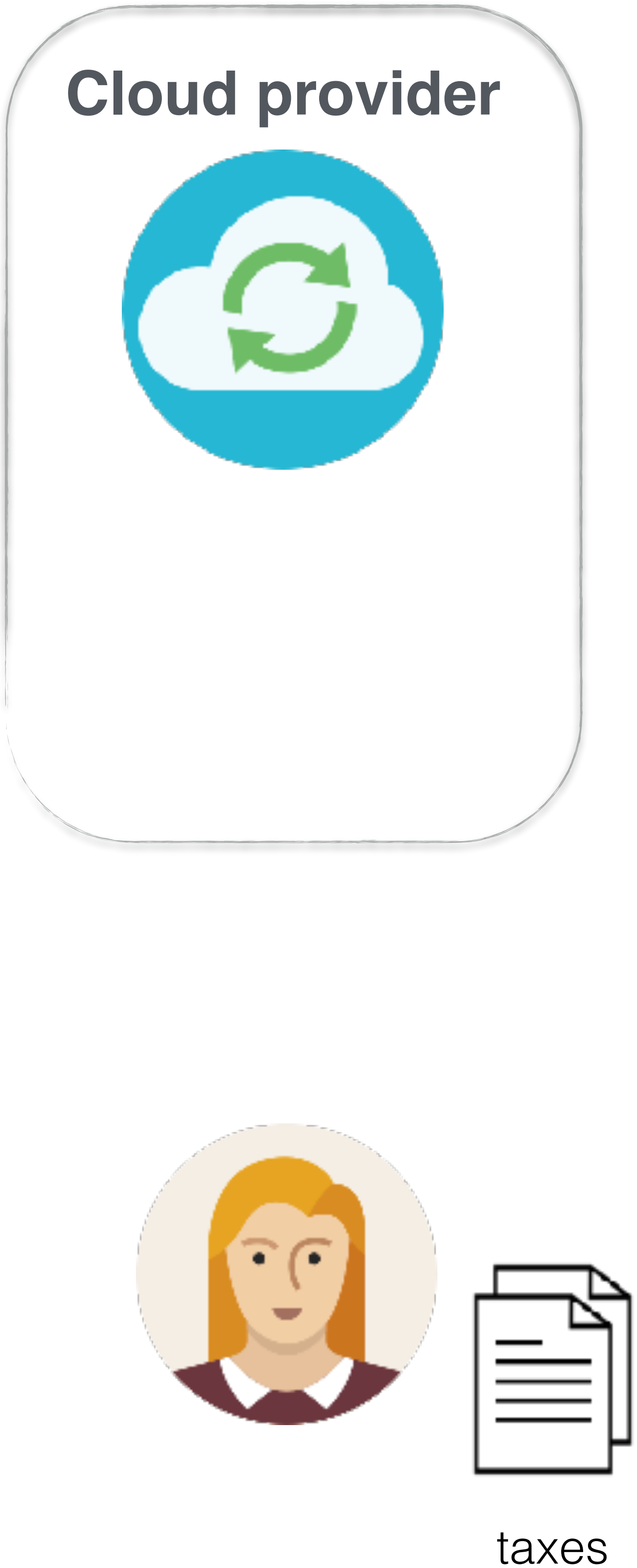
PPSS: Motivation

Cloud provider

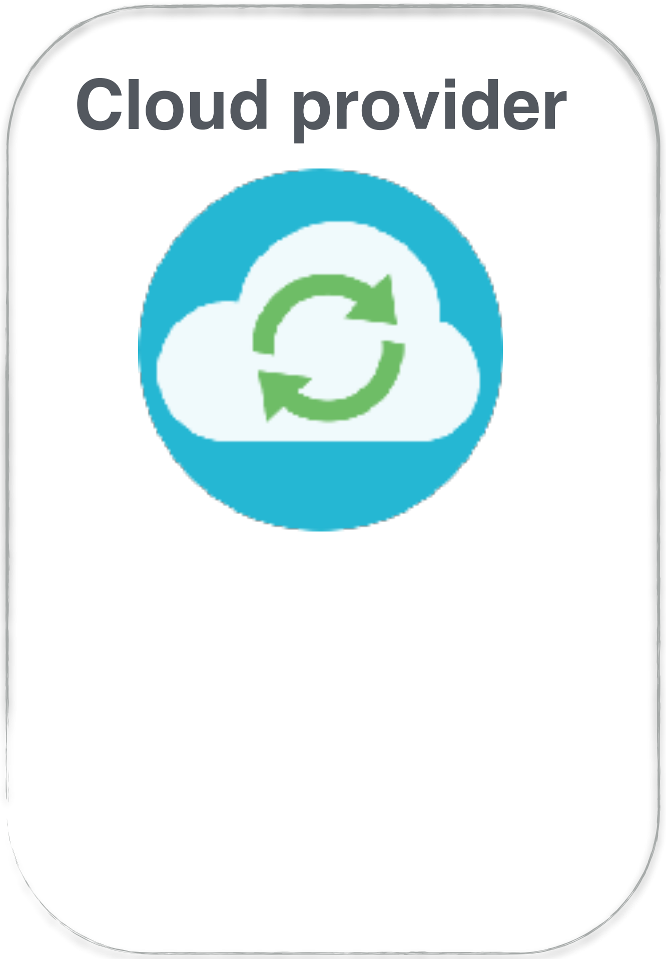


- USB Tokens might not be always available.
- Tokens might fall into the wrong hands.
- **Large keys give better security.**

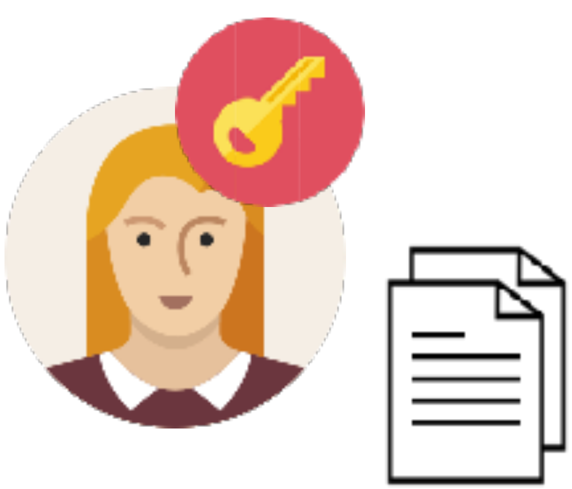
PPSS: Password-Protected Secret Sharing



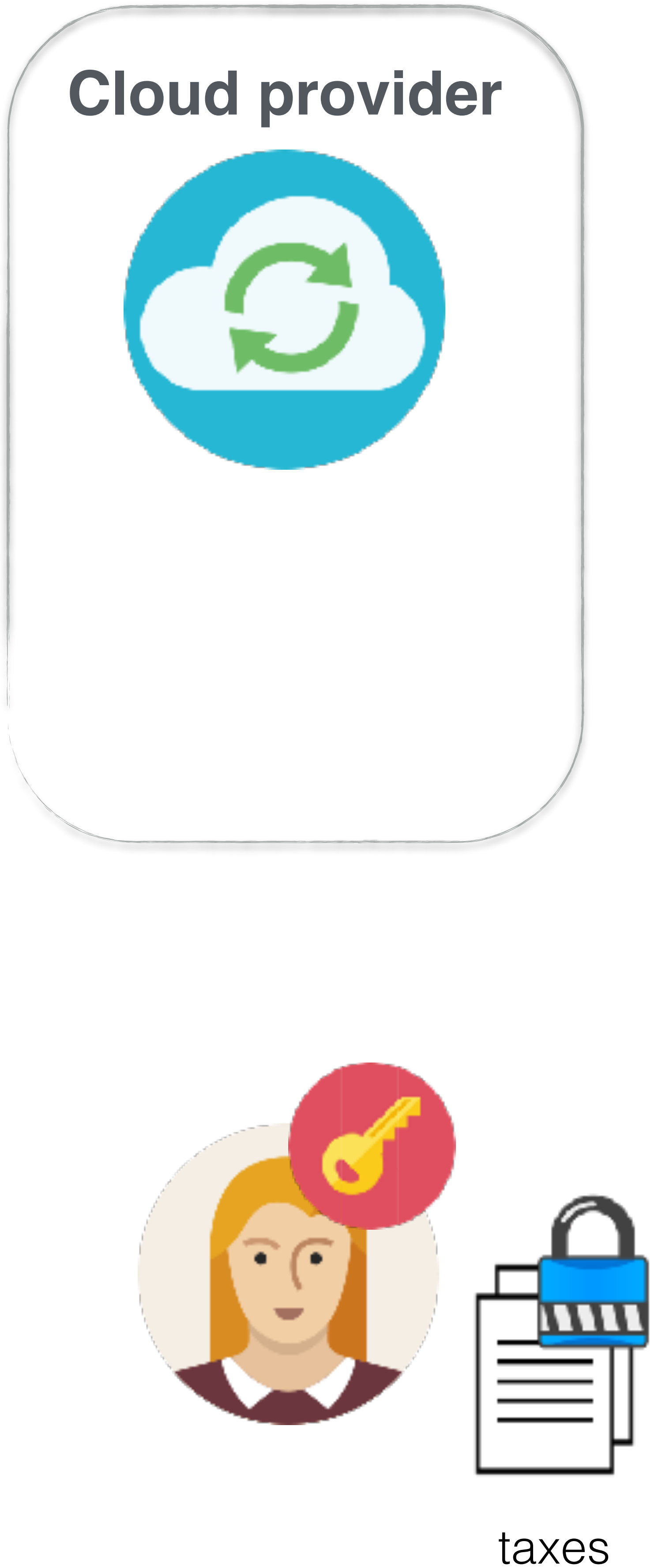
PPSS: Password-Protected Secret Sharing



- User creates a cryptographic key.

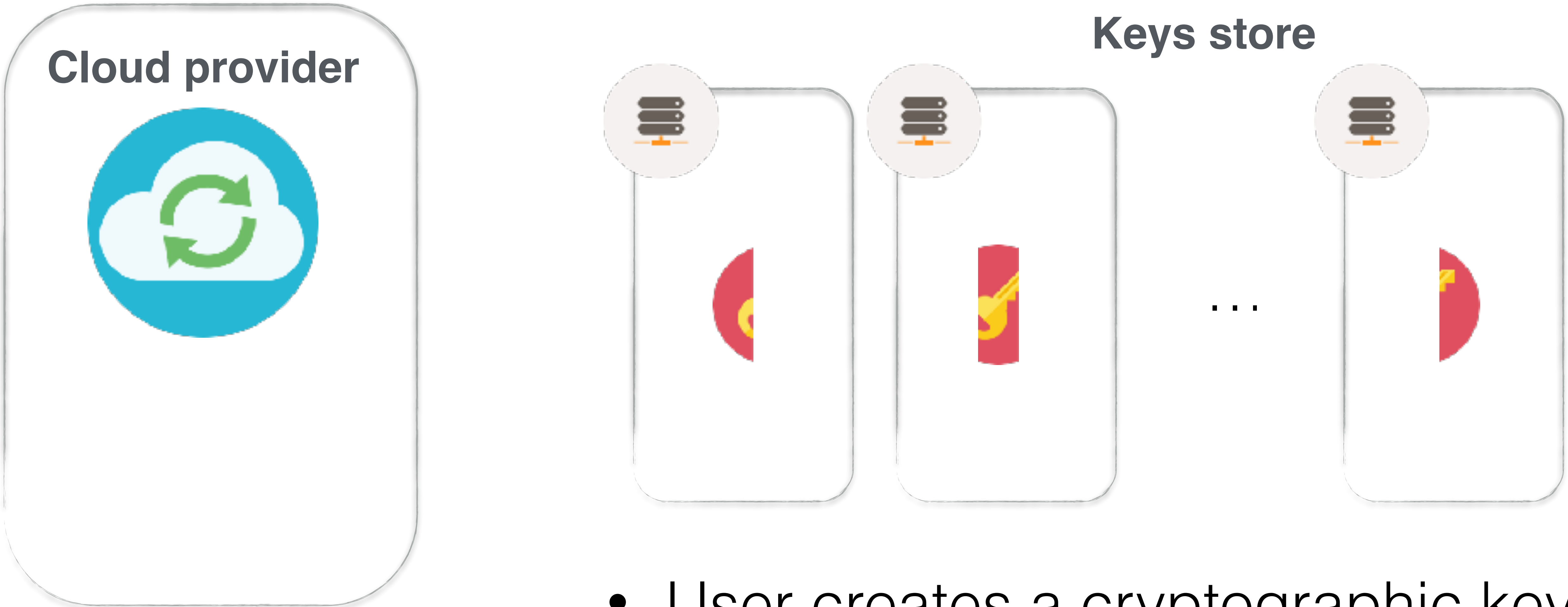


PPSS: Password-Protected Secret Sharing

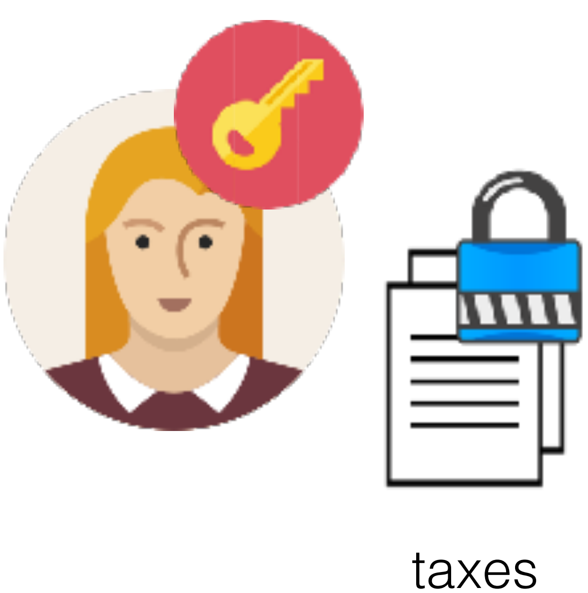


- User creates a cryptographic key.
- Encrypts her data using this key.

