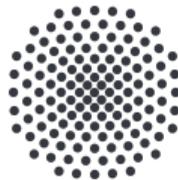


Universally Composable Password-Hardened Encryption

Behzad Abdolmaleki, Ruben Baecker, Paul Gerhart, Mike Graf, Mojtaba Khalili,
Daniel Rausch, and Dominique Schröder



University of
Sheffield



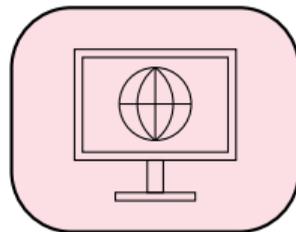
Why Threshold PHE?

Let's add UC security...

...and find a (round-optimal) instantiation

A Single Point of Failure

$$H(\text{"1234"}, 2181) = b481$$



un	s	h	data
alice	2181	b481	"secret"

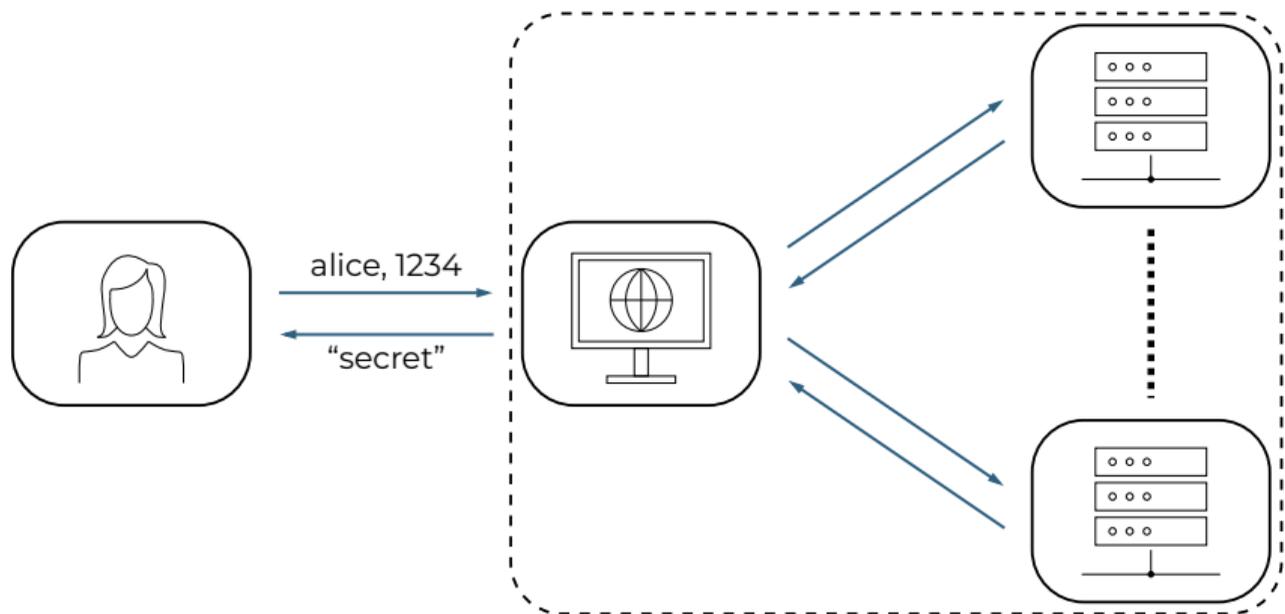
PHE: Distributing Trust

ECSJR'15 (USENIX), SFSB'16 (CCS), LESC'17 (USENIX), LERCMS'18 (USENIX), BELSSZ'20 (CCS)



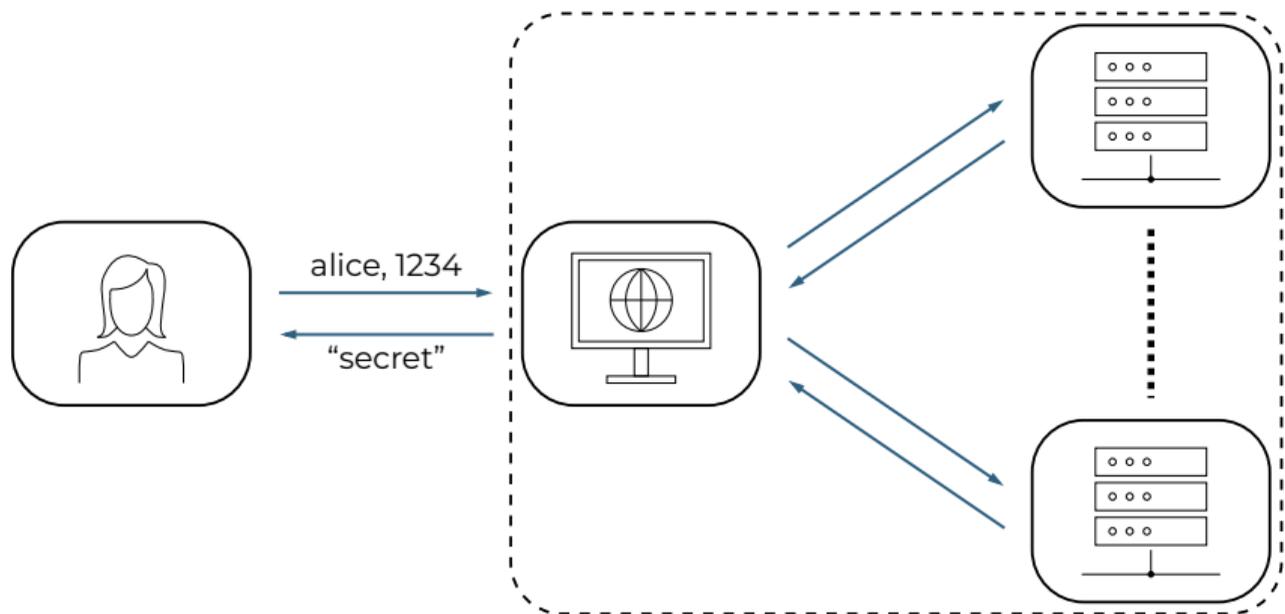
TPHE: Distributing Trust

ECSJR'15 (USENIX), SFSB'16 (CCS), LESC'17 (USENIX), LERCMS'18 (USENIX), BELSSZ'20 (CCS)



TPHE: Distributing Trust **even more**

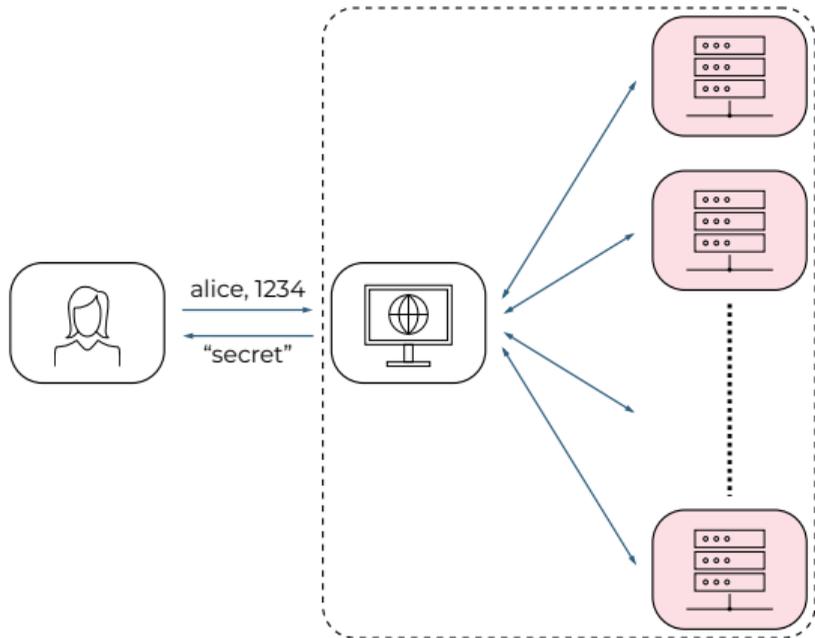
ECSJR'15 (USENIX), SFSB'16 (CCS), LESC'17 (USENIX), LERCMS'18 (USENIX), BELSSZ'20 (CCS)



TPHE: Security Against Corruption

Security against corruption of

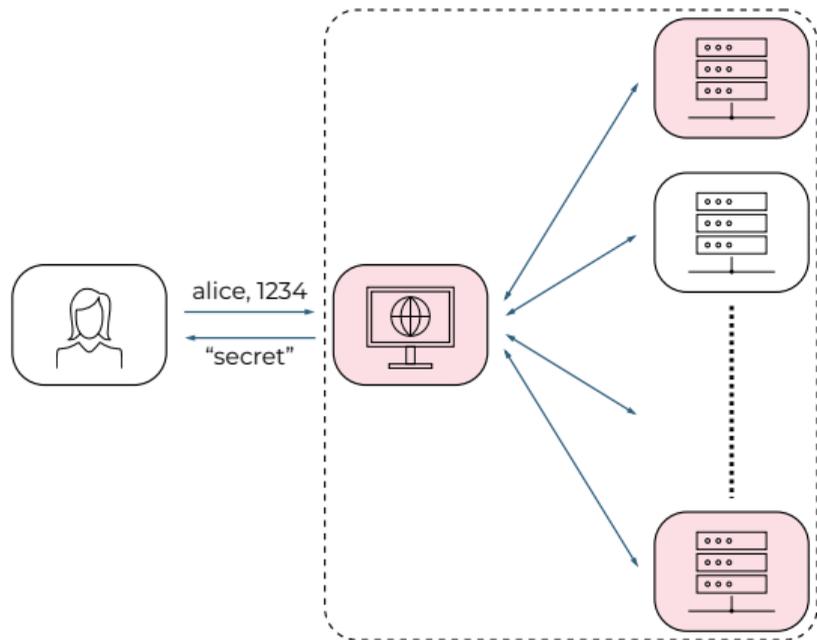
- **All** n ratelimiters
- The server **and** $t - 1$ ratelimiters



TPHE: Security Against Corruption

Security against corruption of

- **All** n ratelimiters
- The server **and** $t - 1$ ratelimiters



Why Threshold PHE?

Let's add UC security...

...and find a (round-optimal) instantiation

Game-based vs Universally Composable

- PH(E) has a history of insufficient security models:
LESC'17 discovered a gap in SFSB'16, and **BGS'25** in LERCMS'18
- PH(E) is used in the context of complex eco-systems
→ standalone security is insufficient
- Human behaviour is not modeled in game-based definitions
(typos, password re-usage, using personal information)

Game-based vs Universally Composable

- PH(E) has a history of insufficient security models:
LESC'17 discovered a gap in SFSB'16, and **BGS'25** in LERCMS'18
- PH(E) is used in the context of complex eco-systems
→ standalone security is insufficient
- Human behaviour is not modeled in game-based definitions
(typos, password re-usage, using personal information)

Game-based vs Universally Composable

- PH(E) has a history of insufficient security models:
LESC'17 discovered a gap in SFSB'16, and BGS'25 in LERCMS'18
- PH(E) is used in the context of complex eco-systems
→ standalone security is insufficient
- Human behaviour is not modeled in game-based definitions
(typos, password re-usage, using personal information)

Game-based vs Universally Composable

- PH(E) has a history of insufficient security models:
LESC'17 discovered a gap in SFSB'16, and BGS'25 in LERCMS'18
 - PH(E) is used in the context of complex eco-systems
→ standalone security is insufficient
 - Human behaviour is not modeled in game-based definitions
(typos, password re-usage, using personal information)
- A security definition in the UC Framework avoids all those problems

Internal State of $\mathcal{F}_{\text{TPHE}}$

- $\text{storage}(id)$
Maps id to a password, message pair (pw, m)
- $\text{retrieveRate}(\mathcal{R}_i, id)$
Tracks the remaining quota of \mathcal{R}_i for ciphertext stored at position id

Internal State of $\mathcal{F}_{\text{TPHE}}$

- $\text{storage}(id)$
Maps id to a password, message pair (pw, m)
- $\text{retrieveRate}(\mathcal{R}_i, id)$
Tracks the remaining quota of \mathcal{R}_i for ciphertext stored at position id