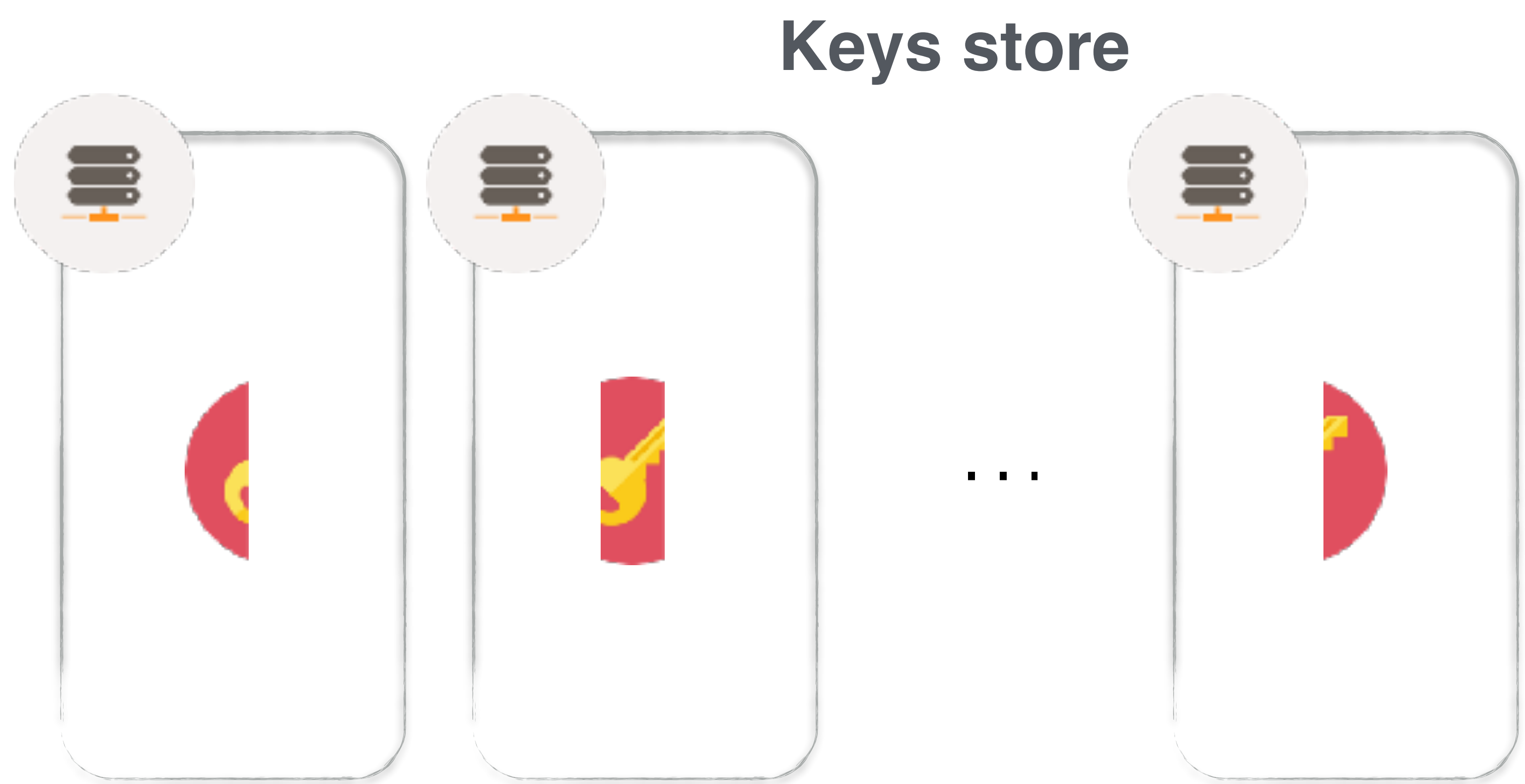
PPSS: Password-Protected Secret Sharing



- User creates a cryptographic key.
- Encrypts her data using this key.
- Stores her secret key into n servers by using her password and some public information.

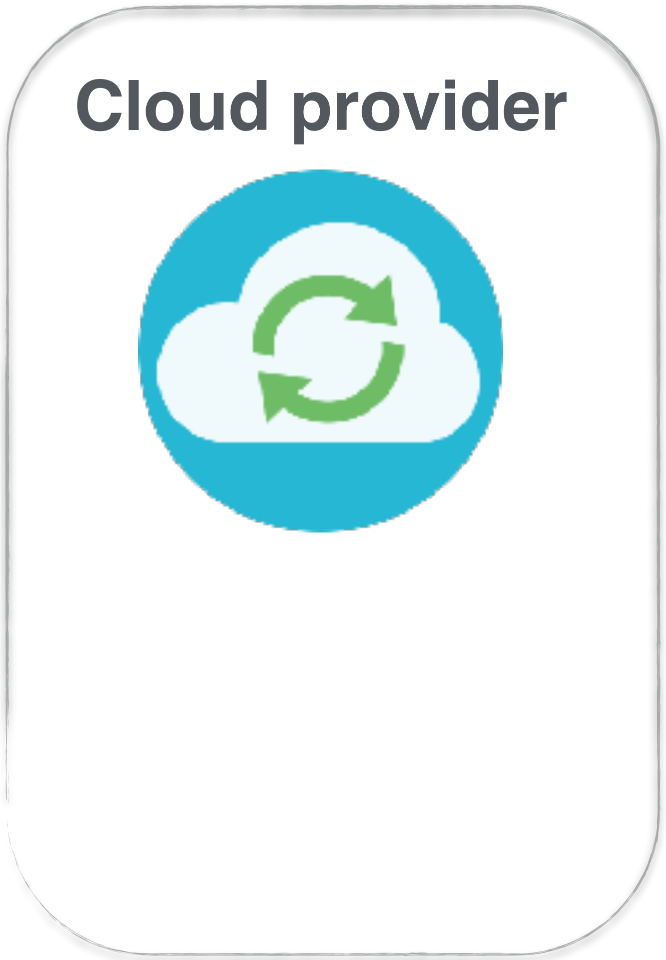


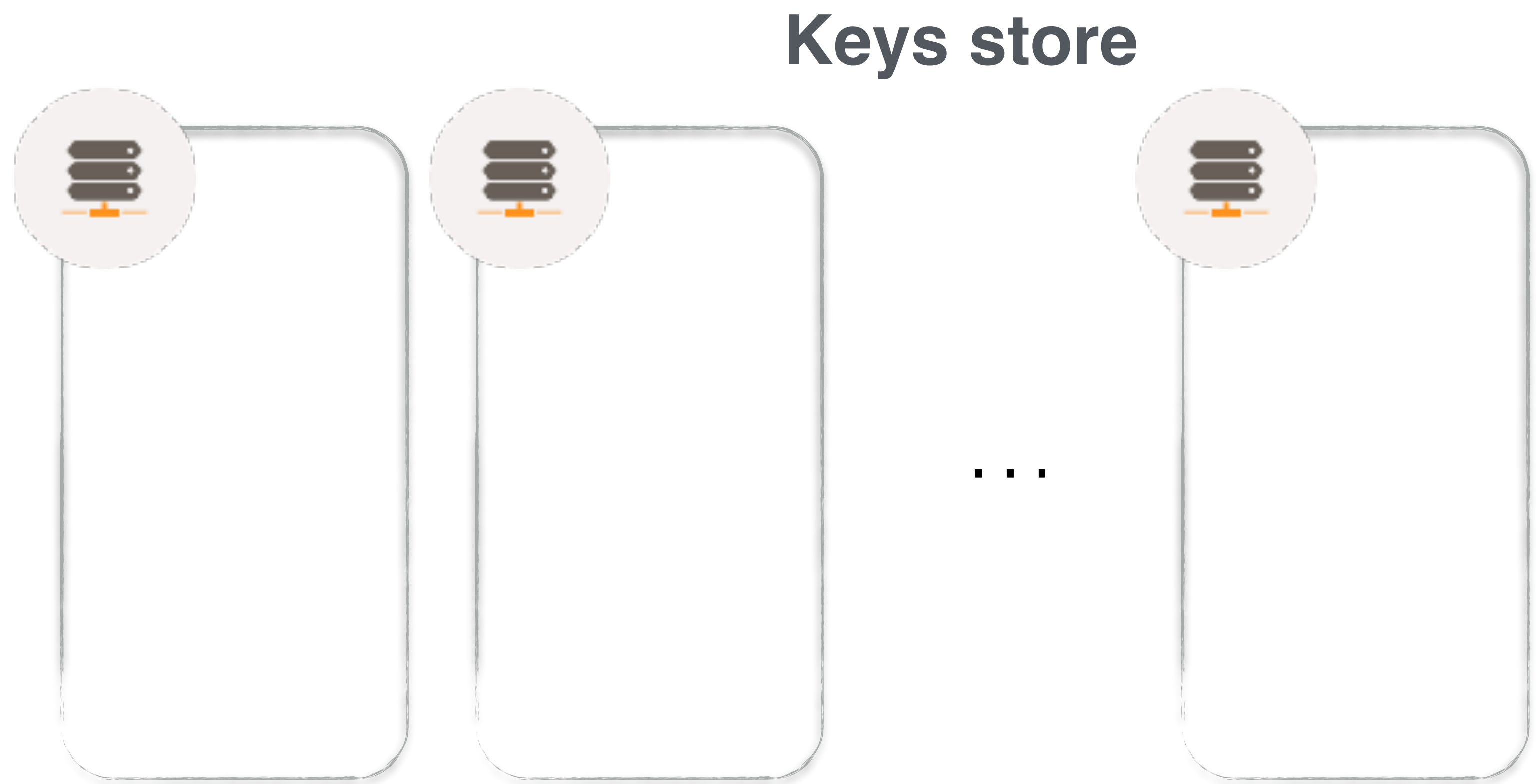
PPSS: Password-Protected Secret Sharing



- User creates a cryptographic key.
- Encrypts her data using this key.
- Stores her secret key into n servers by using her password and some public information.
- Stores the data into the provider.

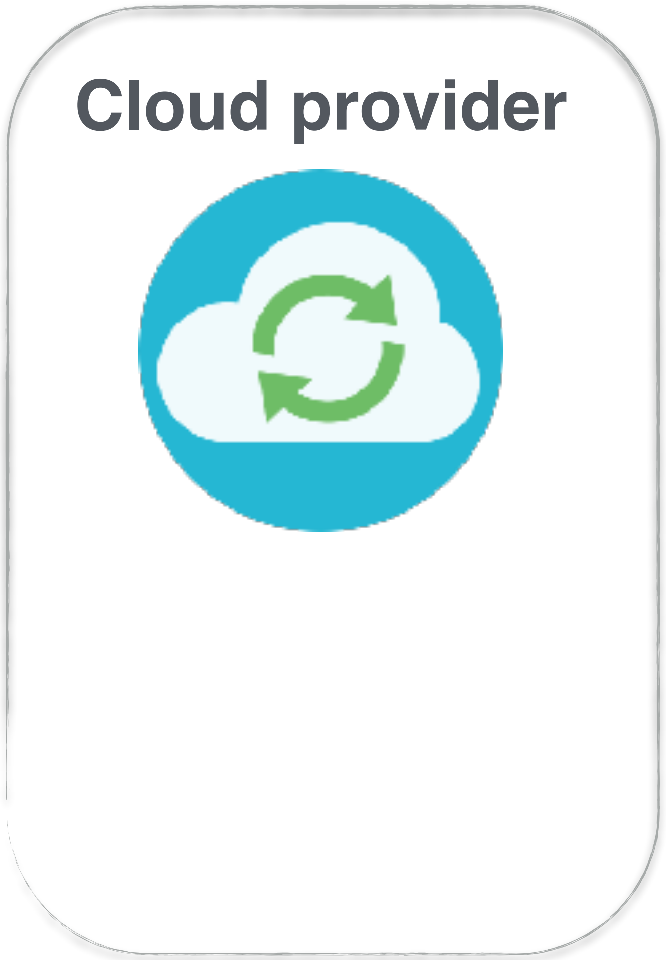
PPSS: Password-Protected Secret Sharing

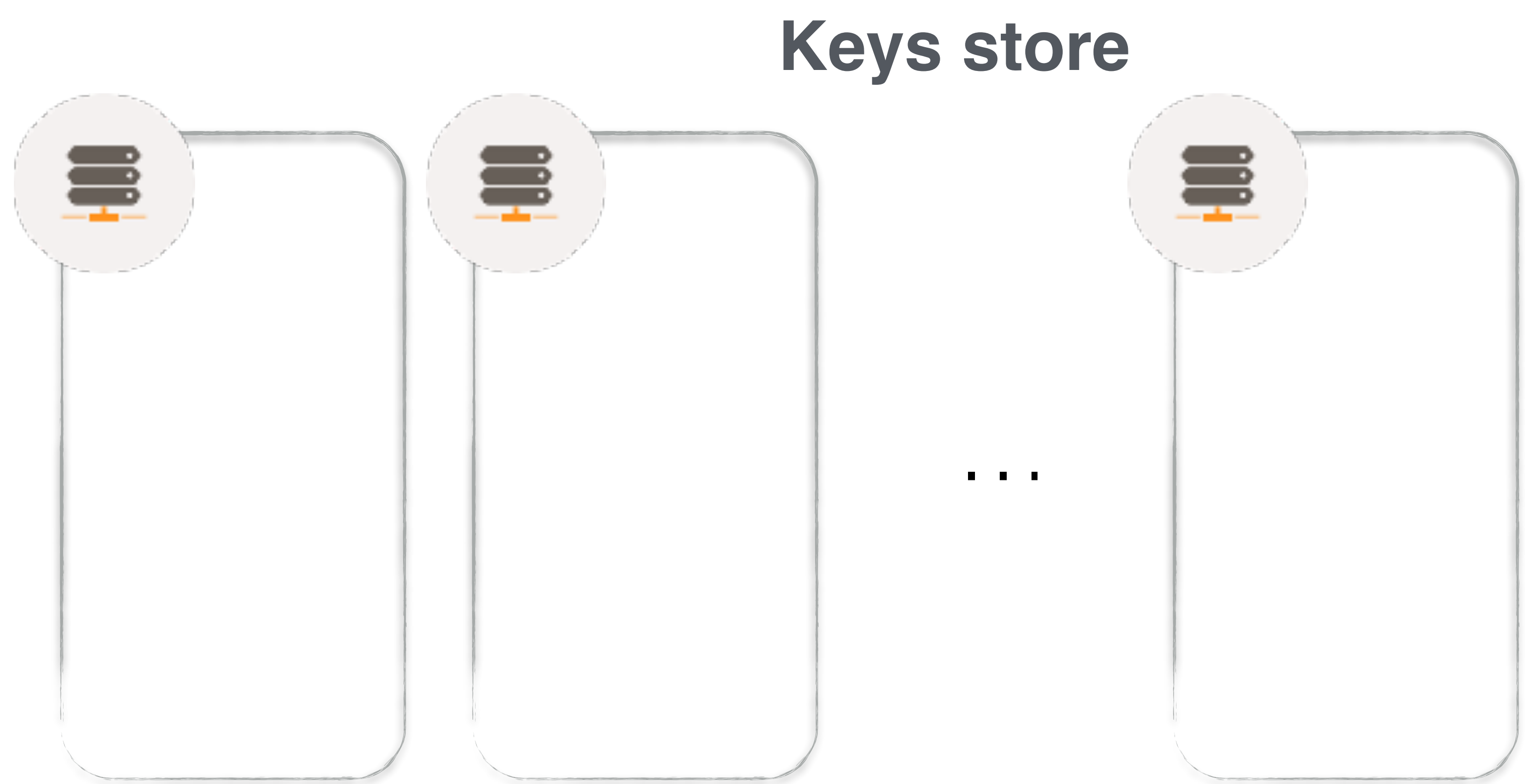




- After $t+1$ interactions using her password, the user can recover her secret key

PPSS: Password-Protected Secret Sharing





- After $t + 1$ interactions using her password, the user can recover her secret key

PPSS: Properties

- A PPSS scheme defines two steps:

Initialization: Secret & password are processed

Reconstruction: The user can recover the secret by interacting with a subset of $t+1$ servers.

- Additional properties:

Soundness: Even if the adversary cannot make the user recover a different secret.

Robustness: The recovery is guaranteed if there are $t+1$ non-corrupt servers.

PPSS: Instantiations of PPSS

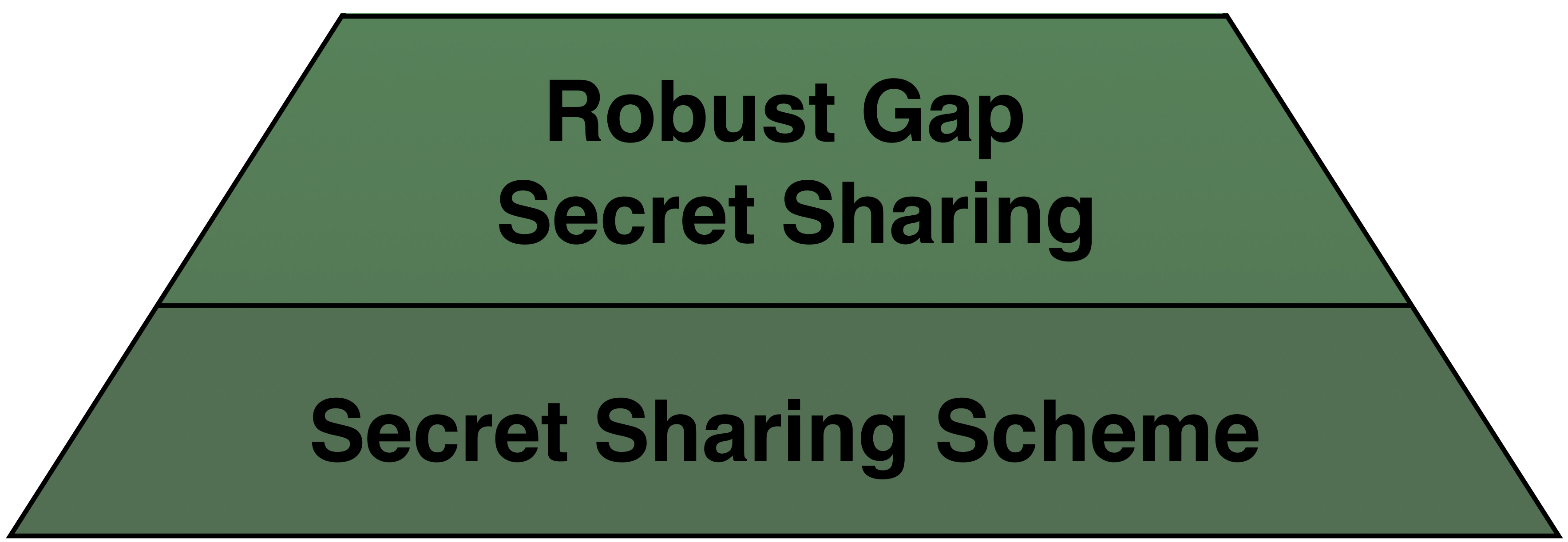
Scheme	Messages	Client	inter-server	Robust	ZKP
BJSL11	4	PKI	PKI	No	Costly
CLLN14	10	Std	PKI	No	Costly
JKK14	2	CRS	None	Yes	Costly
JKKX16	2	CRS	None	No	No

PPSS: Instantiations of PPSS

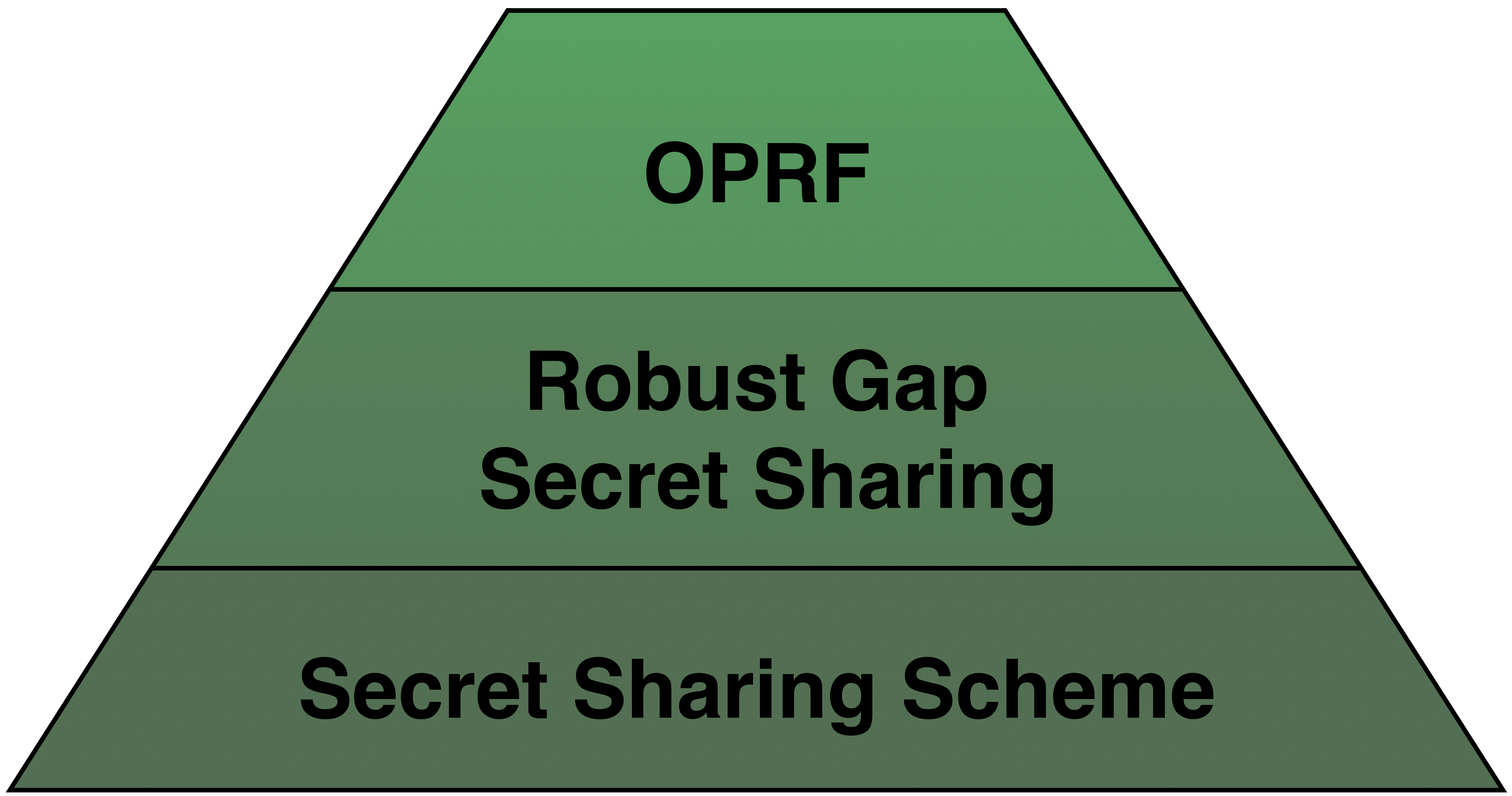
Scheme	Messages	Client	inter-server	Robust	ZKP
BJSL11	4	PKI	PKI	No	Costly
CLLN14	10	Std	PKI	No	Costly
JKK14	2	CRS	None	Yes	Costly
JKKX16	2	CRS	None	No	No
This work	2	CRS	None	Yes	No