Functional Interfaces of $\mathcal{F}_{\text{TPHE}}$

- $(\text{Store}, id, pw, m)$ from \mathcal{S}
 - Store (pw, m) in $\text{storage}(id)$
 - Leak $(\text{Store}, id, |m|)$ to \mathcal{A}
- $(\text{Retrieve}, id, pw')$ from \mathcal{S}
 - Store request with unique $rqid$
 - Leak $(\text{Retrieve}, id, rqid)$ to \mathcal{A}
- $(\text{HelpRetrieve}, id)$ from \mathcal{R}_i
 - Increment $\text{retrieveRate}(\mathcal{R}_i, id)$
 - Notify \mathcal{A}

Functional Interfaces of $\mathcal{F}_{\text{TPHE}}$

- (Store, id, pw, m) from \mathcal{S}
 - Store (pw, m) in $\text{storage}(id)$
 - Leak (Store, $id, |m|$) to \mathcal{A}
- (Retrieve, id, pw') from \mathcal{S}
 - Store request with unique $rqid$
 - Leak (Retrieve, $id, rqid$) to \mathcal{A}
- (HelpRetrieve, id) from \mathcal{R}_i
 - Increment $\text{retrieveRate}(\mathcal{R}_i, id)$
 - Notify \mathcal{A}

Functional Interfaces of $\mathcal{F}_{\text{TPHE}}$

- $(\text{Store}, id, pw, m)$ from \mathcal{S}
 - Store (pw, m) in $\text{storage}(id)$
 - Leak $(\text{Store}, id, |m|)$ to \mathcal{A}
- $(\text{Retrieve}, id, pw')$ from \mathcal{S}
 - Store request with unique $rqid$
 - Leak $(\text{Retrieve}, id, rqid)$ to \mathcal{A}
- $(\text{HelpRetrieve}, id)$ from \mathcal{R}_i
 - Increment $\text{retrieveRate}(\mathcal{R}_i, id)$
 - Notify \mathcal{A}

Adversarial Interfaces of $\mathcal{F}_{\text{TPHE}}$

- (**FinishRetrieve**, $rqid$) from \mathcal{A}
 - Retrieve (**Retrieve**, id , pw') from queue
 - Let \mathcal{A} choose $RLset$ and ensure $\text{retrieveRate}(\mathcal{R}_i, id) \geq 1 \ \forall \mathcal{R}_i \in RLset$ and $|RLset| + n_c \geq t$
 - Decrement $\text{retrieveRate}(\mathcal{R}_i, id) \ \forall \mathcal{R}_i \in RLset$
 - If $pw' == pw$, return (**FinishRetrieve**, m) to \mathcal{S} ; otherwise, return (**FinishRetrieve**, \perp)
- (**ChangeCorruption**, P , $corrupt$) from \mathcal{A}
 - Set corruption status of P to $corrupt$
 - If $P = \mathcal{R}_i$ and $corrupt = \perp$, reset $\text{retrieveRate}(\mathcal{R}_i, id) \leftarrow 0 \ \forall id \in \mathbb{N}$

Adversarial Interfaces of $\mathcal{F}_{\text{TPHE}}$

- (**FinishRetrieve**, $rqid$) from \mathcal{A}
 - Retrieve (**Retrieve**, id , pw') from queue
 - Let \mathcal{A} choose $RLset$ and ensure $\text{retrieveRate}(\mathcal{R}_i, id) \geq 1 \ \forall \mathcal{R}_i \in RLset$ and $|RLset| + n_c \geq t$
 - Decrement $\text{retrieveRate}(\mathcal{R}_i, id) \ \forall \mathcal{R}_i \in RLset$
 - If $pw' == pw$, return (**FinishRetrieve**, m) to \mathcal{S} ; otherwise, return (**FinishRetrieve**, \perp)
- (**ChangeCorruption**, P , $corrupt$) from \mathcal{A}
 - Set corruption status of P to $corrupt$
 - If $P = \mathcal{R}_i$ and $corrupt = \perp$, reset $\text{retrieveRate}(\mathcal{R}_i, id) \leftarrow 0 \ \forall id \in \mathbb{N}$

Adversarial Interfaces of $\mathcal{F}_{\text{TPHE}}$

- $(\text{PwGuessStart}, id)$ from \mathcal{A}
 - Ensure $\text{storage}(id) \neq \perp$
 - Ensure \mathcal{S} is corrupt and perform same ratelimiting checks as for `FinishRetrieve`, including decrementing `retrieveRate`
 - Store new password guess token for the ciphertext stored under id
- $(\text{PwGuessFinish}, id, pw_{\mathcal{A}})$ from \mathcal{A}
 - Check that a password guess token for the ciphertext stored under id exists and delete it
 - Let (pw, m) be the entry of $\text{storage}(id)$
 - If $pw == pw_{\mathcal{A}}$, return $(\text{PwGuessFinish}, m)$ to \mathcal{A} ; otherwise, return $(\text{PwGuessFinish}, \perp)$

Adversarial Interfaces of $\mathcal{F}_{\text{TPHE}}$

- $(\text{PwGuessStart}, id)$ from \mathcal{A}
 - Ensure $\text{storage}(id) \neq \perp$
 - Ensure \mathcal{S} is corrupt and perform same ratelimiting checks as for FinishRetrieve , including decrementing retrieveRate
 - Store new password guess token for the ciphertext stored under id
- $(\text{PwGuessFinish}, id, pw_{\mathcal{A}})$ from \mathcal{A}
 - Check that a password guess token for the ciphertext stored under id exists and delete it
 - Let (pw, m) be the entry of $\text{storage}(id)$
 - If $pw == pw_{\mathcal{A}}$, return $(\text{PwGuessFinish}, m)$ to \mathcal{A} ; otherwise, return $(\text{PwGuessFinish}, \perp)$

The Need for PwGuessStart

Assume 2-out-of-3 TPHE and the following calls to $\mathcal{F}_{\text{TPHE}}$

1. (`Retrieve`, id , pw) from \mathcal{S}
2. (`HelpRetrieve`, id) from \mathcal{R}_1
3. (`HelpRetrieve`, id) from \mathcal{R}_3
4. (`ChangeCorruption`, \mathcal{R}_i , \perp) from $\mathcal{A} \quad \forall i \in [3]$
→ resets all `retrieveRate` to 0
5. (`FinishRetrieve`, $rqid$) from \mathcal{A}
fails **unintendetly** because of insufficient decryption quota

The Need for PwGuessStart

Assume 2-out-of-3 TPHE and the following calls to $\mathcal{F}_{\text{TPHE}}$

1. (Retrieve, id, pw) from \mathcal{S}
2. (HelpRetrieve, id) from \mathcal{R}_1
3. (HelpRetrieve, id) from \mathcal{R}_3
4. (ChangeCorruption, \mathcal{R}_i, \perp) from $\mathcal{A} \quad \forall i \in [3]$
→ resets all retrieveRate to 0
5. (FinishRetrieve, $rqid$) from \mathcal{A}
fails **unintendedly** because of insufficient decryption quota

The Need for PwGuessStart

Assume 2-out-of-3 TPHE and the following calls to $\mathcal{F}_{\text{TPHE}}$

1. (Retrieve, id, pw) from \mathcal{S}
2. (HelpRetrieve, id) from \mathcal{R}_1
3. (HelpRetrieve, id) from \mathcal{R}_3
4. (ChangeCorruption, \mathcal{R}_i, \perp) from $\mathcal{A} \quad \forall i \in [3]$
→ resets all retrieveRate to 0
5. (FinishRetrieve, $rqid$) from \mathcal{A}
fails **unintendedly** because of insufficient decryption quota

The Need for PwGuessStart

Assume 2-out-of-3 TPHE and the following calls to $\mathcal{F}_{\text{TPHE}}$

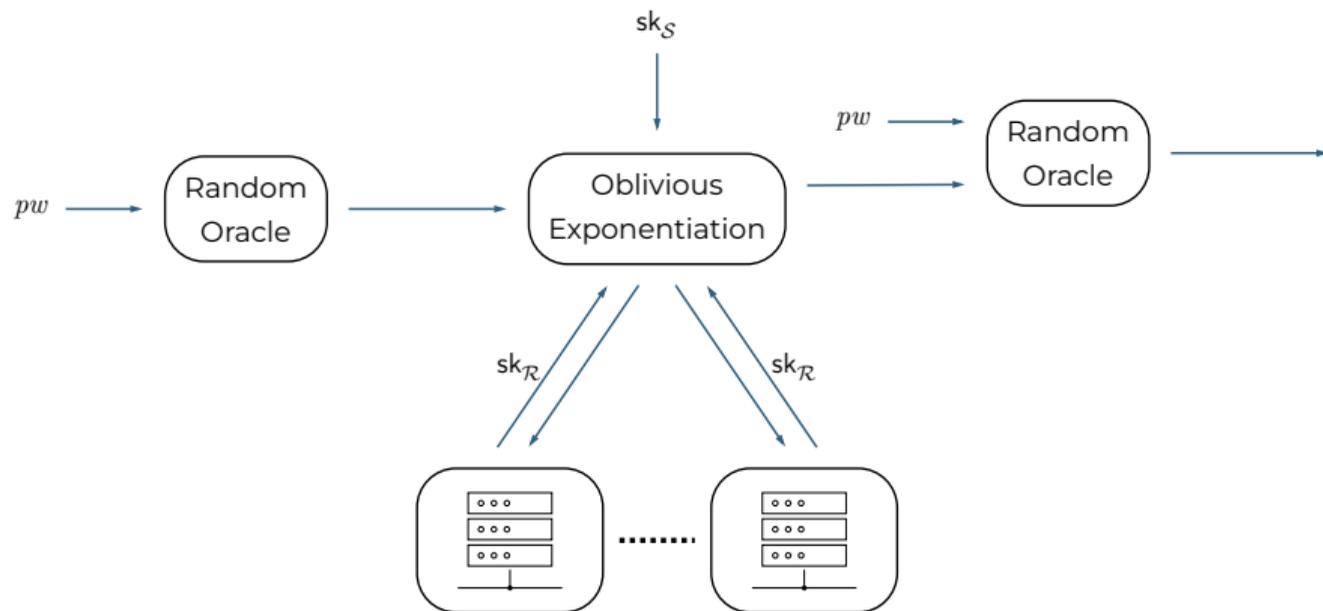
1. (Retrieve, id, pw) from \mathcal{S}
2. (HelpRetrieve, id) from \mathcal{R}_1
3. (HelpRetrieve, id) from \mathcal{R}_3
4. (ChangeCorruption, \mathcal{R}_i, \perp) from $\mathcal{A} \quad \forall i \in [3]$
→ resets all retrieveRate to 0
5. (FinishRetrieve, $rqid$) from \mathcal{A}
fails **unintendently** because of insufficient decryption quota

Why Threshold PHE?

Let's add UC security...

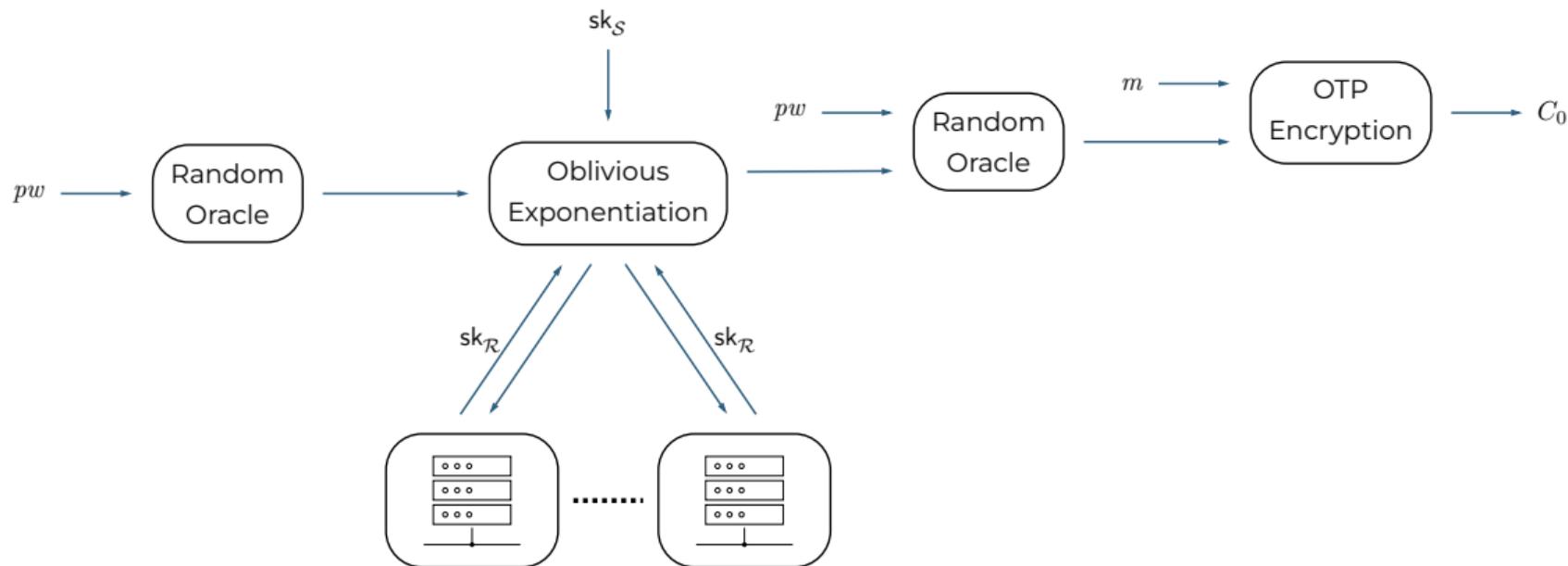
...and find a (round-optimal) instantiation

UCPY: A **Composable** TPHE Protocol



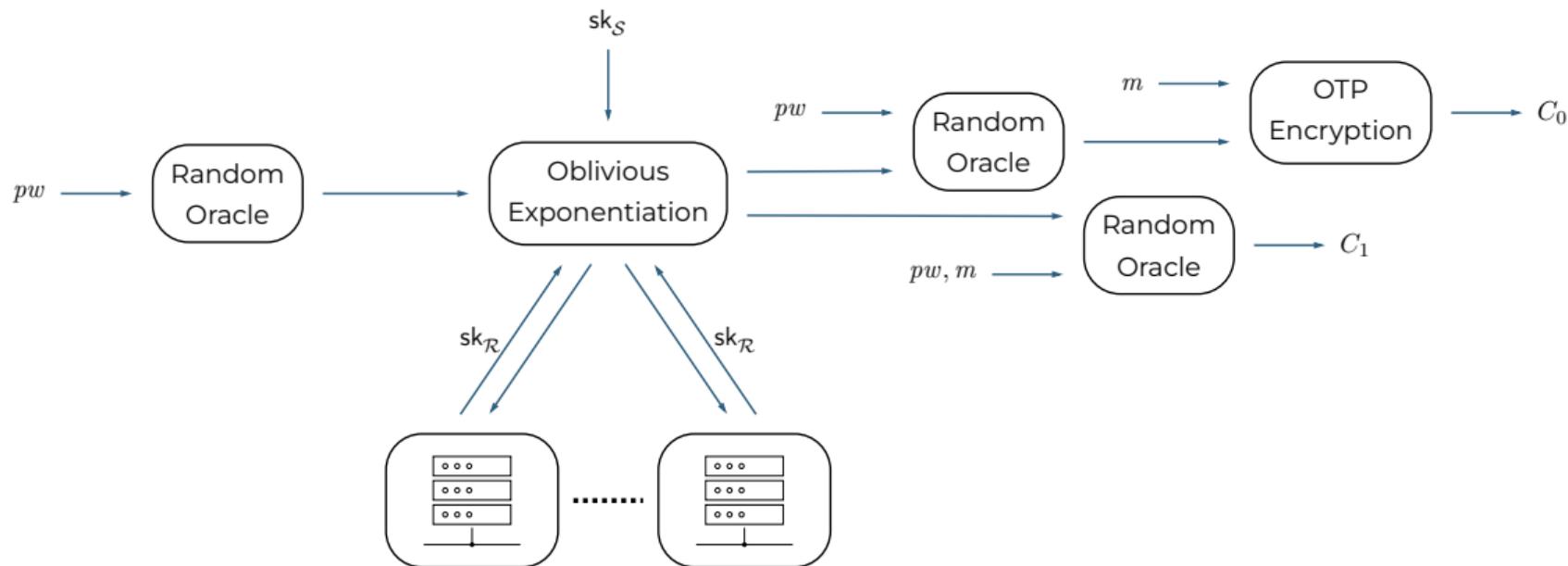
The Oblivious Exponentiation part is similar to **BGRS'25** with a different secret sharing structure

UCPY: A **Composable** TPHE Protocol



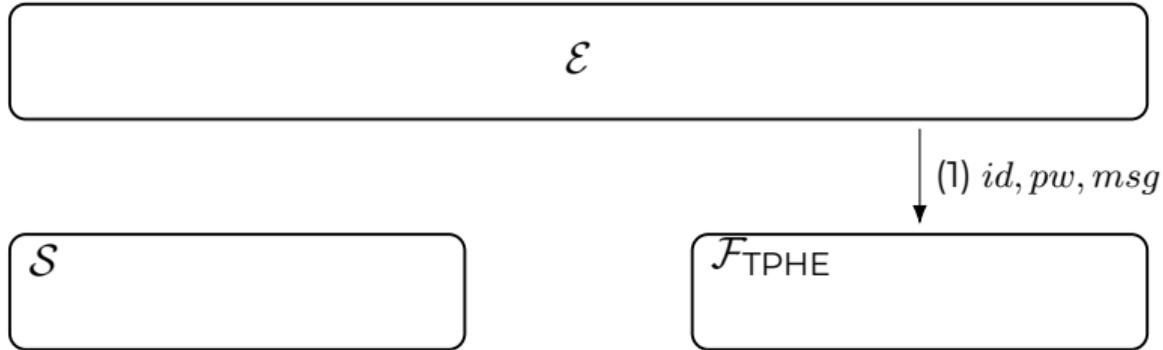
The Oblivious Exponentiation part is similar to **BGRS'25** with a different secret sharing structure

UCPY: A **Composable** TPHE Protocol

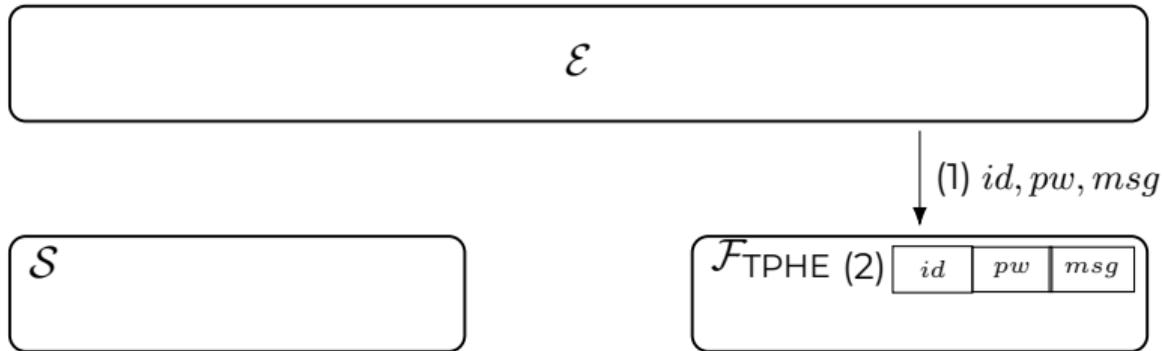


The Oblivious Exponentiation part is similar to **BGRS'25** with a different secret sharing structure