Robust Password-Protected Secret Sharing

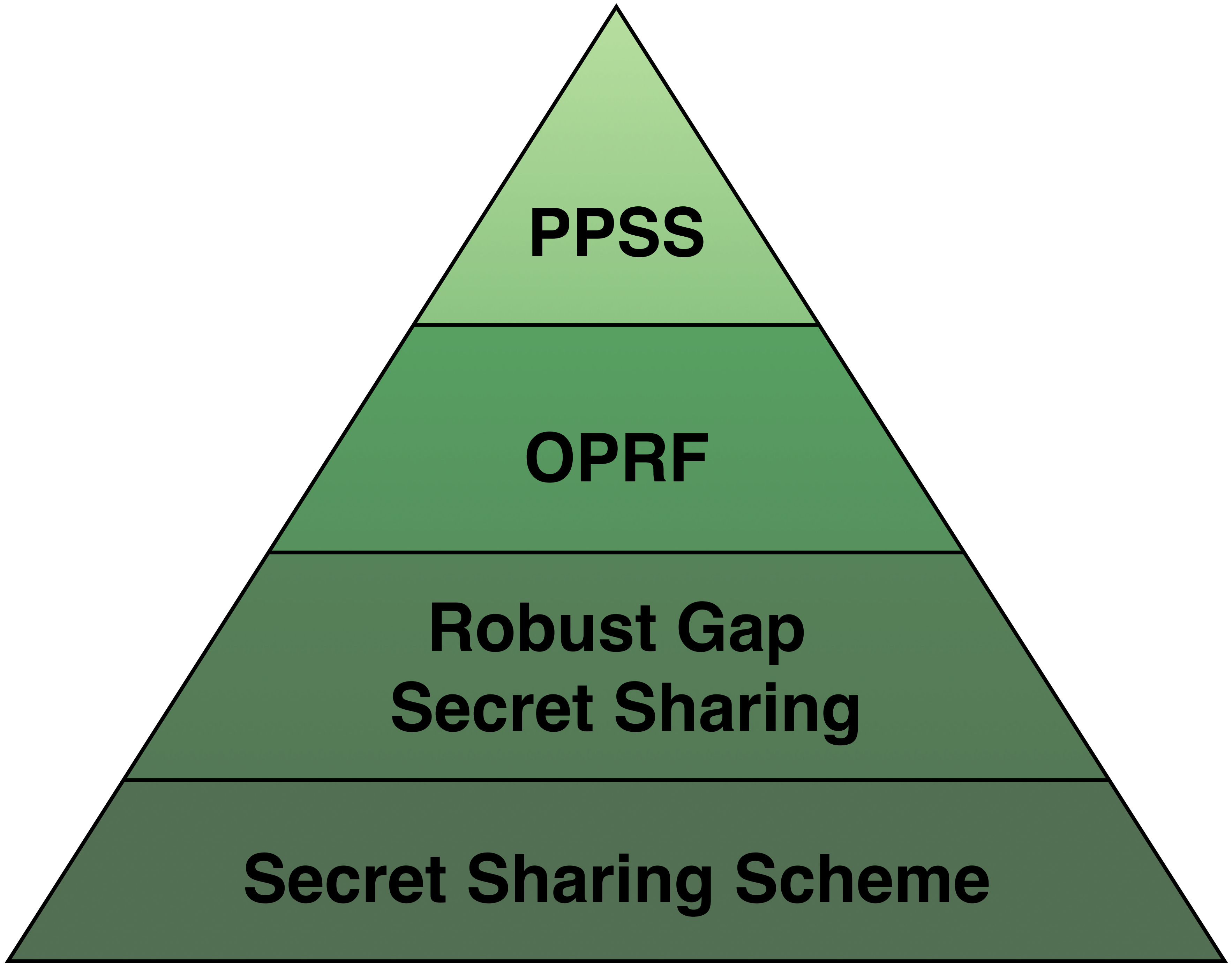
Robust Password-Protected Secret Sharing



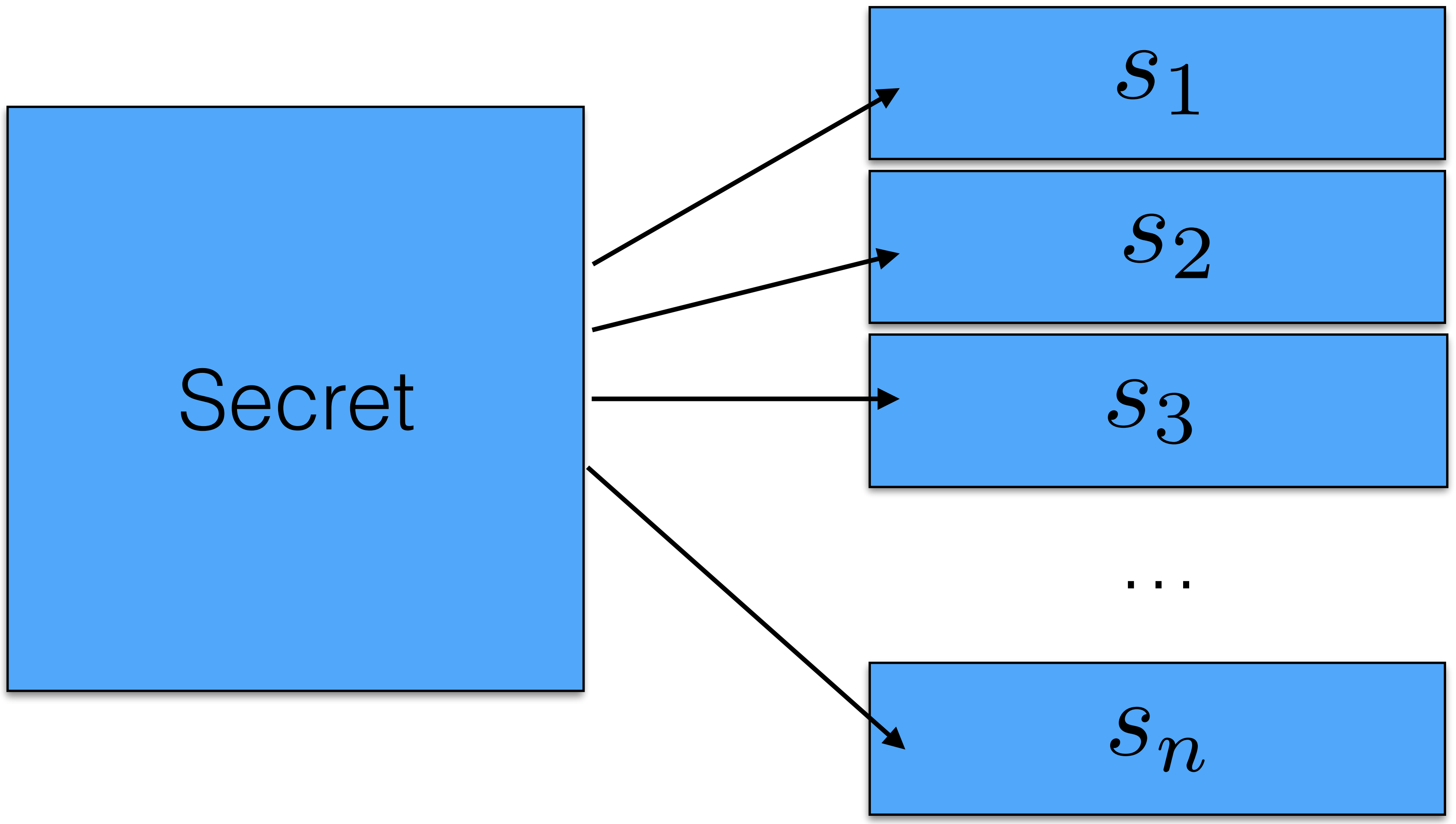
Robust Password-Protected Secret Sharing



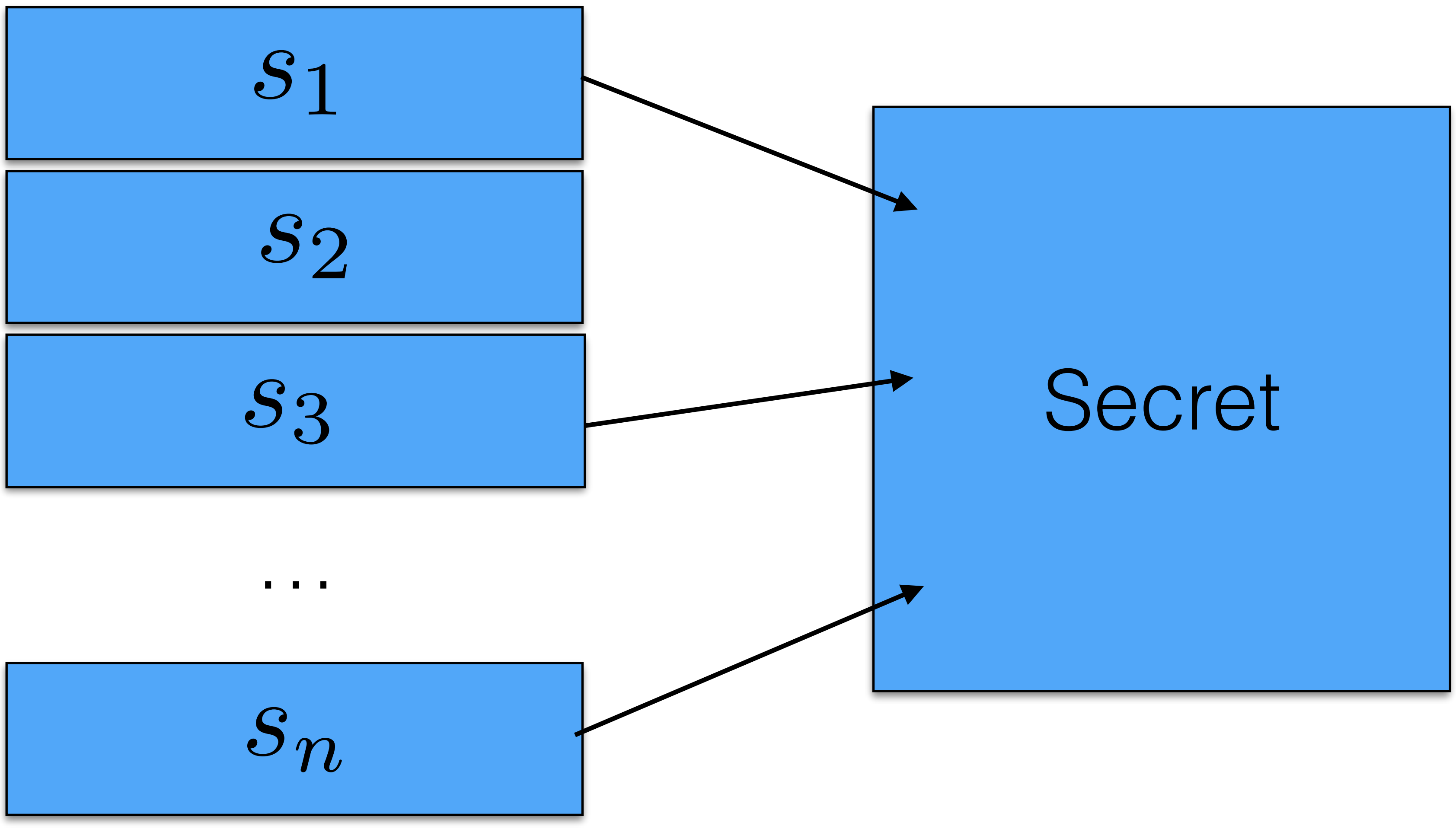
Robust Password-Protected Secret Sharing



PPSS: Secret Sharing Scheme



PPSS: Secret Sharing Scheme



PPSS: Robust Gap Secret Sharing Scheme

How do we implement robustness?