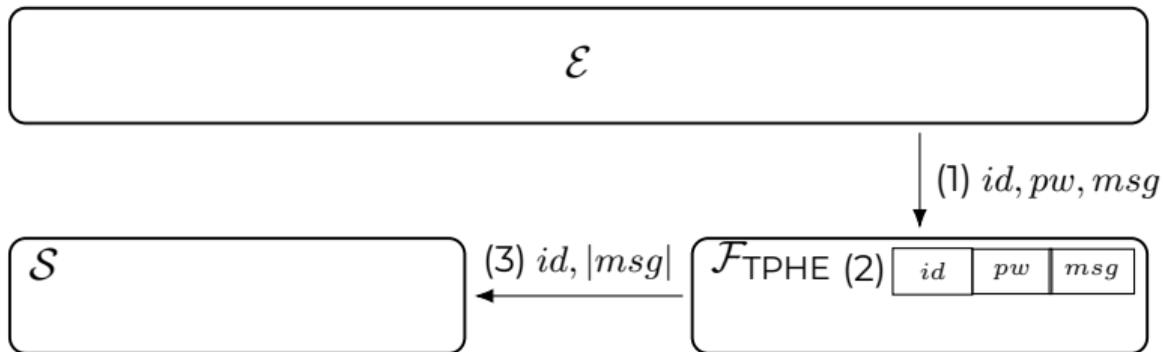
UCPY: Simulating Encryption



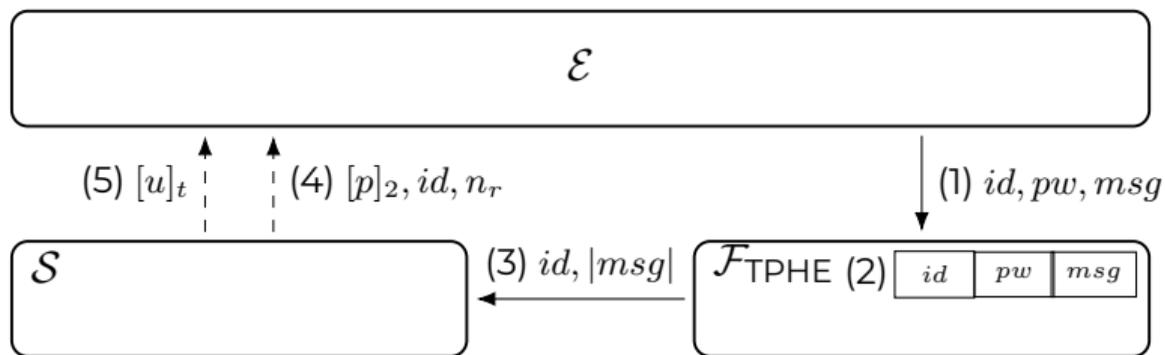
UCPY: Simulating Encryption



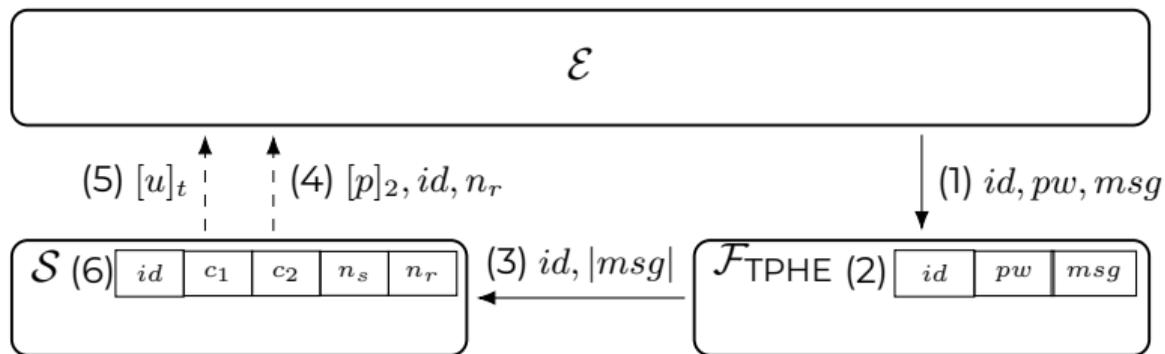
UCPY: Simulating Encryption



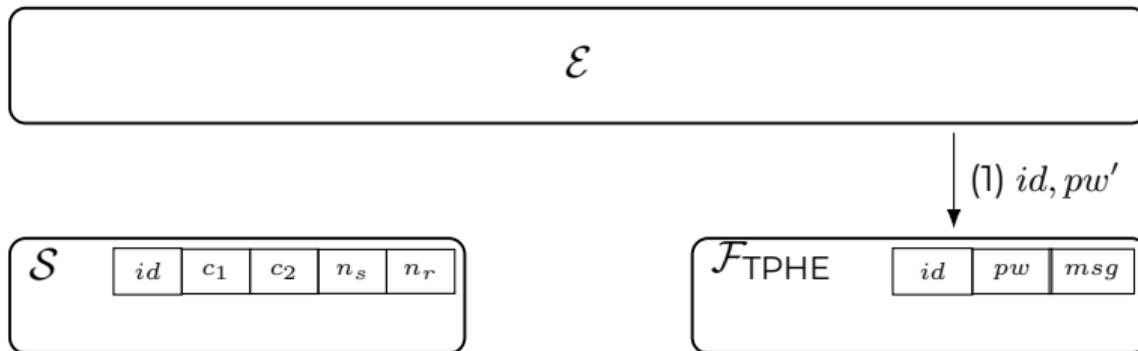
UCPY: Simulating Encryption



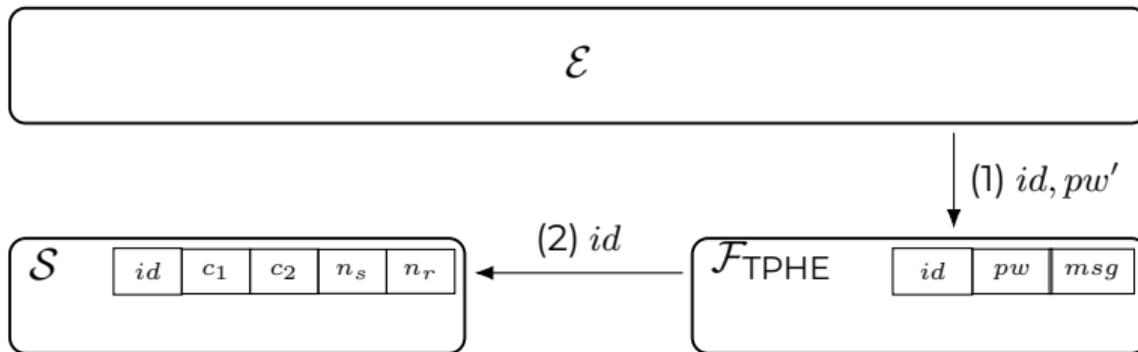
UCPY: Simulating Encryption



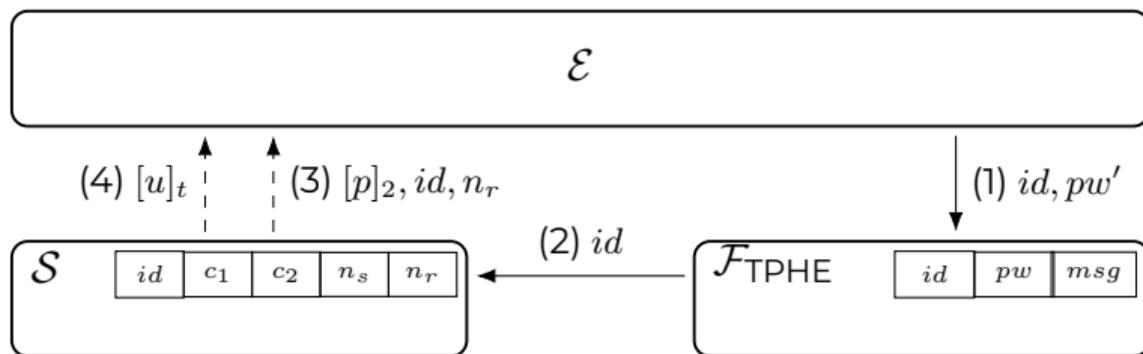
UCPY: Simulating Decryption



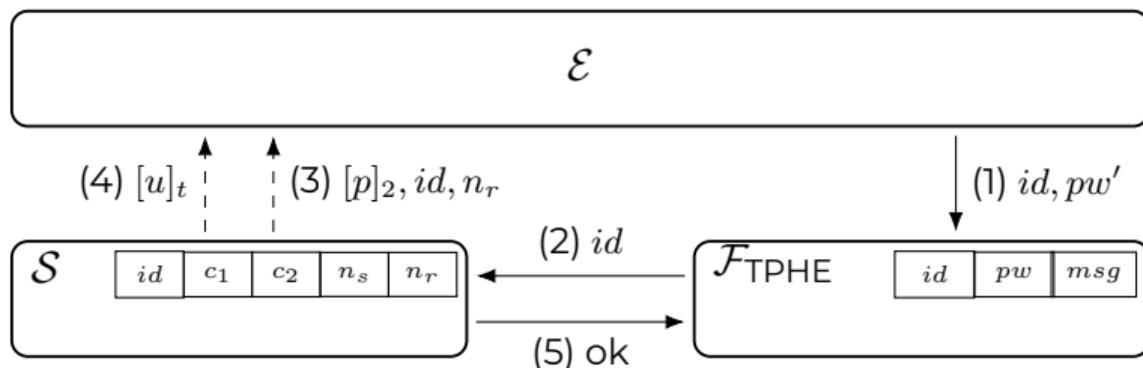
UCPY: Simulating Decryption



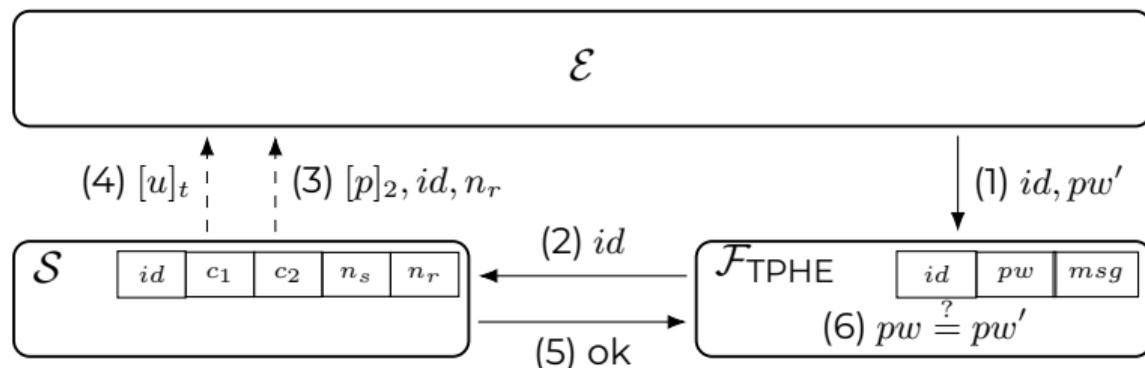
UCPY: Simulating Decryption



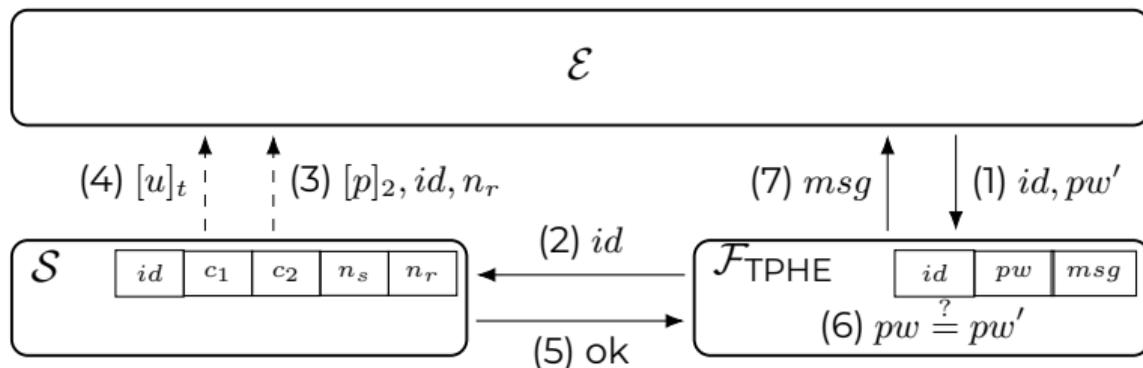
UCPY: Simulating Decryption



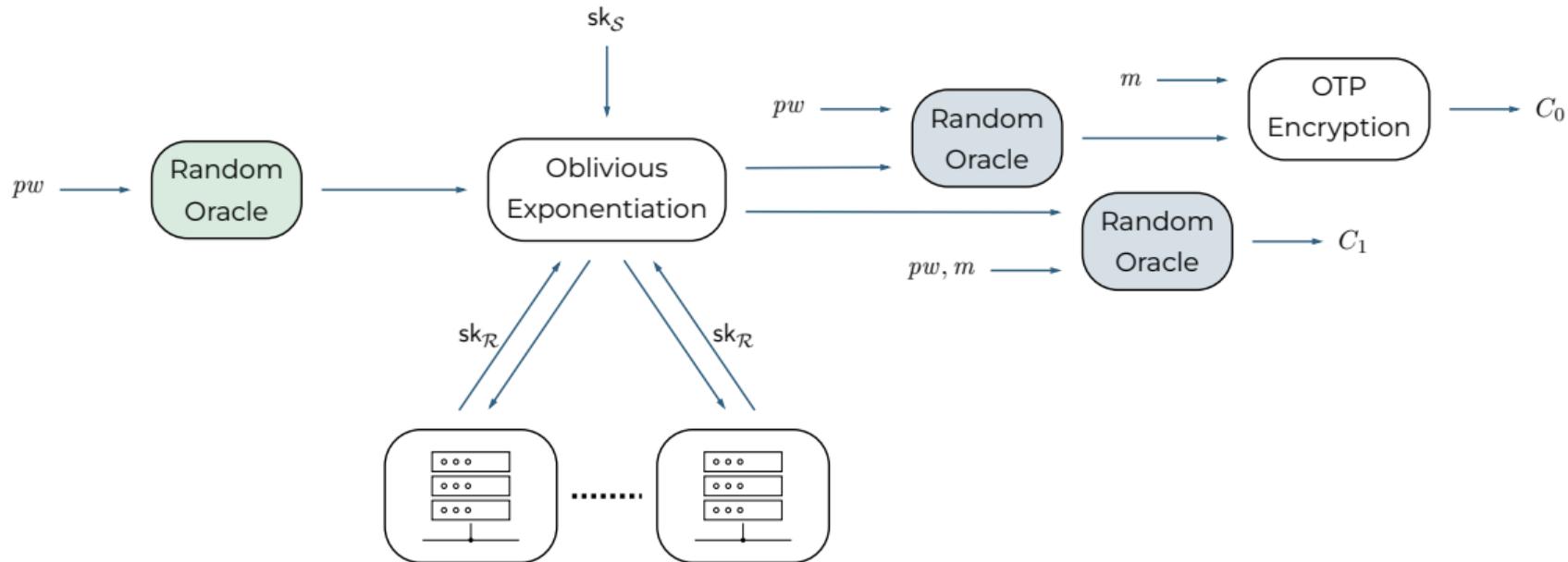
UCPY: Simulating Decryption



UCPY: Simulating Decryption



UCPY: Simulating Random Oracles



$$\begin{aligned}
& 1. \forall id \in \text{storage}^{\text{Real}} : (c_1^{\text{Real}}, c_2^{\text{Real}}, n^{\text{Real}}) \sim (c_1^{\text{Sim}}, c_2^{\text{Sim}}, n^{\text{Sim}}) \wedge [\\
& \quad [\exists i \in \mathbb{N} \text{ s.t. } (id \in \text{correctMessageIDs}^{\text{Ideal}} \implies \text{storageHistory}^{\text{Ideal}}[id, i + 1] = \perp) \\
& \quad \wedge H_{\text{OTP}}^{\text{Real}}(\text{sk}^{\text{Real}} \cdot H_1^{\text{Real}}(pw^{\text{Ideal}}, n^{\text{Real}}) \cdot H_2^{\text{Real}}(id, n^{\text{Real}}), pw^{\text{Ideal}}, id, n^{\text{Real}}) = c_1^{\text{Real}} \oplus m^{\text{Ideal}} \\
& \quad \wedge H_{\text{MAC}}^{\text{Real}}(\text{sk}^{\text{Real}} \cdot H_1^{\text{Real}}(pw^{\text{Ideal}}, n^{\text{Real}}) \cdot H_2^{\text{Real}}(id, n^{\text{Real}}), m^{\text{Ideal}}, pw^{\text{Ideal}}, id, n^{\text{Real}}) = c_2^{\text{Real}} \\
& \quad \wedge H_{\text{OTP}}^{\text{Sim}}(\text{sk}^{\text{Sim}} \cdot H_1^{\text{Sim}}(pw^{\text{Ideal}}, n^{\text{Sim}}) \cdot H_2^{\text{Sim}}(id, n^{\text{Sim}}), pw^{\text{Ideal}}, id, n^{\text{Sim}}) \in \{c_1^{\text{Sim}} \oplus m^{\text{Ideal}}, \perp\} \\
& \quad \wedge H_{\text{MAC}}^{\text{Sim}}(\text{sk}^{\text{Sim}} \cdot H_1^{\text{Sim}}(pw^{\text{Ideal}}, n^{\text{Sim}}) \cdot H_2^{\text{Sim}}(id, n^{\text{Sim}}), m^{\text{Ideal}}, pw^{\text{Ideal}}, id, n^{\text{Sim}}) \in \{c_2^{\text{Sim}}, \perp\}] \\
& \quad \vee [\exists (id^{\text{Sim}}, n^{\text{Sim}}, u, m^{\text{Sim}}, pw^{\text{Sim}}) \in \text{injectedMessages}^{\text{Sim}} \text{ s.t. } id = id^{\text{Sim}} \\
& \quad \wedge c_1^{\text{Real}} = H_{\text{OTP}}^{\text{Real}}(\text{sk}^{\text{Real}} \cdot H_1^{\text{Real}}(pw^{\text{Sim}}, n^{\text{Real}}) \cdot H_2^{\text{Real}}(id, n^{\text{Real}}), pw^{\text{Sim}}, id, n^{\text{Real}}) \oplus m^{\text{Sim}} \\
& \quad \wedge c_2^{\text{Real}} = H_{\text{MAC}}^{\text{Real}}(\text{sk}^{\text{Real}} \cdot H_1^{\text{Real}}(pw^{\text{Sim}}, n^{\text{Real}}) \cdot H_2^{\text{Real}}(id, n^{\text{Real}}), m^{\text{Sim}}, pw^{\text{Sim}}, id, n^{\text{Real}}) \\
& \quad \wedge c_1^{\text{Sim}} = H_{\text{OTP}}^{\text{Sim}}(\text{sk}^{\text{Sim}} \cdot H_1^{\text{Sim}}(pw^{\text{Sim}}, n^{\text{Sim}}) \cdot H_2^{\text{Sim}}(id, n^{\text{Sim}}), pw^{\text{Sim}}, id, n^{\text{Sim}}) \oplus m^{\text{Sim}} \\
& \quad \wedge c_2^{\text{Sim}} = H_{\text{MAC}}^{\text{Sim}}(\text{sk}^{\text{Sim}} \cdot H_1^{\text{Sim}}(pw^{\text{Sim}}, n^{\text{Sim}}) \cdot H_2^{\text{Sim}}(id, n^{\text{Sim}}), m^{\text{Sim}}, pw^{\text{Sim}}, id, n^{\text{Sim}})] \\
& \quad \vee [\exists (pw, m) \text{ s.t.} \\
& \quad c_1^{\text{Real}} = H_{\text{OTP}}^{\text{Real}}(\text{sk}^{\text{Real}} \cdot H_1^{\text{Real}}(pw, n^{\text{Real}}) \cdot H_2^{\text{Real}}(id, n^{\text{Real}}), pw, id, n^{\text{Real}}) \oplus m \\
& \quad \wedge c_2^{\text{Real}} = H_{\text{MAC}}^{\text{Real}}(\text{sk}^{\text{Real}} \cdot H_1^{\text{Real}}(pw, n^{\text{Real}}) \cdot H_2^{\text{Real}}(id, n^{\text{Real}}), m, pw, id, n^{\text{Real}}) \\
& \quad \wedge c_1^{\text{Sim}} = H_{\text{OTP}}^{\text{Sim}}(\text{sk}^{\text{Sim}} \cdot H_1^{\text{Sim}}(pw, n^{\text{Sim}}) \cdot H_2^{\text{Sim}}(id, n^{\text{Sim}}), pw, id, n^{\text{Sim}}) \oplus m \\
& \quad \wedge c_2^{\text{Sim}} = H_{\text{MAC}}^{\text{Sim}}(\text{sk}^{\text{Sim}} \cdot H_1^{\text{Sim}}(pw, n^{\text{Sim}}) \cdot H_2^{\text{Sim}}(id, n^{\text{Sim}}), m, pw, id, n^{\text{Sim}})]] \\
& \quad \text{with } \text{storage}^{\text{Real}}[id] = (c_1^{\text{Real}}, c_2^{\text{Real}}, n^{\text{Real}}), \\
& \quad \text{storage}^{\text{Sim}}[id] = (c_1^{\text{Sim}}, c_2^{\text{Sim}}, n^{\text{Sim}}), \\
& \quad \text{storageHistory}^{\text{Ideal}}[id, i] = (pw^{\text{Ideal}}, m^{\text{Ideal}}) \\
& 2. \forall id \in \text{reqQueue}_{\text{Enc}}^{\text{Real}} : (pw^{\text{Real}}, m^{\text{Real}}, T^{\text{Real}}) = (pw^{\text{Ideal}}, m^{\text{Ideal}}, T^{\text{Sim}}) \wedge |m^{\text{Real}}| = |m^{\text{Sim}}| = |m^{\text{Ideal}}| \\
& \quad \wedge i^{\text{Sim}} = i^{\text{Ideal}} \wedge n^{\text{Real}} \sim n^{\text{Sim}} \\
& \quad \text{with } \text{reqQueue}_{\text{Enc}}^{\text{Real}}[id] = (pw^{\text{Real}}, m^{\text{Real}}, r, n^{\text{Real}}, T^{\text{Real}}), \\
& \quad \text{reqQueue}_{\text{Enc}}^{\text{Sim}}[id] = (i^{\text{Sim}}, |m^{\text{Sim}}|, n^{\text{Real}}, T^{\text{Real}}, p) \\
& \quad \text{storageHistory}^{\text{Ideal}}[id, i^{\text{Ideal}}] = (pw^{\text{Ideal}}, m^{\text{Ideal}}) \text{ with } i^{\text{Ideal}} \max \\
& 3. \forall \text{retrieveCounter} \in \text{reqQueue}_{\text{Dec}}^{\text{Real}} : \exists \text{retrieveCounter}' \in \text{reqQueue}_{\text{Dec}}^{\text{Ideal}} \text{ s.t. } (pw'^{\text{Real}}, T^{\text{Real}}) = (pw'^{\text{Ideal}}, T^{\text{Sim}}) \\
& \quad \wedge (id^{\text{Real}}, \text{caller}^{\text{Real}}) = (id^{\text{Sim}}, \text{caller}^{\text{Sim}}) = (id^{\text{Ideal}}, \text{caller}^{\text{Ideal}}) \wedge (n^{\text{Real}}, p^{\text{Real}}) \sim (n^{\text{Sim}}, p^{\text{Sim}}) \\
& \quad \text{with } \text{reqQueue}_{\text{Dec}}^{\text{Real}}[\text{retrieveCounter}] = (id^{\text{Real}}, pw^{\text{Real}}, r, n^{\text{Real}}, T^{\text{Real}}, p^{\text{Real}}, \text{caller}^{\text{Real}}), \\
& \quad \text{reqQueue}_{\text{Dec}}^{\text{Sim}}[\text{retrieveCounter}] = (id^{\text{Sim}}, n^{\text{Sim}}, T^{\text{Sim}}, p^{\text{Sim}}, \text{caller}^{\text{Sim}}) \\
& \quad \text{reqQueue}_{\text{Dec}}^{\text{Ideal}}[\text{retrieveCounter}] = (id^{\text{Ideal}}, \text{storageHistory}[id, i]^{\text{Ideal}}, pw'^{\text{Ideal}}, \text{caller}^{\text{Ideal}}) \\
& 4. \forall id \in \text{retrieveRate}^{\text{Real}}, rl \notin \text{currentlyCorrupted} : \text{retrieveRate}_{rl}^{\text{Real}}[id] = \text{retrieveRate}_{rl}^{\text{Sim}}[id] \leq \\
& \quad \leq \text{retrieveRate}_{rl}^{\text{Ideal}}[id] - |\{(rl, _, id, _) \in \text{retrieveRequests}^{\text{Sim}}\}| \\
& 5. \forall id, i \in \text{storageHistory}^{\text{Sim}} : H_{\text{OTP}}^{\text{Sim}}(\text{sk}^{\text{Sim}} \cdot H_1^{\text{Sim}}(pw^{\text{Ideal}}, n^{\text{Sim}}) \cdot H_2^{\text{Sim}}(id, n^{\text{Sim}}), pw^{\text{Ideal}}, id, n^{\text{Sim}}) \in \{c_1^{\text{Sim}} \oplus m^{\text{Ideal}}, \perp\} \\
& \quad \wedge H_{\text{MAC}}^{\text{Sim}}(\text{sk}^{\text{Sim}} \cdot H_1^{\text{Sim}}(pw^{\text{Ideal}}, n^{\text{Sim}}) \cdot H_2^{\text{Sim}}(id, n^{\text{Sim}}), m^{\text{Ideal}}, pw^{\text{Ideal}}, id, n^{\text{Sim}}) \in \{c_2^{\text{Sim}}, \perp\}] \\
& \quad \text{with } \text{storageHistory}^{\text{Sim}}[id, i] = (c_1^{\text{Sim}}, c_2^{\text{Sim}}, n^{\text{Sim}}), \\
& \quad \text{storageHistory}^{\text{Ideal}}[id, i] = (pw^{\text{Ideal}}, m^{\text{Ideal}})
\end{aligned}$$

Fig. 27: Invariants for the notion of synchronization between the ideal and real world

Why Threshold PHE?

Let's add UC security...

...and find a (round-optimal) instantiation

Why Threshold PHE?

Let's add UC security...