PPSS: Robust Gap Secret Sharing Scheme

Assume a set of valid shares from a Threshold SSS

s_1

s_2

s_3

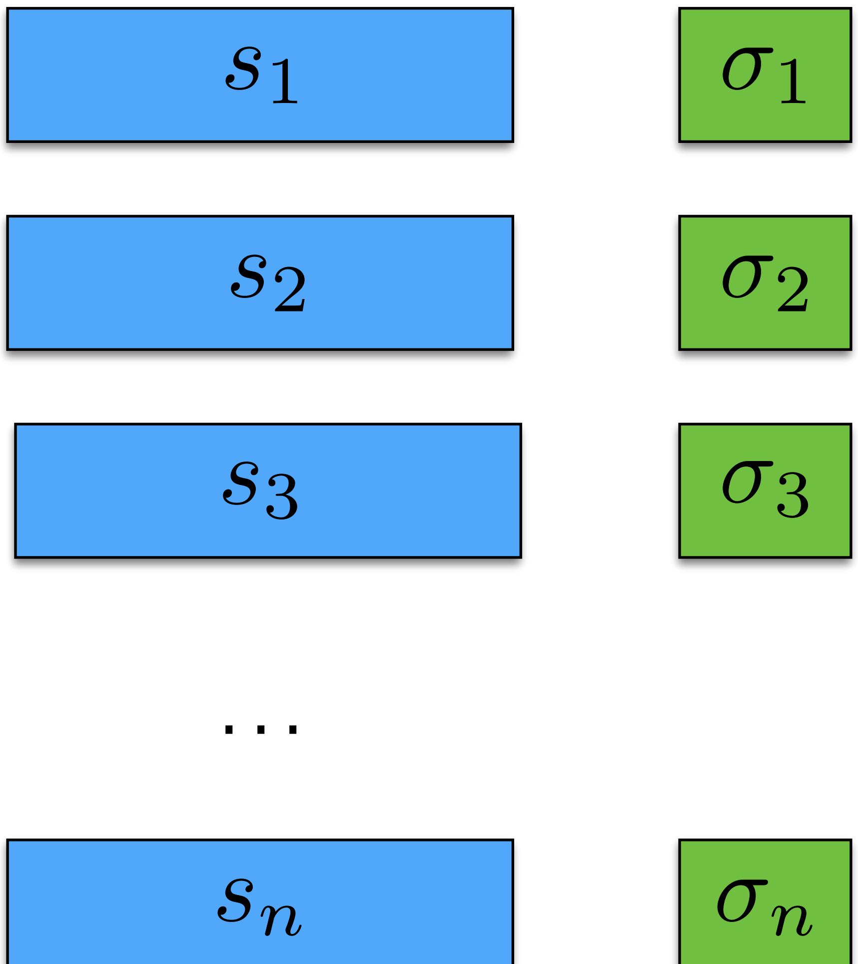
$\ldots$

s_n

$$(s_1, \ldots, s_n)$$

PPSS: Robust Gap Secret Sharing Scheme

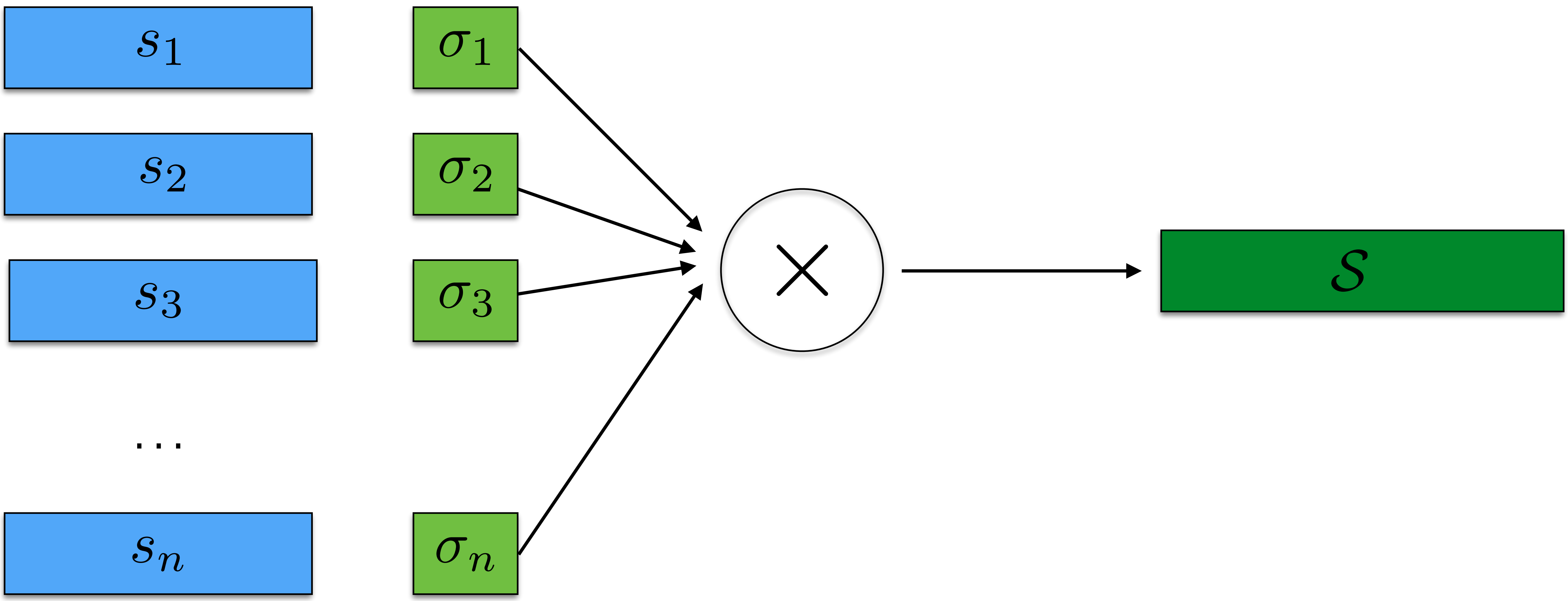
Fingerprint function: Hash function



$$(s_1, \ldots, s_n) \quad (\sigma_1, \ldots, \sigma_n)$$

PPSS: Robust Gap Secret Sharing Scheme

Generate a prime number N $\quad 2^{2k(n-t_r)+1} < N \leq 2^{2k(n-t_r)+2}$



$(s_1, \dots, s_n) \quad (\sigma_1, \dots, \sigma_n)$

$\mathcal{S} = \prod_{i=1}^{n} \sigma_i \mod N$

PPSS: Robust Gap Secret Sharing Scheme

Generate a prime number N $\quad 2^{2k(n-t_r)+1} < N \leq 2^{2k(n-t_r)+2}$

s_1

s_2

s_3

$\dots$

s_n

Output:

$\{s_k\}_n = \boxed{s_1}\ \boxed{s_2}\ \boxed{s_3} \dots \boxed{s_n}$

$\mathsf{SSInfo} = \{ \boxed{\mathcal{S}}, N \}$

$\mathcal{S}$

$(s_1, \dots, s_n) \quad (\sigma_1, \dots, \sigma_n)$

$\mathcal{S} = \prod_{i=1}^{n} \sigma_i \mod N$

PPSS: Robust Gap Secret Sharing Scheme

How can we decide which are the valid sets of shares to reconstruct?

PPSS: Robust Gap Secret Sharing Scheme

Given $\mathsf{SSInfo} = \{\ \mathcal{S}\ , N\ \}$

s_1

s_2

s_3

$\ldots$

s_n

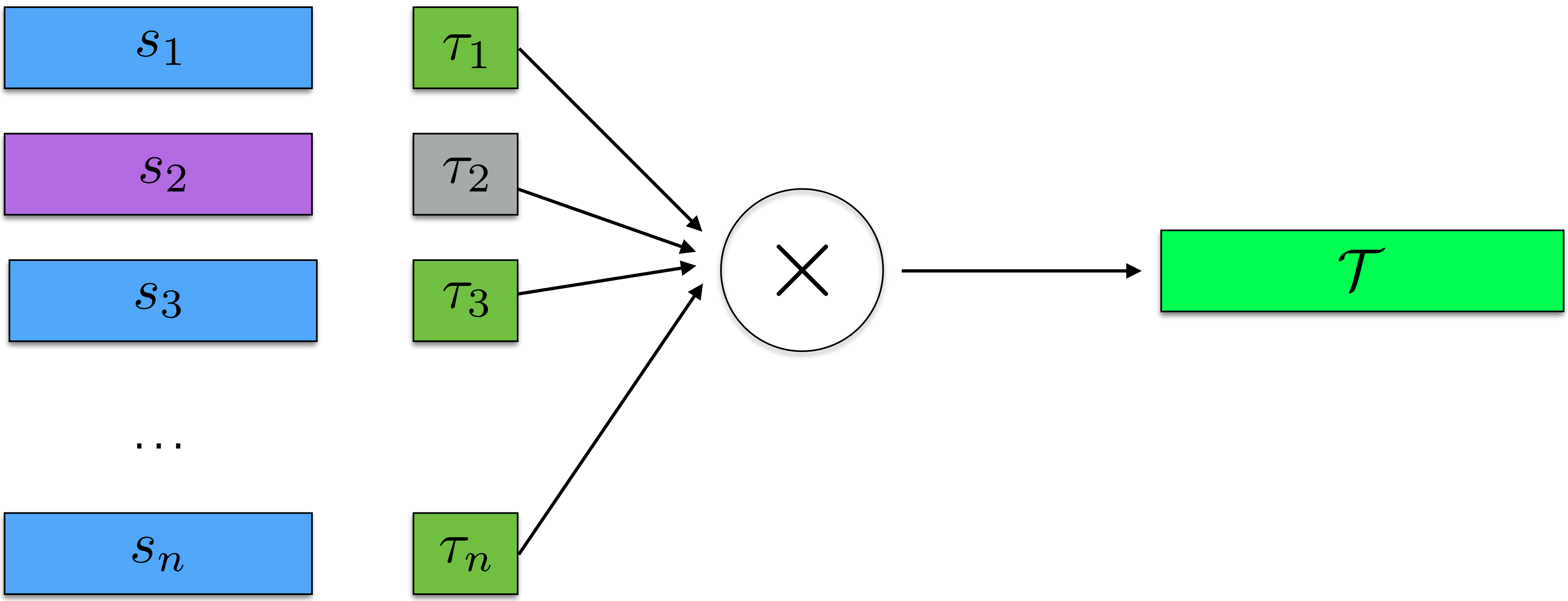
PPSS: Robust Gap Secret Sharing Scheme

Given $\mathsf{SSInfo} = \{ \boxed{\mathcal{S}} , N \}$

s_1	τ_1
s_2	τ_2
s_3	τ_3
$\ldots$	
s_n	τ_n

PPSS: Robust Gap Secret Sharing Scheme

Given $\mathsf{SSInfo} = \{ \ \mathcal{S} \ , N \ \}$



PPSS: Robust Gap Secret Sharing Scheme

Given $\mathsf{SSInfo} = \{\ \mathcal{S}\ , N\ \}$

$$\gamma = \frac{\mathcal{T}}{\mathcal{S}}$$

PPSS: Robust Gap Secret Sharing Scheme

Given $\mathsf{SSInfo} = \{ \boxed{\mathcal{S}}, N \}$

$$\gamma = \frac{\boxed{\mathcal{T}}}{\boxed{\mathcal{S}}} = \frac{\boxed{\tau_1}\ \boxed{\tau_2}\ \boxed{\tau_3}\ \ldots\ \boxed{\tau_n}}{\boxed{\sigma_1}\ \boxed{\sigma_2}\ \boxed{\sigma_3}\ \ldots\ \boxed{\sigma_n}}$$

PPSS: Robust Gap Secret Sharing Scheme

Given $\mathsf{SSInfo} = \{ \mathcal{S}, N \}$

$$\gamma = \frac{\mathcal{T}}{\mathcal{S}} = \frac{\tau_1 \quad \tau_2 \quad \tau_3 \quad \dots \quad \tau_n}{\sigma_1 \quad \sigma_2 \quad \sigma_3 \quad \dots \quad \sigma_n} = \frac{\mathcal{T}'}{\mathcal{S}'}$$

PPSS: Robust Gap Secret Sharing Scheme

Given $\mathsf{SSInfo} = \{ \mathcal{S}, N \}$

$$\gamma = \frac{\mathcal{T}}{\mathcal{S}} = \frac{\tau_1 \quad \tau_2 \quad \tau_3 \quad \ldots \quad \tau_n}{\sigma_1 \quad \sigma_2 \quad \sigma_3 \quad \ldots \quad \sigma_n} = \frac{\mathcal{T}'}{\mathcal{S}'}$$

$$|\gcd(\tau_1, \mathcal{T}')| \approx 1$$

PPSS: Robust Gap Secret Sharing Scheme

Given $\mathsf{SSInfo} = \{\ \boxed{\mathcal{S}}\ , N\ \}$

$$\gamma = \frac{\boxed{\mathcal{T}}}{\boxed{\mathcal{S}}} = \frac{\boxed{\tau_1}\ \boxed{\tau_2}\ \boxed{\tau_3}\ \ldots\ \boxed{\tau_n}}{\boxed{\sigma_1}\ \boxed{\sigma_2}\ \boxed{\sigma_3}\ \ldots\ \boxed{\sigma_n}} = \frac{\boxed{\mathcal{T}'}}{\boxed{\mathcal{S}'}}$$

$$|\gcd(\boxed{\tau_1}, \boxed{\mathcal{T}'})| \approx 1$$

Correct fingerprint!

PPSS: Robust Gap Secret Sharing Scheme

Given SSInfo $= \{ \boxed{\mathcal{S}} , N \}$

$$\gamma = \frac{\boxed{\mathcal{T}}}{\boxed{\mathcal{S}}} = \frac{\boxed{\tau_1}\ \boxed{\tau_2}\ \boxed{\tau_3}\ \ldots\ \boxed{\tau_n}}{\boxed{\sigma_1}\ \boxed{\sigma_2}\ \boxed{\sigma_3}\ \ldots\ \boxed{\sigma_n}} = \frac{\boxed{\mathcal{T}'}}{\boxed{\mathcal{S}'}}$$

$|\gcd(\boxed{\tau_1}, \boxed{\mathcal{T}'})| \approx 1$ Correct fingerprint!