...and find a (round-optimal) instantiation

We also found a gap in the proof of BELSSZ'20

A Gap in BELSSZ'20

Dec(\dots)		Rate-Limiter $R_L(\text{rate}, \delta_{\text{in}}, \delta_{\text{out}}, \forall t \in [m])$
Server $S(\text{rate}, \delta_{\text{in}}, \delta_{\text{out}}, \text{pw}, \kappa, C)$ returns $(K, \hat{U}, \hat{V}, \hat{W})$ $(G, C) \leftarrow \text{SKE.Dec}(\text{pw}, C)$ returns $(G, \kappa, C) \neq \perp$ $X_0 \leftarrow H_0(\kappa), X_1 \leftarrow H_1(\kappa)$ $Y_{\text{enc}} \leftarrow G^{\text{enc}} \cdot H_0(\text{pw}, \kappa)$		returns $(K, \hat{U}, \hat{V}, \hat{W})$ $X_0 \leftarrow H_0(\kappa), X_1 \leftarrow H_1(\kappa)$ $Y_{\text{enc}} \leftarrow G^{\text{enc}}$
Computing encryption of $Z = Y_{\text{enc}}^{-1} \cdot \prod_{i \in P} V_i^{\delta_i^{\text{enc}}}$ for some t -subset $P \subseteq [m]$		
$K \leftarrow K_0 \cdot K_1$ $\kappa_0 \leftarrow \text{KDF}_0(\hat{U}, \hat{V}, \hat{W}) = (G^{\text{enc}}, K^{\text{enc}}, V_{K_0}^{\text{enc}})$ $S_j = \prod_{i \in P} \hat{V}_i^{\delta_i^{\text{enc}}}, \forall j \in [m]$ $\pi_{1,j} \leftarrow \text{Proof}(\text{enc}, (G, \hat{U}, \hat{V}, \kappa_0))$ $\mathcal{P} = \left\{ j \in [m] : \text{Ver} \left(\text{enc} \left(\frac{G}{K_0}, \frac{\hat{U}}{K}, \frac{\hat{V}}{V_j} \right), \pi_{1,j} \right) = 1 \right\}$ returns $(\mathcal{P}) \geq t$ $\mathcal{P} \leftarrow \text{Subset}(\mathcal{P})$	$\hat{U}, \hat{V}, \kappa_{1,t}$	$K \leftarrow K_0 \cdot K_1$ $\kappa_1 \leftarrow \text{KDF}_1(\hat{U}, \hat{V}, \hat{W}) = (G^{\text{enc}}, K^{\text{enc}}, V_{K_1})$ $S_j = G^{\text{enc}} \cdot \hat{V}_j^{\delta_j^{\text{enc}}}, \forall j \in [m] \setminus \{t\}$ $S_t = G^{\text{enc}}$ $\pi_{1,t} \leftarrow \text{Proof} \left(\text{enc} \left(\frac{G}{K_0}, \frac{\hat{U}}{K}, \frac{\hat{V}}{V_t} \right), \pi_{1,t} \right)$ returns $\forall j \in P \cup \{t\} : \text{Ver} \left(\text{enc} \left(\frac{G}{K_0}, \frac{\hat{U}}{K}, \frac{\hat{V}}{V_j} \right), \pi_{1,j} \right)$ returns $\forall j \in P \cup \{t\} : \text{Ver} \left(\text{enc} \left(\frac{G}{K_0}, \frac{\hat{U}}{K}, \frac{\hat{V}}{V_j} \right), \pi_{1,j} \right)$ $(\hat{U}, \hat{V}) = \left(\prod_{i \in P \cup \{t\}} \hat{U}_i^{\delta_i^{\text{enc}}}, \prod_{i \in P \cup \{t\}} \hat{V}_i^{\delta_i^{\text{enc}}} \right)$
Computing encryption of $(Z^{\text{enc}}, Z^{\text{enc}} \cdot H_1(\text{pw}, \kappa) \cdot H_1(\kappa)^{\sum_{i \in P} \delta_i^{\text{enc}}})$ for some random t' and t''		
$\hat{U}, \hat{V} \leftarrow \text{KDF}_0$ $(\hat{U}, \hat{V}) = (u^{\text{enc}}, v^{\text{enc}}), (U, V) = (u^{\text{enc}}, v^{\text{enc}} \cdot H_1(\text{pw}, \kappa))$ $\pi_{2,t'} \leftarrow \text{Proof}(\text{enc}, (G, \hat{U}, \hat{V}))$ $\pi_{2,t''} \leftarrow \text{Proof}(\text{enc}, (G, U, V))$	$\hat{U}, \hat{V}, \pi_{2,t'}, \pi_{2,t''}$	$\hat{U}, \hat{V} \leftarrow \text{KDF}_0$ $(\hat{U}, \hat{V}) = (u^{\text{enc}}, v^{\text{enc}}), (U, V) = (u^{\text{enc}}, v^{\text{enc}} \cdot X_1^{\text{enc}})$ $\pi_{2,t'} \leftarrow \text{Proof} \left(\text{enc} \left(\frac{G}{V}, \frac{\hat{U}}{V}, \hat{V} \right), \pi_{2,t'} \right)$ $\pi_{2,t''} \leftarrow \text{Proof} \left(\text{enc} \left(\frac{G}{V}, X_1^{\text{enc}}, \frac{\hat{V}}{V} \right), \pi_{2,t''} \right)$ returns $\forall j \in P \cup \{t'\} : \text{Ver} \left(\text{enc} \left(\frac{G}{V}, \frac{\hat{U}}{V}, \hat{V} \right), \pi_{2,j} \right)$ returns $\forall j \in P \cup \{t'\} : \text{Ver} \left(\text{enc} \left(\frac{G}{V}, X_1^{\text{enc}}, \frac{\hat{V}}{V} \right), \pi_{2,j} \right)$ $(\hat{U}, \hat{V}) = \left(\prod_{i \in P \cup \{t'\}} \hat{U}_i^{\delta_i^{\text{enc}}}, \prod_{i \in P \cup \{t'\}} \hat{V}_i^{\delta_i^{\text{enc}}} \right) \wedge \text{Ver} \left(\text{enc}, (G, \hat{U}, \hat{V}, \pi_{2,t'}) \right)$ $(\hat{U}, \hat{V}) = \left(\prod_{i \in P \cup \{t'\}} \hat{U}_i^{\delta_i^{\text{enc}}}, \prod_{i \in P \cup \{t'\}} \hat{V}_i^{\delta_i^{\text{enc}}} \right)$
Join decryption		
$\hat{U} \leftarrow \hat{U}^{\text{enc}}, \hat{V} \leftarrow \hat{V}^{\text{enc}}$ $K_2 = \prod_{i \in P} \hat{V}_i^{\delta_i^{\text{enc}}}, \forall j \in P$ $\pi_{3,j} \leftarrow \text{Proof} \left(\text{enc} \left(\frac{G}{K_2}, \frac{\hat{U}}{K_2}, \hat{V} \right), \pi_{3,j} \right)$ $\pi_{3,t} \leftarrow \text{Proof} \left(\text{enc} \left(\frac{G}{K_2}, \frac{\hat{U}}{K_2}, \hat{V} \right), \pi_{3,t} \right)$ $Z = \prod_{i \in P \cup \{t\}} \hat{U}_i^{\delta_i^{\text{enc}}}, Z' = \prod_{i \in P \cup \{t\}} \hat{V}_i^{\delta_i^{\text{enc}}}$ if $(\hat{V} \neq Z')$ then return \perp $M = G^{\text{enc}} \cdot (Z' \cdot Z^{-1})$ return M	$\hat{U}, \pi_{3,t'}, \pi_{3,t''}$	$Z_1 \leftarrow \hat{U}^{\text{enc}}, Z_2 \leftarrow \hat{V}^{\text{enc}}$ $K_2 = G^{\text{enc}}$ $\pi_{3,t'} \leftarrow \text{Proof} \left(\text{enc} \left(\frac{G}{K_2}, \frac{K_2}{K_2}, \hat{U} \right), \pi_{3,t'} \right)$ $\pi_{3,t''} \leftarrow \text{Proof} \left(\text{enc} \left(\frac{G}{K_2}, \frac{K_2}{K_2}, \hat{V} \right), \pi_{3,t''} \right)$ returns M

Figure 4: Decryption Protocol (Procedures for fine-grained rate-limiting in Figure 8)

A Gap in BELSSZ'20

Dec(...)	Rate-Limiter $R_L(\text{rate}, \delta_{\text{in}}, \delta_{\text{out}}, \forall t \in [m])$
Server $S(\text{rate}, \delta_{\text{in}}, \delta_{\text{out}}, \text{pw}, \kappa, C)$ returns $(\hat{C}, \hat{C}_1, \hat{C}_2)$ $(\hat{C}, \hat{C}_1) \leftarrow \text{SKE}(\text{Dec}(\text{pw}, C))$ returns $(\hat{C}, \hat{C}_1, \hat{C}_2) \neq \perp$ $X_0 \leftarrow H_0(\kappa), X_1 \leftarrow H_1(\kappa)$ $X_2 \leftarrow G^{\text{rate}} \cdot H_2(\text{pw}, \kappa)$	returns $(\hat{C}, \hat{C}_1, \hat{C}_2)$ $X_0 \leftarrow H_0(\kappa), X_1 \leftarrow H_1(\kappa)$ $X_2 \leftarrow G^{\text{rate}}$
Computing encryption of $Z = X_0^{-1} \cdot \prod_{i \in P} V_i^{K_i^{r_i}}$ for some t -subset $P \subseteq [m]$	
$K \leftarrow K_0 \cdot K_1$ $\kappa_0 \leftarrow \text{KDF}(\hat{C}, \hat{C}_1, \hat{C}_2) = (G^{\text{rate}}, K^{\text{rate}}, V_K^{\text{rate}})$ $S_j = \prod_{i \in P} \hat{C}_i^{K_i^{r_i}}, \forall j \in [m]$ $\pi_{1,j} \leftarrow \text{Prove}(\text{enc}(\hat{C}, \hat{C}_1, \kappa_0))$ $\mathcal{P} = \left\{ j \in [m] : \text{Ver} \left(\text{enc} \left(\begin{matrix} \hat{C} & \hat{C}_1 \\ X_0 & K \end{matrix}, \begin{matrix} \hat{C}_2 \\ V_j \end{matrix} \right), \pi_{1,j} \right) = 1 \right\}$ returns $\mathcal{P} \geq t$ $\mathcal{P} \leftarrow \text{Subset}(\mathcal{P})$	$K \leftarrow K_0 \cdot K_1$ $\kappa_0 \leftarrow \text{KDF}(\hat{C}, \hat{C}_1, \hat{C}_2) = (G^{\text{rate}}, K^{\text{rate}}, V_K^{\text{rate}})$ $S_j = G^{\text{rate}} \cdot \hat{C}_i^{K_i^{r_i}}, \forall j \in [m] \setminus \{t\}$ $S_t = G^{\text{rate}}$ $\pi_{1,t} \leftarrow \text{Prove} \left(\text{enc} \left(\begin{matrix} \hat{C} & \hat{C}_1 \\ X_0 & K \end{matrix}, \begin{matrix} \hat{C}_2 \\ V_t \end{matrix} \right), \pi_{1,t} \right)$ returns $\forall j \in P \setminus \{t\} : \text{Ver} \left(\text{enc} \left(\begin{matrix} \hat{C} & \hat{C}_1 \\ X_0 & K \end{matrix}, \begin{matrix} \hat{C}_2 \\ V_j \end{matrix} \right), \pi_{1,j} \right)$ returns $\forall t : \text{Ver} \left(\text{enc}(\hat{C}, \hat{C}_1, \kappa_0) \right)$ $(\hat{C}, \hat{C}_1) \leftarrow \left(\prod_{i \in P \setminus \{t\}} \hat{C}_i^{K_i^{r_i}}, \prod_{i \in P \setminus \{t\}} V_i^{K_i^{r_i}} \right)$
Computing encryption of $(Z^{\text{rate}}, Z^{\text{rate}} \cdot H_1(\text{pw}, \kappa) \cdot H_2(\kappa)^{\text{rate}} \cdot P^{\text{rate}})$ for some random t and t'	
$\hat{\kappa}_0, \hat{\kappa}_1 \leftarrow \text{KDF}$ $(\hat{C}_0, \hat{C}_1) \leftarrow (G^{\text{rate}}, V^{\text{rate}}), (\hat{C}_2, \hat{C}_3) \leftarrow (G^{\text{rate}}, V^{\text{rate}} \cdot H_1(\text{pw}, \kappa))$ $\pi_{2,t} \leftarrow \text{Prove}(\text{enc}(\hat{C}_0, \hat{C}_1, \hat{\kappa}_0))$ $\pi_{2,t'} \leftarrow \text{Prove}(\text{enc}(\hat{C}_0, \hat{C}_1, \hat{\kappa}_1))$	$\hat{\kappa}_0, \hat{\kappa}_1 \leftarrow \text{KDF}$ $(\hat{C}_0, \hat{C}_1) \leftarrow (G^{\text{rate}}, V^{\text{rate}}), (\hat{C}_2, \hat{C}_3) \leftarrow (G^{\text{rate}}, V^{\text{rate}} \cdot X_0^{\text{rate}})$ $\pi_{2,t} \leftarrow \text{Prove} \left(\text{enc} \left(\begin{matrix} \hat{C}_0 \\ V \end{matrix}, \begin{matrix} \hat{C}_1 \\ \hat{\kappa}_0 \end{matrix} \right), \pi_{2,t} \right)$ $\pi_{2,t'} \leftarrow \text{Prove} \left(\text{enc} \left(\begin{matrix} \hat{C}_0 \\ V \end{matrix}, \begin{matrix} \hat{C}_1 \\ X_0^{\text{rate}} \cdot \hat{\kappa}_1 \end{matrix} \right), \pi_{2,t'} \right)$
returns $\forall j \in P : \text{Ver} \left(\text{enc} \left(\begin{matrix} \hat{C} \\ V_j \end{matrix}, \begin{matrix} \hat{C}_2 \\ \hat{\kappa}_0 \end{matrix} \right), \pi_{1,j} \right)$ returns $\forall j \in P : \text{Ver} \left(\text{enc} \left(\begin{matrix} \hat{C} \\ V_j \end{matrix}, \begin{matrix} \hat{C}_2 \\ X_0^{\text{rate}} \cdot \hat{\kappa}_1 \end{matrix} \right), \pi_{1,j} \right)$	returns $\forall j \in P \setminus \{t\} : \text{Ver} \left(\text{enc} \left(\begin{matrix} \hat{C} \\ V_j \end{matrix}, \begin{matrix} \hat{C}_2 \\ \hat{\kappa}_0 \end{matrix} \right), \pi_{1,j} \right)$ returns $\forall t \in P \setminus \{t\} : \text{Ver} \left(\text{enc} \left(\begin{matrix} \hat{C} \\ V_j \end{matrix}, \begin{matrix} \hat{C}_2 \\ X_0^{\text{rate}} \cdot \hat{\kappa}_1 \end{matrix} \right), \pi_{1,j} \right)$ returns $\forall t : \text{Ver}(\text{enc}(\hat{C}_0, \hat{C}_1, \kappa_0)) \wedge \text{Ver}(\text{enc}(\hat{C}_0, \hat{C}_1, \hat{\kappa}_0))$ $(\hat{C}, \hat{C}_1) \leftarrow \left(\prod_{i \in P \setminus \{t\}} \hat{C}_i^{K_i^{r_i}}, \prod_{i \in P \setminus \{t\}} V_i^{K_i^{r_i}} \right)$ $(\hat{C}', \hat{C}') \leftarrow \left(\prod_{i \in P \setminus \{t\}} \hat{C}_i^{K_i^{r_i}}, \prod_{i \in P \setminus \{t\}} V_i^{K_i^{r_i}} \right)$
Join decryption	
$\hat{\kappa}_0 \leftarrow G^{\text{rate}}, \hat{\kappa}_1 \leftarrow G^{\text{rate}}$ $K_j = \prod_{i \in P} \hat{C}_i^{K_i^{r_i}}, \forall j \in P$ returns $\forall j \in P : \text{Ver} \left(\text{enc} \left(\begin{matrix} \hat{C} \\ V_j \end{matrix}, \begin{matrix} \hat{C}_2 \\ \hat{\kappa}_0 \end{matrix} \right), \pi_{1,j} \right)$ returns $\forall j \in P : \text{Ver} \left(\text{enc} \left(\begin{matrix} \hat{C} \\ V_j \end{matrix}, \begin{matrix} \hat{C}_2 \\ \hat{\kappa}_1 \end{matrix} \right), \pi_{1,j} \right)$ $T = \prod_{i \in P \setminus \{t\}} V_i^{K_i^{r_i}}, T' = \prod_{i \in P \setminus \{t\}} V_i^{K_i^{r_i}}$ if $(T \neq T')$ then return \perp $M \leftarrow G^{\text{rate}}(V^{\text{rate}}, T^{-1})$ return M	$\kappa_0 \leftarrow G^{\text{rate}}, \kappa_1 \leftarrow G^{\text{rate}}$ $K_i \leftarrow G^{\text{rate}}$ $\pi_{2,t} \leftarrow \text{Prove} \left(\text{enc} \left(\begin{matrix} \hat{C} \\ V_t \end{matrix}, \begin{matrix} \hat{C}_2 \\ \hat{\kappa}_0 \end{matrix} \right), \pi_{2,t} \right)$ $\pi_{2,t'} \leftarrow \text{Prove} \left(\text{enc} \left(\begin{matrix} \hat{C} \\ V_t \end{matrix}, \begin{matrix} \hat{C}_2 \\ \hat{\kappa}_1 \end{matrix} \right), \pi_{2,t'} \right)$ returns κ