$|\gcd(\boxed{\tau_2}, \boxed{\mathcal{T}'})| \approx k$ Incorrect fingerprint!

PPSS: Robust Gap Secret Sharing Scheme

Given $\mathsf{SSInfo} = \{ \boxed{\mathcal{S}}, N \}$

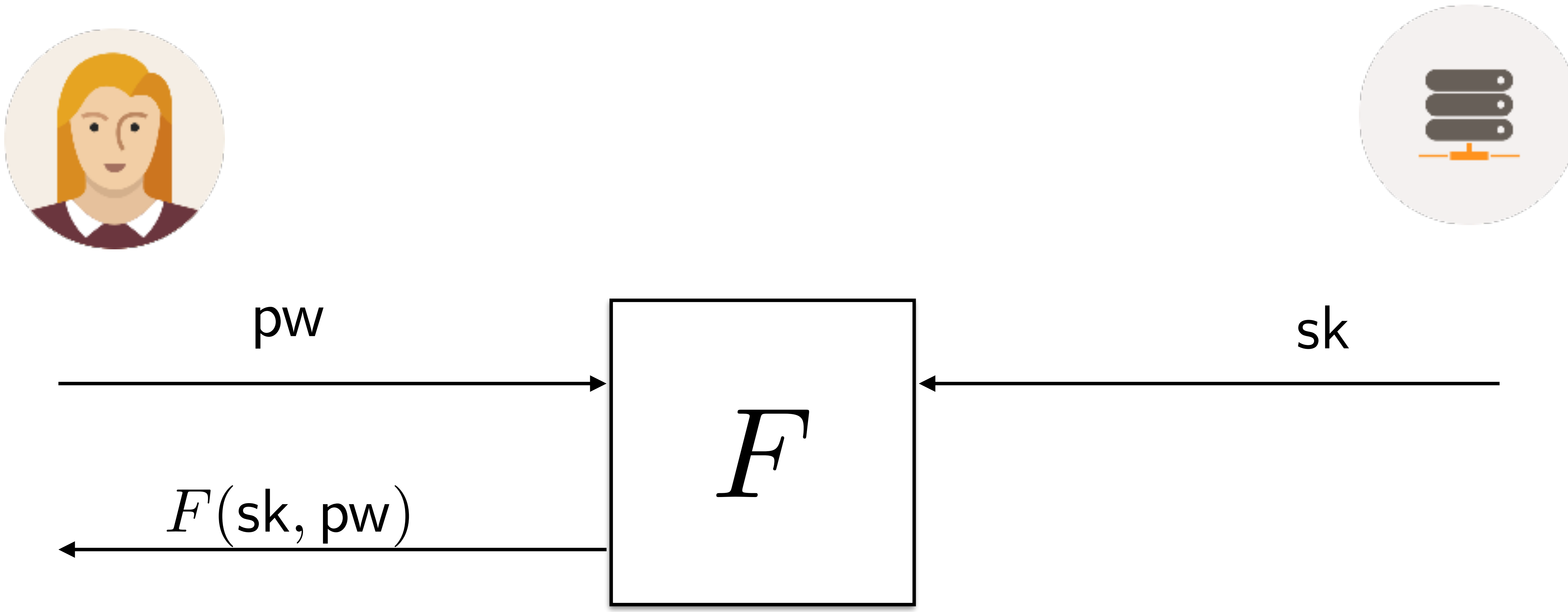
$$\gamma = \frac{\boxed{\mathcal{T}}}{\boxed{\mathcal{S}}} = \frac{\boxed{\tau_1}\ \boxed{\tau_2}\ \boxed{\tau_3}\ \ldots\ \boxed{\tau_n}}{\boxed{\sigma_1}\ \boxed{\sigma_2}\ \boxed{\sigma_3}\ \ldots\ \boxed{\sigma_n}} = \frac{\boxed{\mathcal{T}'}}{\boxed{\mathcal{S}'}}$$

$|\gcd(\boxed{\tau_1}, \boxed{\mathcal{T}'})| \approx 1$ Correct fingerprint!

$|\gcd(\boxed{\tau_2}, \boxed{\mathcal{T}'})| \approx k$ Incorrect fingerprint!

$|\gcd(\boxed{\tau_3}, \boxed{\mathcal{T}'})| \approx 1$ Correct fingerprint!

PPSS: Oblivious PRF



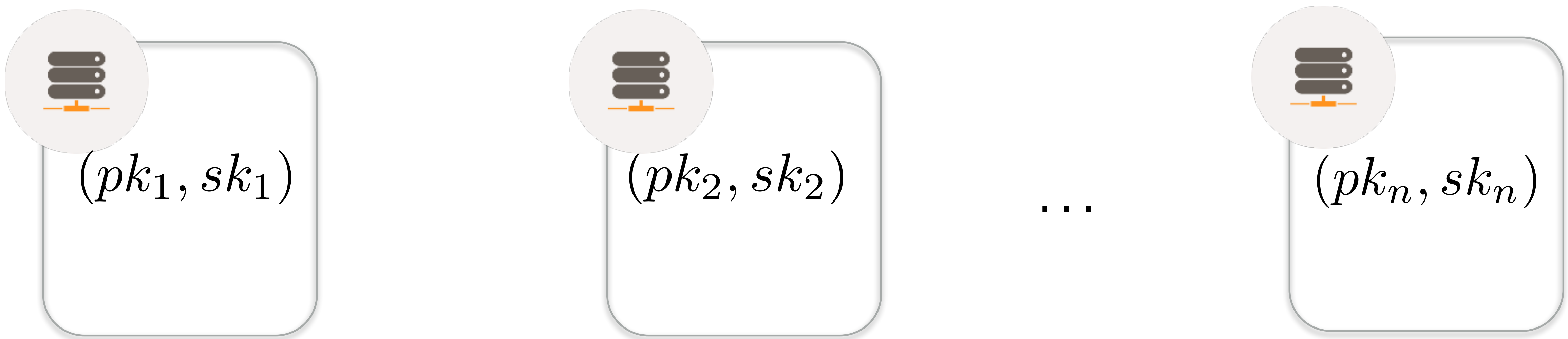
- The output is indistinguishable from random
- The server learns nothing

PPSS: Password-Protected Secret Sharing

Initialization phase

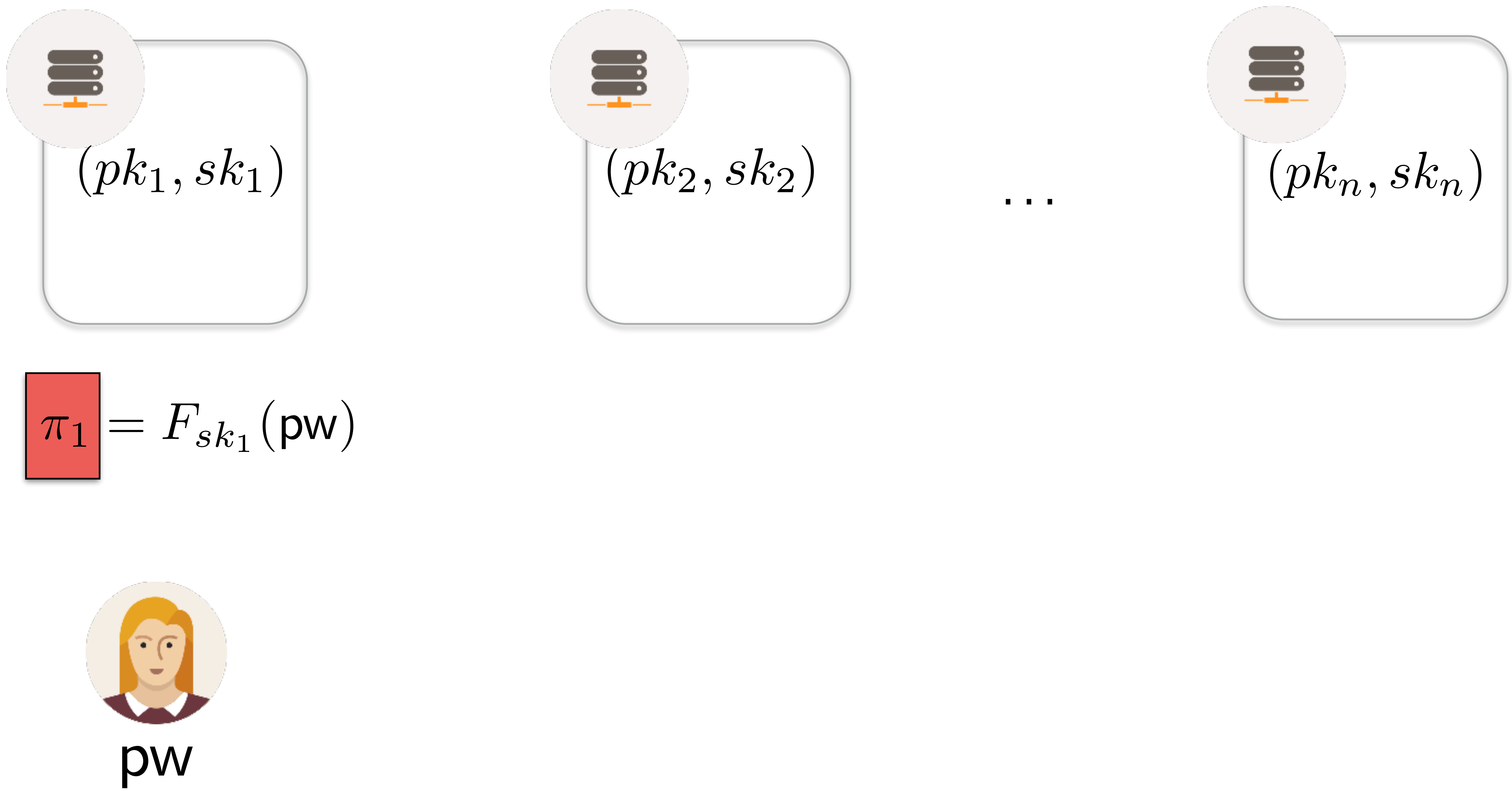
PPSS: Initialization

The user interacts with n servers to obliviously evaluate the PRF



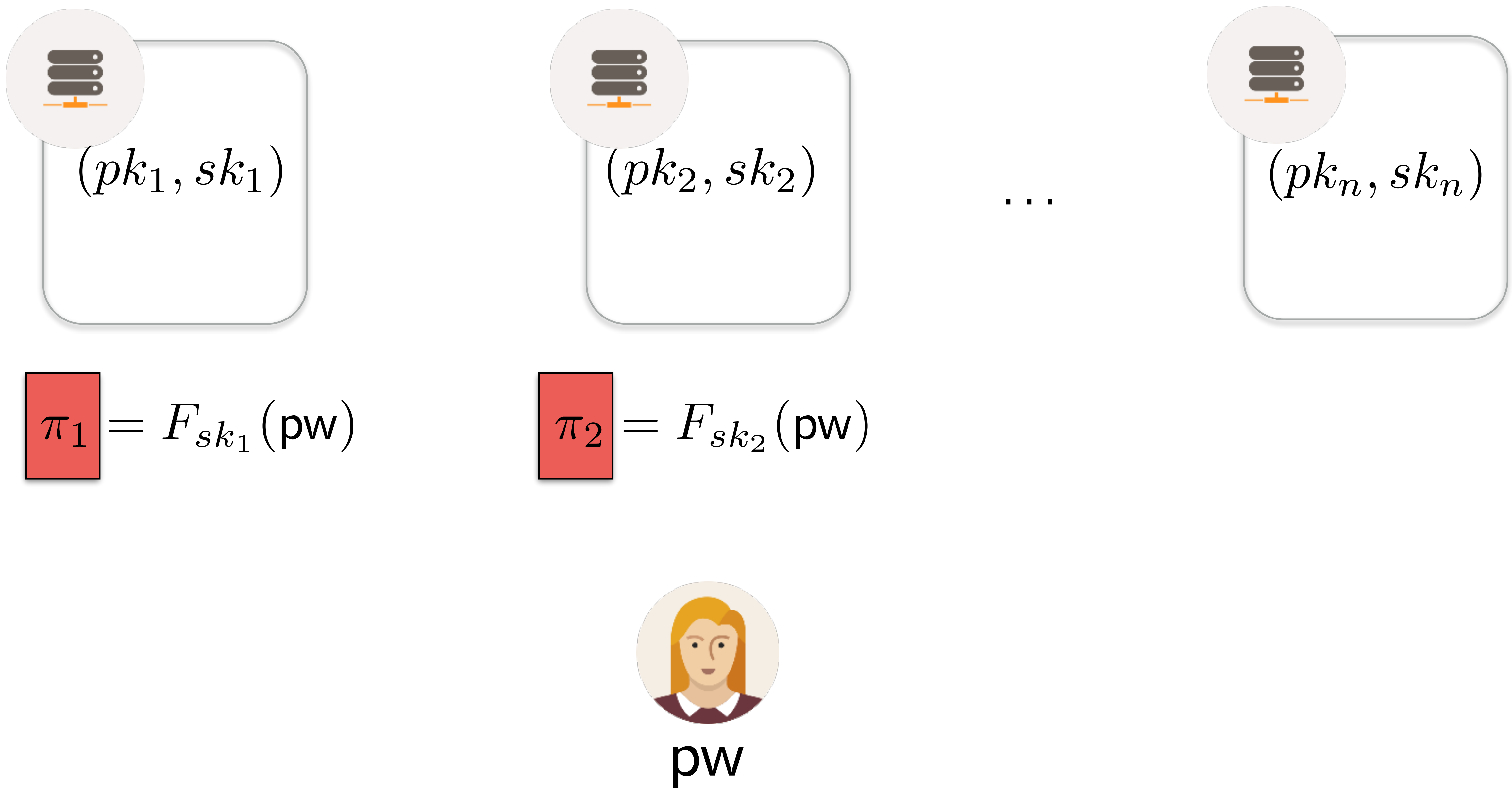
PPSS: Initialization

The user interacts with n servers to obliviously evaluate the PRF



PPSS: Initialization

The user interacts with n servers to obliviously evaluate the PRF



PPSS: Initialization

The user interacts with n servers to obliviously evaluate the PRF

(pk_1, sk_1) (pk_2, sk_2) $\dots$ (pk_n, sk_n)