Figure 4: Decryption Protocol (Procedures for fine-grained rate-limiting in Figure 8)

A Gap in BELSSZ'20

$$T_0 := \tilde{U}^{k_0}, \quad T'_0 := \tilde{U}'^{k_0}$$

Dec(\dots, \dots)		Rate-limiter $R_0(\cdot, \text{rate } c, \delta_0, \lambda, \forall c \in [m])$
Server $\mathcal{S}(T, T', \tilde{U}, \tilde{U}', \text{pw}, \kappa, C)$ returns (M, τ) $(G, C) \leftarrow \text{SKE.Dec}(\text{sk}, C)$ ensure $(G, C) \neq \perp$ $X_0 \leftarrow H_0(\kappa), X_1 \leftarrow H_1(\kappa)$ $\tilde{U}_0^c \leftarrow C_0^{-1} \cdot H_0(\text{pw}, \kappa)$	$\xrightarrow{\kappa}$	returns (K, V, τ) $X_0 \leftarrow H_0(\kappa), X_1 \leftarrow H_1(\kappa)$ $\tilde{U}_0 \leftarrow X_0^c$
Computing encryption of $Z = \tilde{U}_0^c \cdot \prod_{i \in P} V_i^{k_i^c}$ for some t subset $P \subseteq [m]$		
$K \leftarrow K_0 \cdot K_1$ $\kappa_0 \leftarrow \text{KDF}(\tilde{U}_0, \tilde{U}_1) \in \{0^m, \mathcal{K}^m \cdot \mathbb{F}_q^m\}$ $S_j \leftarrow \prod_{i \in P} \tilde{U}_i^{k_i^c}, \forall j \in [m]$ $\pi_{j,0} \leftarrow \text{Proof}(\text{enc}, (G, \tilde{U}_0, \kappa_0))$ $\mathcal{P} \leftarrow \left\{ j \in [m] : \text{Ver} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ K \\ \tilde{U}_j \end{bmatrix}, \kappa_0 \right) \right) = 1 \right\}$ ensure $ \mathcal{P} \geq t$ $\mathcal{P} \leftarrow \text{Subset}(\mathcal{P})$	$\xrightarrow{G, \tilde{U}_0, \kappa_0}$	$K \leftarrow K_0 \cdot K_1$ $\kappa_1 \leftarrow \text{KDF}(\tilde{U}_0, \tilde{U}_1) \in \{0^m, \mathcal{K}^m \cdot \mathbb{F}_q^m\}$ $S_j \leftarrow \prod_{i \in P} \tilde{U}_i^{k_i^c}, \forall j \in [m] \setminus \{0\}$ $\delta_j \leftarrow G^{\kappa_j}$ $\pi_{j,1} \leftarrow \text{Proof} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ K \\ \tilde{U}_j \end{bmatrix}, \kappa_1 \right) \right)$ ensure $\forall j \in P \setminus \{0\} : \text{Ver} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ K \\ \tilde{U}_j \end{bmatrix}, \kappa_1 \right) \right) = 1$ ensure $\forall j \in P \setminus \{0\} : \text{Ver}(\text{enc}, (G, \tilde{U}_0, \kappa_1))$ $(\tilde{U}, V) \leftarrow \left(\prod_{i \in P \setminus \{0\}} \tilde{U}_i^{k_i^c}, \prod_{i \in P \setminus \{0\}} V_i^{k_i^c} \right)$
Computing encryption of $(Z^t, Z^{t'} \cdot H_1(\text{pw}, \kappa) \cdot H_1(\kappa)^{\sum_{i \in P} \delta_i^c})$ for some random t and t'		
$\tilde{U}_0, \tilde{U}_1^c \leftarrow \text{KDF}$ $(\tilde{U}_0, \tilde{U}_1) \leftarrow (\kappa^m, \nu^m), (\tilde{U}_0^c, \tilde{U}_1^c) \leftarrow (\nu^m, \kappa^m \cdot H_1(\text{pw}, \kappa))$ $\kappa_{j,0} \leftarrow \text{Proof}(\text{enc}, (G, \tilde{U}_0, \tilde{U}_1^c))$ $\kappa_{j,1}^c \leftarrow \text{Proof}(\text{enc}, (G, \tilde{U}_0^c, \tilde{U}_1^c))$	$\xrightarrow{G, \tilde{U}_0, \tilde{U}_1^c, \tilde{U}_0^c, \tilde{U}_1^c, \kappa_{j,0}^c, \kappa_{j,1}^c}$	$\tilde{U}_0, \tilde{U}_1^c \leftarrow \text{KDF}$ $(\tilde{U}_0, \tilde{U}_1) \leftarrow (\kappa^m, \nu^m), (\tilde{U}_0^c, \tilde{U}_1^c) \leftarrow (\nu^m, \kappa^m \cdot H_1(\text{pw}, \kappa))$ $\kappa_{j,0} \leftarrow \text{Proof}(\text{enc}, (G, \tilde{U}_0, \tilde{U}_1^c))$ $\kappa_{j,1}^c \leftarrow \text{Proof} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ X_0^{k_0^c} \\ \tilde{U}_j \end{bmatrix}, \kappa_{j,0}^c \right) \right)$ $\kappa_{j,1}^c \leftarrow \text{Proof} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ X_0^{k_0^c} \\ \tilde{U}_j \end{bmatrix}, \kappa_{j,1}^c \right) \right)$
ensure $\forall j \in P : \text{Ver} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ \tilde{U}_j \end{bmatrix}, \kappa_{j,0} \right) \right) = 1$ ensure $\forall j \in P : \text{Ver} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ X_0^{k_0^c} \\ \tilde{U}_j \end{bmatrix}, \kappa_{j,1}^c \right) \right) = 1$	$\xrightarrow{(\tilde{U}_0, \tilde{U}_1, \tilde{U}_0^c, \tilde{U}_1^c, \kappa_{j,0}^c, \kappa_{j,1}^c)_{i \in P \setminus \{0\}}}$	ensure $\forall j \in P \setminus \{0\} : \text{Ver} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ \tilde{U}_j \end{bmatrix}, \kappa_{j,0} \right) \right) = 1$ ensure $\forall j \in P \setminus \{0\} : \text{Ver} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ X_0^{k_0^c} \\ \tilde{U}_j \end{bmatrix}, \kappa_{j,1}^c \right) \right) = 1$ ensure $\forall j \in P \setminus \{0\} : \text{Ver}(\text{enc}, (G, \tilde{U}_0, \tilde{U}_1)) \wedge \text{Ver}(\text{enc}, (G, \tilde{U}_0^c, \tilde{U}_1^c))$ $(\tilde{U}, V) \leftarrow \left(\prod_{i \in P \setminus \{0\}} \tilde{U}_i^{k_i^c}, \prod_{i \in P \setminus \{0\}} V_i^{k_i^c} \right)$ $(\tilde{U}', V') \leftarrow \left(\prod_{i \in P \setminus \{0\}} \tilde{U}_i^{\delta_i^c}, \prod_{i \in P \setminus \{0\}} V_i^{\delta_i^c} \right)$
Join decryption.....		
$\tilde{U}_0 \leftarrow G^{\kappa_0}, \tilde{U}_1^c \leftarrow G^{\kappa_1^c}$ $K_j \leftarrow \prod_{i \in P} \tilde{U}_i^{k_i^c}, \forall j \in P$ ensure $\forall j \in P : \text{Ver} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ \tilde{U}_j \end{bmatrix}, \kappa_{j,0} \right) \right) = 1$ ensure $\forall j \in P : \text{Ver} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ X_0^{k_0^c} \\ \tilde{U}_j \end{bmatrix}, \kappa_{j,1}^c \right) \right) = 1$ $T \leftarrow \prod_{i \in P \setminus \{0\}} V_i^{k_i^c}, T' \leftarrow \prod_{i \in P \setminus \{0\}} V_i^{\delta_i^c}$ if $(\tilde{V} \neq T)$ then return \perp $M \leftarrow G^{\kappa_0} \cdot (T^t \cdot T'^{t'})$ return M	$\xrightarrow{T_0, \kappa_0, T'_0, \kappa_1}$	$\tau_0 \leftarrow G^{\kappa_0}, \tau_1^c \leftarrow G^{\kappa_1^c}$ $K_0 \leftarrow G^{\kappa_0}$ $\kappa_{j,0} \leftarrow \text{Proof} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ \tilde{U}_j \end{bmatrix}, \kappa_{j,0} \right) \right)$ $\kappa_{j,1}^c \leftarrow \text{Proof} \left(\text{enc} \left(\begin{bmatrix} \tilde{U}_0 \\ \tilde{U}_1 \\ X_0^{k_0^c} \\ \tilde{U}_j \end{bmatrix}, \kappa_{j,1}^c \right) \right)$

Figure 4: Decryption Protocol (Procedures for fine-grained rate-limiting in Figure 8)