$\pi_1 = F_{sk_1}(\mathsf{pw})$ $\pi_2 = F_{sk_2}(\mathsf{pw})$ $\pi_n = F_{sk_n}(\mathsf{pw})$

pw

PPSS: Initialization

Each share is encrypted using the each PRF evaluation

(pk_1, sk_1)

(pk_2, sk_2)

$\dots$

(pk_n, sk_n)

$\boxed{\pi_1} = F_{sk_1}(\mathsf{pw})$

$\boxed{\pi_2} = F_{sk_2}(\mathsf{pw})$

$\boxed{\pi_n} = F_{sk_n}(\mathsf{pw})$

$$R = K || r$$

$\mathsf{pw} \quad \{\pi_k\}_n \quad \{pk_k\}_n$

PPSS: Initialization

Each share is encrypted using the each PRF evaluation

(pk_1, sk_1)

(pk_2, sk_2)

$\ldots$

(pk_n, sk_n)

$\pi_1 = F_{sk_1}(\mathsf{pw})$ $\pi_2 = F_{sk_2}(\mathsf{pw})$ $\pi_n = F_{sk_n}(\mathsf{pw})$

$$R = K || r$$

$$(s_1, \ldots, s_n, \mathsf{SSInfo}) \leftarrow \mathsf{ShareGen}(R)$$

pw $\{\pi_k\}_n$ $\{pk_k\}_n$

PPSS: Initialization

Each share is encrypted using the each PRF evaluation

(pk_1, sk_1) (pk_2, sk_2) $\ldots$ (pk_n, sk_n)

$\pi_1 = F_{sk_1}(\mathsf{pw})$ $\pi_2 = F_{sk_2}(\mathsf{pw})$ $\pi_n = F_{sk_n}(\mathsf{pw})$

$$R = K || r$$

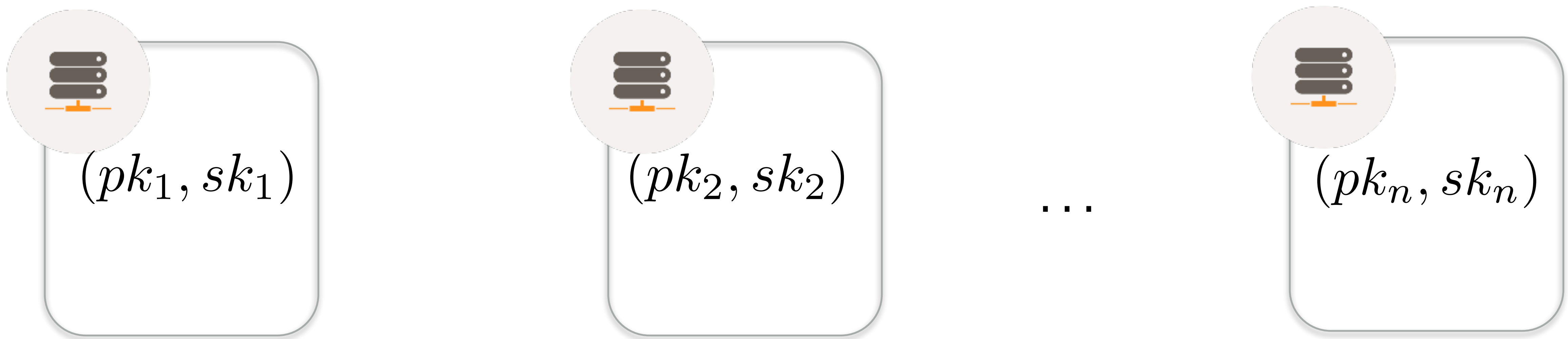
$$(s_1, \ldots, s_n, \mathsf{SSInfo}) \leftarrow \mathsf{ShareGen}(R)$$

$$\sigma_k = \pi_k \oplus s_k$$

pw $\{\pi_k\}_n$ $\{pk_k\}_n$

PPSS: Initialization

The user computes a commitment



$$C = \mathsf{Commit}(pw, H(\{pk_k\}_n, \{\sigma_k\}_n, \mathsf{SSInfo}, K); r)$$

pw $\{\pi_k\}_n$ $\{pk_k\}_n$ K r SSInfo $\{\sigma_k\}_n$

PPSS: Initialization

The user uploads the encrypted data

(pk_1, sk_1)

PInfo

(pk_2, sk_2)

PInfo

$\dots$

(pk_n, sk_n)

PInfo

$$\text{PInfo} = (\boxed{\{pk_k\}_n}, \boxed{\{\sigma_k\}_n}, \boxed{\text{SSInfo}}, C)$$

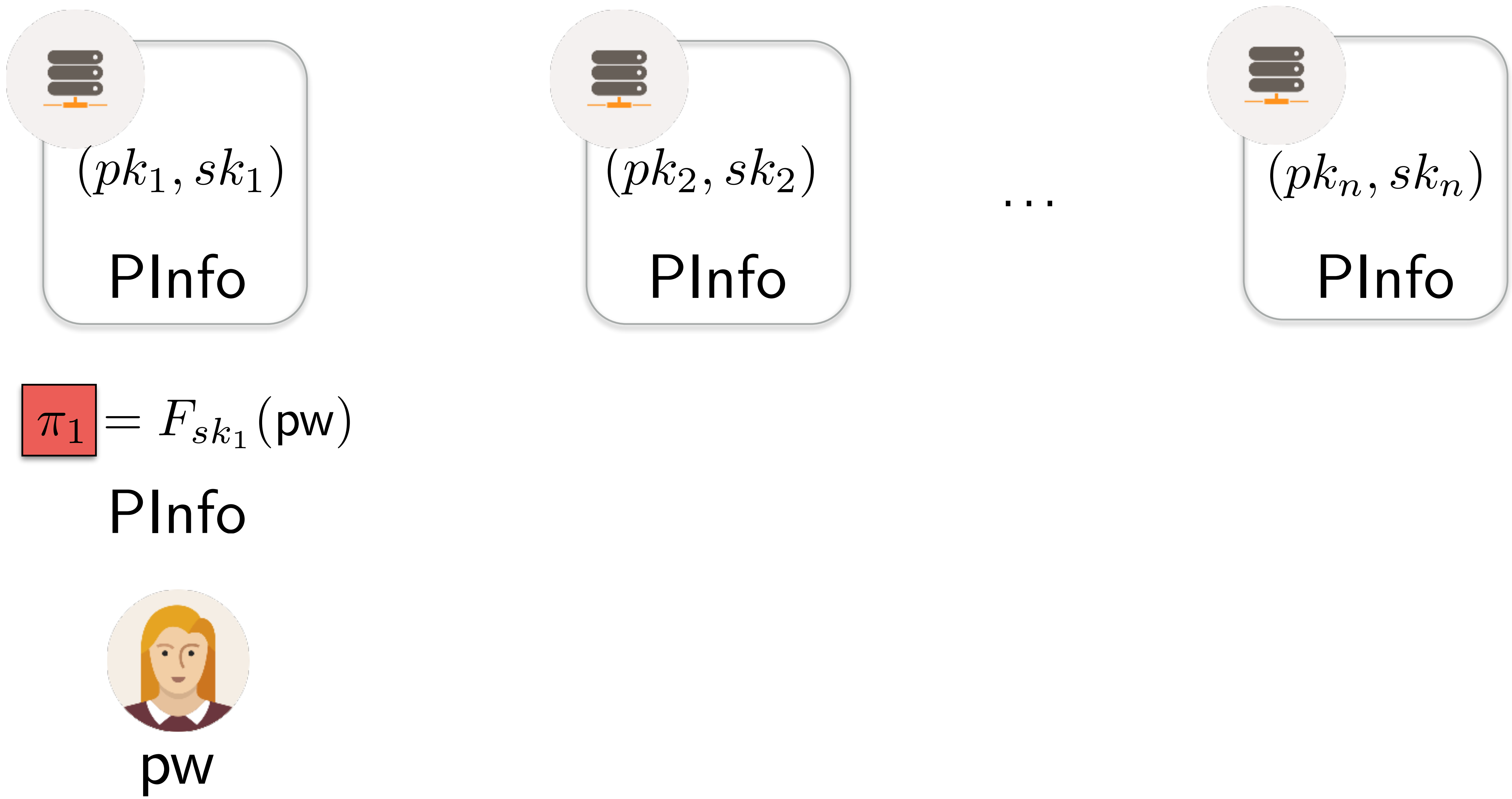
pw $\{\pi_k\}_n$ $\{pk_k\}_n$ K r SSInfo $\{\sigma_k\}_n$ C

PPSS: Password-Protected Secret Sharing

Reconstruction phase

PPSS: Reconstruction

The user interacts with the server



PPSS: Reconstruction

The user interacts with the server

(pk_1, sk_1)

PInfo

(pk_2, sk_2)

PInfo

$\ldots$

(pk_n, sk_n)

PInfo

$\boxed{\pi_1} = F_{sk_1}(\mathsf{pw})$

$\boxed{\pi_2} = F_{sk_2}(\mathsf{pw})$

PInfo

pw

PPSS: Reconstruction

The user interacts with the server

(pk_1, sk_1) PInfo

(pk_2, sk_2) PInfo

$\ldots$

(pk_n, sk_n) PInfo

$\boxed{\pi_1} = F_{sk_1}(\text{pw})$

$\boxed{\pi_2} = F_{sk_2}(\text{pw})$

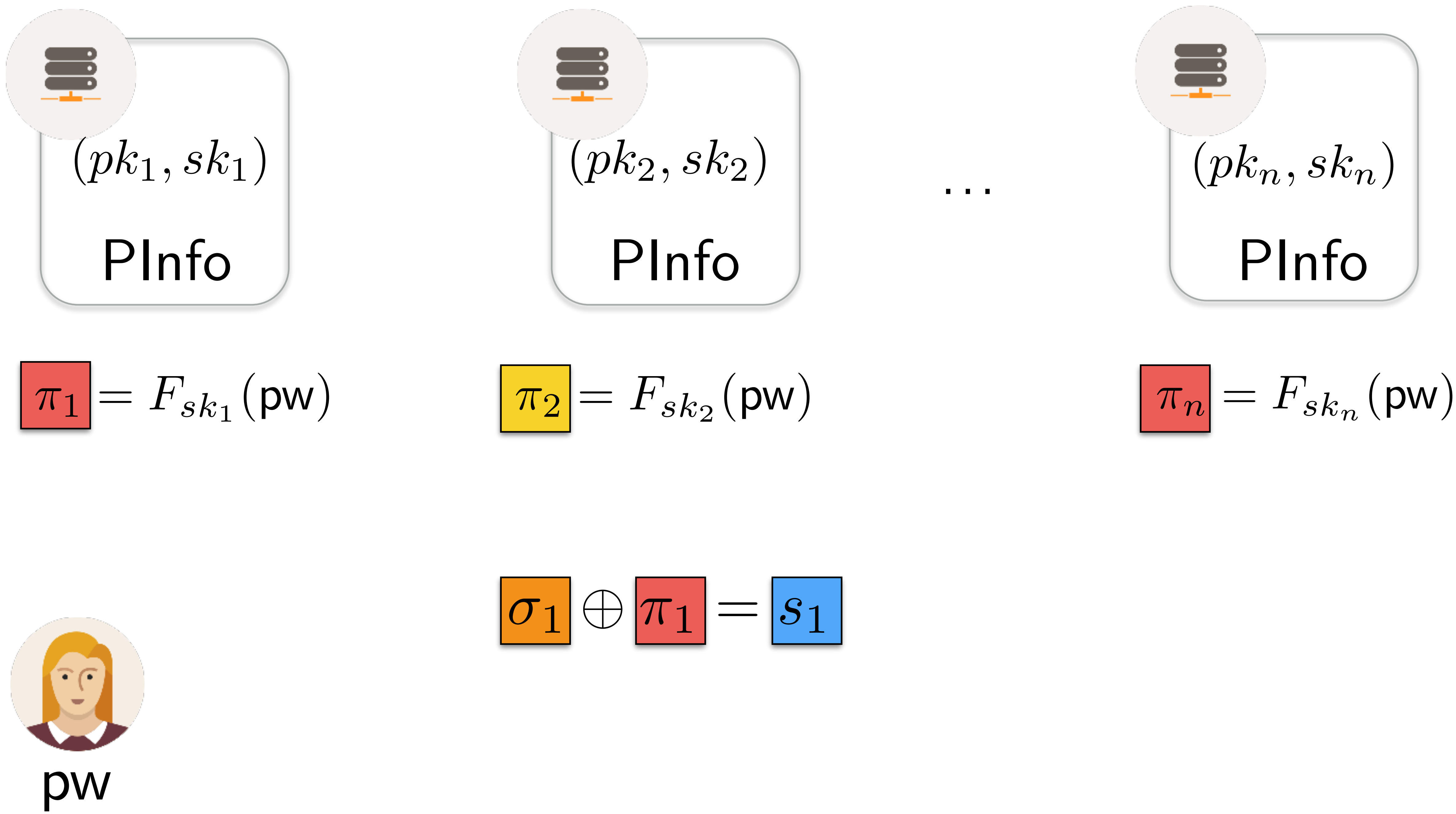
$\boxed{\pi_n} = F_{sk_n}(\text{pw})$

PInfo

pw

PPSS: Reconstruction

The user interacts with the server



PPSS: Reconstruction

The user interacts with the server

(pk_1, sk_1)

PInfo

(pk_2, sk_2)

PInfo

$\cdots$

(pk_n, sk_n)

PInfo

$\pi_1 = F_{sk_1}(\text{pw})$

$\pi_2 = F_{sk_2}(\text{pw})$

$\pi_n = F_{sk_n}(\text{pw})$

$\sigma_1 \oplus \pi_1 = s_1$

$\sigma_2 \oplus \pi_2 = s_2$

pw

PPSS: Reconstruction

The user interacts with the server

(pk_1, sk_1)

PInfo

(pk_2, sk_2)

PInfo

$\ldots$

(pk_n, sk_n)

PInfo

$\pi_1 = F_{sk_1}(\text{pw})$

$\pi_2 = F_{sk_2}(\text{pw})$

$\pi_n = F_{sk_n}(\text{pw})$

pw

$\sigma_1 \oplus \pi_1 = s_1$

$\sigma_2 \oplus \pi_2 = s_2$

$\sigma_3 \oplus \pi_3 = s_3$

PPSS: Reconstruction

The user interacts with the server

(pk_1, sk_1)

PInfo

(pk_2, sk_2)

PInfo

$\dots$

(pk_n, sk_n)

PInfo

$\pi_1 = F_{sk_1}(\text{pw})$

$\pi_2 = F_{sk_2}(\text{pw})$

$\pi_n = F_{sk_n}(\text{pw})$

pw

$$\sigma_1 \oplus \pi_1 = s_1$$

$$\sigma_2 \oplus \pi_2 = s_2$$

$$\sigma_3 \oplus \pi_3 = s_3$$

$$\sigma_n \oplus \pi_n = s_n$$

PPSS: Reconstruction

The user interacts with the server

(pk_1, sk_1)

PInfo

(pk_2, sk_2)

PInfo

$\dots$

(pk_n, sk_n)

PInfo

$\boxed{\pi_1} = F_{sk_1}(\mathsf{pw})$

$\boxed{\pi_2} = F_{sk_2}(\mathsf{pw})$

$\boxed{\pi_n} = F_{sk_n}(\mathsf{pw})$

pw

$$\text{Reconstruct}\left(\begin{array}{c} \boxed{s_1} \\ \boxed{s_2} \\ \boxed{s_3} \\ \boxed{s_n} \end{array} \right) \stackrel{?}{=} R = K||r$$

PPSS: Reconstruction

The user interacts with the server

(pk_1, sk_1)

PInfo

(pk_2, sk_2)

PInfo

$\ldots$

(pk_n, sk_n)

PInfo

$\boxed{\pi_1} = F_{sk_1}(\mathsf{pw})$

$\boxed{\pi_2} = F_{sk_2}(\mathsf{pw})$

$\boxed{\pi_n} = F_{sk_n}(\mathsf{pw})$

pw

$$\mathsf{Commit}(pw, H(\boxed{\{pk_k\}_n}, \boxed{\{\sigma_k\}_n}, \boxed{\mathsf{SSInfo}}, K); r) \stackrel{?}{=} C$$

PPSS: Proof [Sketch]

We build simulators for each PRFs

Adversary's probability is bounded by:

PPSS: Proof [Sketch]

We build simulators for each PRFs

Adversary's probability is bounded by:

Probability of guessing pw

$$\Pr[\mathsf{PWinC}] = \frac{q_u}{\#Dict}$$

PPSS: Proof [Sketch]

We build simulators for each PRFs

Adversary’s probability is bounded by:

Probability of breaking the OPRF

$$\Pr[\mathsf{PWinF}] = \varepsilon$$

PPSS: Comparison

- By using our robust threshold secret sharing we avoid the verifiability requirements for the OPRF.

- We reduce the communication to the half, because of the simplification of the OPRF.

- Our communication and computation complexities are asymptotically equivalent to [JKK14], in real life they are twice better.

Robust Password-Protected Secret Sharing

Michel Abdalla, Mario Cornejo,
Anca Niţulescu, David Pointcheval

École Normale Supérieure, CNRS and INRIA, Paris, France

PPSS: Experimental Results

Given SSInfo $= \{ \boxed{\mathcal{S}}, N \}$

$$\gamma = \frac{\boxed{\mathcal{T}}}{\boxed{\mathcal{S}}} = \frac{\boxed{\tau_1}\ \boxed{\tau_2}\ \boxed{\tau_3}\ \ldots\ \boxed{\tau_n}}{\boxed{\sigma_1}\ \boxed{\sigma_2}\ \boxed{\sigma_3}\ \ldots\ \boxed{\sigma_n}} = \frac{\boxed{\mathcal{T}'}}{\boxed{\mathcal{S}'}}$$

PPSS: Experimental Results

Given $\mathsf{SSInfo} = \{ \boxed{\mathcal{S}}, N \}$

$$\gamma = \frac{\boxed{\mathcal{T}}}{\boxed{\mathcal{S}}} = \frac{\boxed{\tau_1} \; \boxed{\tau_2} \; \boxed{\tau_3} \; \dots \; \boxed{\tau_n}}{\boxed{\sigma_1} \; \boxed{\sigma_2} \; \boxed{\sigma_3} \; \dots \; \boxed{\sigma_n}} = \frac{\boxed{\mathcal{T}''}}{\boxed{\mathcal{S}''}}$$

$$|\gcd(\boxed{\tau_1}, \boxed{\mathcal{T}''})| = 1$$

Correct fingerprint!

PPSS: Experimental Results

Given SSInfo $= \{ \boxed{\mathcal{S}}, N \}$

$$\gamma = \frac{\boxed{\mathcal{T}}}{\boxed{\mathcal{S}}} = \frac{\boxed{\tau_1}\ \boxed{\tau_2}\ \boxed{\tau_3}\ \ldots\ \boxed{\tau_n}}{\boxed{\sigma_1}\ \boxed{\sigma_2}\ \boxed{\sigma_3}\ \ldots\ \boxed{\sigma_n}} = \frac{\boxed{\mathcal{T}''}}{\boxed{\mathcal{S}''}}$$

$|\,\mathrm{gcd}(\boxed{\tau_1}, \boxed{\mathcal{T}''})| = 1$ Correct fingerprint!

$|\,\mathrm{gcd}(\boxed{\tau_1}, \boxed{\mathcal{T}''})| = 2$ Correct fingerprint!

$|\,\mathrm{gcd}(\boxed{\tau_1}, \boxed{\mathcal{T}''})| = 3$ Correct fingerprint!

PPSS: Experimental Results

$$|\gcd(\boxed{\tau_2}, \boxed{\mathcal{T}''})| = k$$ Incorrect fingerprint!

$$|\gcd(\boxed{\tau_2}, \boxed{\mathcal{T}''})| = k - 1$$ Incorrect fingerprint!

$$|\gcd(\boxed{\tau_2}, \boxed{\mathcal{T}''})| = k - 2$$ Incorrect fingerprint!

(a) $\gcd(\mathcal{T}'', \tau_i)$-bitlength for valid τ_i. (b) $\gcd(\mathcal{T}'', \tau_i)$-bitlength for invalid τ_i.

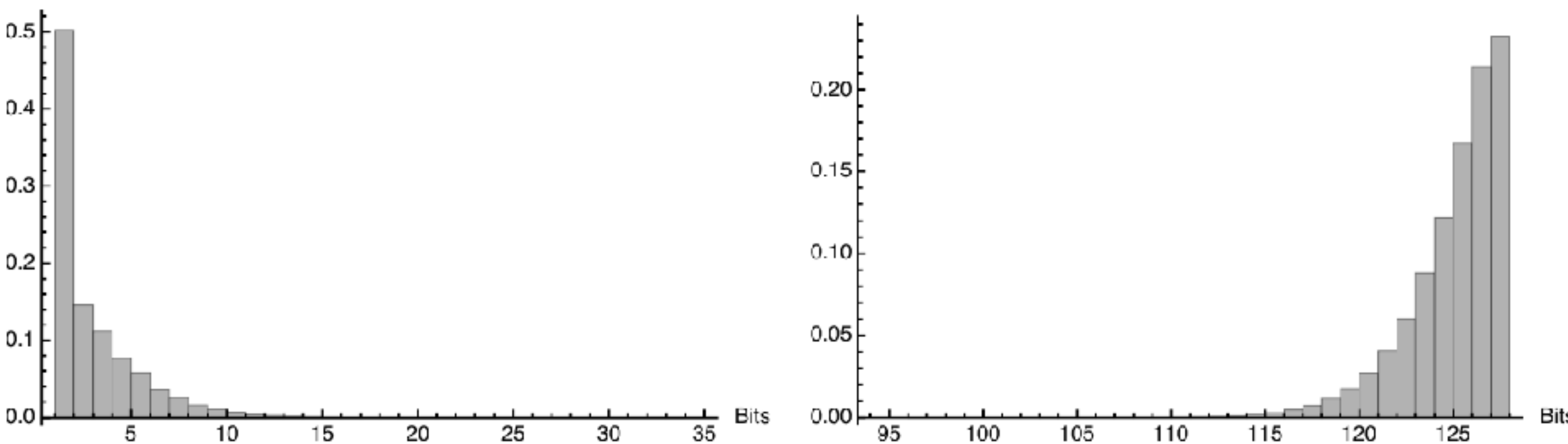


Fig. 1: Length in bits of $\gcd(\mathcal{T}'', \tau_i)$ for a fingerprint of size 128-bits and 32 shares

PPSS: CDH-based PRF (One-More Gap DH)

pw

$\mathsf{pk} = g^{\mathsf{sk}}$

sk

$\alpha \leftarrow \mathbb{Z}^*$

$A \leftarrow H_1(\mathsf{pw})^{\alpha}$

$\xrightarrow{\quad A \quad}$

$B \leftarrow A^{sk}$

$\xleftarrow{\quad B \quad}$

$C \leftarrow B^{1/\alpha} = H_1(\mathsf{pw})^{\mathsf{sk}}$

$F_{sk}(pw) = H_2(\mathsf{pw}, \mathsf{pk}, C)$

PPSS: DDH-based PRF

pk, $\{c_i = \mathsf{Enc}_{\mathsf{pk}}(a_i)\}$

$x = (x_1, x_2, \ldots, x_\ell) \in \{0,1\}^\ell$

$\alpha \leftarrow \mathbb{G}_s$

$C \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\alpha \times a_0 \prod {a_i}^{x_i})$

sk $\in \mathbb{Z}_s$

$$\xrightarrow{\quad C \quad}$$

$$\longleftrightarrow$$

$$\xleftrightarrow{\mathsf{Proof}(\alpha, x_i)}$$

$$\xleftarrow{\quad G \quad}$$

$D \leftarrow \mathsf{Dec}_{\mathsf{sk}}(C)$

$G \leftarrow g^D$

$R \leftarrow G^{1/\alpha}$