A Gap in BELSSZ'20

Dec(\dots)		Rate-Limiter $\mathcal{R}_i(\text{rate}, \text{sk}_i, \forall i \in [m])$
Server $\mathcal{S}(1^{\lambda}, 1^{\epsilon}, \text{pk}, \text{pw}, \kappa, C)$ returns $(U, V, \tilde{U}, \tilde{V})$ $(G, C) \leftarrow \text{SKE}(\text{Dec}(\text{sk}, C))$ returns $(G, C, \kappa) \neq \perp$ $X_0 \leftarrow \mathcal{H}_0(\kappa), X_1 \leftarrow \mathcal{H}_1(\kappa)$ $Y_{0,1} \leftarrow \mathcal{C}_{\text{enc}}^{-1}(\text{rk}_0(\text{pw}, \kappa))$		returns $(\text{sk}_i^0, \text{sk}_i^1)$ $X_i \leftarrow \mathcal{H}_0(\kappa), X_i \leftarrow \mathcal{H}_1(\kappa)$ $Y_{i,0} \leftarrow \mathcal{C}_{\text{enc}}^0$
Computing encryption of $Z = Y_{0,1}^{-1} \cdot \prod_{i \in P} Y_{i,0}^{r_i}$ for some t -subset $P \subseteq [m]$		
$K \leftarrow K_0 \oplus K_1$ $\kappa_0 \leftarrow \mathcal{R}_0, (\tilde{U}_0, \tilde{V}_0) \leftarrow (G^{\kappa_0}, K^{\kappa_0} \cdot Y_{0,1})$ $S_j \leftarrow \prod_{i \in P} \mathcal{C}_{\text{enc}}^{r_i} \cdot Y_j \in [m]$ $\kappa_{1,t} \leftarrow \text{Prave}(\text{enc}, (G, \tilde{U}_0, \kappa_1))$ $\mathcal{P} \leftarrow \left\{ j \in [m] : \text{Vt} \left(\text{enc} \left(\frac{G}{K_0}, \frac{t}{K_1}, \frac{\tilde{U}_0}{\tilde{V}_0} \right), \kappa_{1,t} \right) = 1 \right\}$ returns $\mathcal{P} \geq t$ $\mathcal{P} \leftarrow \text{Subst}(\mathcal{P})$		$K \leftarrow K_0 \oplus K_1$ $\kappa_1 \leftarrow \mathcal{R}_1, (\tilde{U}_1, \tilde{V}_1) \leftarrow (G^{\kappa_1}, K^{\kappa_1} \cdot Y_{1,0})$ $S_j \leftarrow \prod_{i \in P} \mathcal{C}_{\text{enc}}^{r_i} \cdot Y_j \in [m] \setminus \{t\}$ $S_t \leftarrow G^{\kappa_1}$ $\kappa_{1,t} \leftarrow \text{Prave} \left(\text{enc} \left(\frac{G}{K_0}, \frac{t}{K_1}, \frac{\tilde{U}_1}{\tilde{V}_1} \right), \kappa_{1,t} \right)$
Computing encryption of $(Z^t, Z^{t'}) \cdot \mathcal{H}_1(\text{pw}, \kappa) \cdot \mathcal{H}_1(\kappa)^{\sum_{i \in P} r_i \cdot \text{pw}_i}$ for some random t'		
$(\tilde{U}, \tilde{V}) \leftarrow \left(\prod_{i \in P, i \neq t} \mathcal{C}_{\text{enc}}^{r_i}, \prod_{i \in P, i \neq t} Y_i^{r_i} \right)$ $\tilde{u}, \tilde{v} \leftarrow \mathcal{R}_2$ $(\tilde{U}_0, \tilde{V}_0) \leftarrow (G^{\kappa_0}, \tilde{U}_0), (\tilde{U}_1, \tilde{V}_1) \leftarrow (G^{\kappa_1}, \tilde{U}_1)$ $\kappa_{2,t} \leftarrow \text{Prave}(\text{enc}, (G, \tilde{U}_0, \tilde{u}))$ $\kappa_{2,t'} \leftarrow \text{Prave}(\text{enc}, (G, \tilde{U}_1, \tilde{v}))$ returns $\forall j \in P : \text{Vt} \left(\text{enc} \left(\frac{G}{V}, \frac{\tilde{U}_j}{\tilde{V}_j} \right), \kappa_{2,t} \right)$ returns $\forall j \in P : \text{Vt} \left(\text{enc} \left(\frac{G}{V}, \frac{X_j^{t'} \cdot \tilde{U}_j}{\tilde{V}_j}, \frac{\tilde{U}_j}{\tilde{V}_j} \right), \kappa_{2,t'} \right)$ $(\tilde{U}, \tilde{V}) \leftarrow \left(\prod_{i \in P, i \neq t} \tilde{U}_i, \prod_{i \in P, i \neq t} \tilde{V}_i \right)$ $(\tilde{U}', \tilde{V}') \leftarrow \left(\prod_{i \in P, i \neq t} \tilde{U}'_i, \prod_{i \in P, i \neq t} \tilde{V}'_i \right)$		$\kappa_{2,t} \leftarrow \mathcal{R}_3$ $(\tilde{U}_0, \tilde{V}_0) \leftarrow (G^{\kappa_0}, \tilde{U}_0), (\tilde{U}'_0, \tilde{V}'_0) \leftarrow (G^{\kappa_0}, \tilde{U}'_0)$ $\kappa_{2,t} \leftarrow \text{Prave} \left(\text{enc} \left(\frac{G}{V}, \frac{\tilde{U}_0}{\tilde{V}_0} \right), \kappa_{2,t} \right)$ $\kappa_{2,t'} \leftarrow \text{Prave} \left(\text{enc} \left(\frac{G}{V}, \frac{X_t^{t'} \cdot \tilde{U}_0}{\tilde{V}_0}, \frac{\tilde{U}_0}{\tilde{V}_0} \right), \kappa_{2,t'} \right)$ returns $\forall j \in P \setminus \{t\} : \text{Vt} \left(\text{enc} \left(\frac{G}{V}, \frac{\tilde{U}_j}{\tilde{V}_j} \right), \kappa_{2,t} \right)$ returns $\forall j \in P \setminus \{t\} : \text{Vt} \left(\text{enc} \left(\frac{G}{V}, \frac{X_j^{t'} \cdot \tilde{U}_j}{\tilde{V}_j}, \frac{\tilde{U}_j}{\tilde{V}_j} \right), \kappa_{2,t'} \right)$ returns $\forall j \in [m] \setminus P : \text{Vt} \left(\text{enc} \left(\frac{G}{V}, \frac{\tilde{U}_j}{\tilde{V}_j} \right), \kappa_{2,t} \right) \wedge \text{Vt} \left(\text{enc} \left(\frac{G}{V}, \frac{\tilde{U}'_j}{\tilde{V}'_j} \right), \kappa_{2,t'} \right)$ $(\tilde{U}, \tilde{V}) \leftarrow \left(\prod_{i \in P, i \neq t} \tilde{U}_i, \prod_{i \in P, i \neq t} \tilde{V}_i \right)$ $(\tilde{U}', \tilde{V}') \leftarrow \left(\prod_{i \in P, i \neq t} \tilde{U}'_i, \prod_{i \in P, i \neq t} \tilde{V}'_i \right)$
Join decryption.....		
$\tilde{u} \leftarrow \mathcal{R}_4, \tilde{v} \leftarrow \mathcal{R}_5$ $K_j \leftarrow \prod_{i \in P} \mathcal{C}_{\text{enc}}^{r_i} \cdot Y_j \in P$ returns $\forall j \in P : \text{Vt} \left(\text{enc} \left(\frac{G}{V}, \frac{K_j}{\tilde{V}_j} \right), \kappa_{2,t} \right)$ returns $\forall j \in P : \text{Vt} \left(\text{enc} \left(\frac{G}{V}, \frac{K_j}{\tilde{V}'_j} \right), \kappa_{2,t'} \right)$ $\tilde{T} \leftarrow \prod_{i \in P, i \neq t} Y_i^{r_i}, \tilde{T}' \leftarrow \prod_{i \in P, i \neq t} Y_i^{r_i}$ if $(\tilde{V} \neq \tilde{T})$ then return ϵ $M \leftarrow \mathcal{C}_{\text{dec}}(\tilde{U}, \tilde{T}^{-1})$ return M		$\tilde{u} \leftarrow \mathcal{R}_4, \tilde{v} \leftarrow \mathcal{R}_5$ $K_i \leftarrow G^{\kappa_i}$ $\kappa_{2,t} \leftarrow \text{Prave} \left(\text{enc} \left(\frac{G}{V}, \frac{K_i}{\tilde{V}_i} \right), \kappa_{2,t} \right)$ $\kappa_{2,t'} \leftarrow \text{Prave} \left(\text{enc} \left(\frac{G}{V}, \frac{K_i}{\tilde{V}'_i} \right), \kappa_{2,t'} \right)$

Figure 4: Decryption Protocol (Procedures for fine-grained rate-limiting in Figure 8)

$$T_0 := \tilde{U}^{k_0}, T'_0 := \tilde{U}'^{k_0}$$

\mathcal{B} computes the views of all parties honestly except for the values U_0, V_0, T_0 and T'_0 . For the q -th query, \mathcal{B} sets $U_0 := G^\beta$ and $V_0 := G^Y \cdot G^{\beta k_0} \cdot Y_{0,0}^{-1}$. For other queries, \mathcal{B} computes U_0 and V_0 honestly. For all queries, to compute T_0 and T'_0 , \mathcal{B} runs the extractor of the NIZKPoK to extract the discrete logarithm \tilde{u} and \tilde{u}' such that $\tilde{U} = G^{\tilde{u}}$ and $\tilde{U}' = G^{\tilde{u}'}$. It then compute $T_0 := G^{\alpha \tilde{u}}$ and $T'_0 := G^{\alpha \tilde{u}'}$.

Clearly, if $(G, G^\alpha, G^\beta, G^Y)$ is a DH tuple, \mathcal{B} simulates $\text{Hyb}_{b,3,q-1}$ perfectly. Else, if $(G, G^\alpha, G^\beta, G^Y)$ is a random tuple, \mathcal{B} simulates $\text{Hyb}'_{b,3,q}$ perfectly. The claim then follows.



A Gap in BELSSZ'20

Dec(\dots)		Rate-Limiter $\mathcal{R}_i(\tau, \epsilon, \delta, \eta, \lambda, \forall i \in [m])$
Server $\mathcal{S}(1^{\lambda}, 1^{\tau}, \mathcal{M}_0, \text{pw}, \kappa, C)$ returns (U_0, V_0) $(G, C) \leftarrow \text{SKE}(\text{Dec}(\text{sk}_0, C))$ returns $(G, C, \kappa) \neq \perp$ $X_0 = H_0(\kappa), X_1 = H_1(\kappa)$ $X_{i,0} = G^{X_0} \cdot H_0(\text{pw}, \kappa)$		returns (U_0, V_0) $X_0 = H_0(\kappa), X_1 = H_1(\kappa)$ $X_{i,0} = X_{i,0}^*$
Computing encryption of $Z = \{z_i\}_{i \in P} \in \{0,1\}^{\tau}$ for some t -subset $P \subseteq [m]$		
$K = K_0 \cdot K_1$ $\kappa_0 = \text{KDF}(\text{sk}_0, \text{pw}) = (G^{\kappa_0}, K^{\kappa_0}, V_{K_0}^{\kappa_0})$ $S_j = \prod_{i \in P} z_i^{K_i^{\kappa_0}}, \forall j \in [m]$ $\kappa_{1,j} \leftarrow \text{Prave}(\text{pr}_1, (G, U_0, \kappa_0))$ $\mathcal{P} = \left\{ j \in [m] : \text{Ver} \left(\text{cm} \left(\frac{G}{K_0}, \frac{z_j}{K_0}, \frac{U_0}{V_0}, \kappa_{1,j} \right), \tau_j \right) \right\}$ returns $(\mathcal{P}) \geq t$ $P \leftarrow \text{Subset}(\mathcal{P})$		$K = K_0 \cdot K_1$ $\kappa_0 = \text{KDF}(\text{sk}_0, \text{pw}) = (G^{\kappa_0}, K^{\kappa_0}, V_{K_0}^{\kappa_0})$ $S_j = G^{\prod_{i \in P} z_i^{K_i^{\kappa_0}}}, \forall j \in [m] \setminus \{t\}$ $S_t = G^{\tau^t}$ $\kappa_{1,j} \leftarrow \text{Prave} \left(\text{cm} \left(\frac{G}{K_0}, \frac{z_j}{K_0}, \frac{U_0}{V_0}, \frac{U_0}{V_0} \right), \kappa_{1,j} \right)$
Computing encryption of $(Z^t, Z^{t'}) = (H_1(\text{pw}, \kappa), H_2(\kappa)) \in \{0,1\}^{\tau + \tau'}$ for some random t and t'		
$\tilde{\kappa}_0, \tilde{\kappa}_0' = \text{KDF}$ $(\tilde{U}_0, \tilde{V}_0) = (\text{cm}(\tilde{\kappa}_0, \tilde{V}_0), \text{dec}(\tilde{\kappa}_0, \tilde{V}_0)) = (G^{\tilde{\kappa}_0}, V_{K_0}^{\tilde{\kappa}_0}, H_1(\text{pw}, \kappa))$ $\kappa_{2,t} \leftarrow \text{Prave}(\text{pr}_2, (G, \tilde{U}_0, \tilde{\kappa}_0))$ $\kappa_{2,t'} \leftarrow \text{Prave}(\text{pr}_2, (G, \tilde{U}_0, \tilde{\kappa}_0'))$ $\kappa_{2,j} \leftarrow \text{Prave} \left(\text{cm} \left(\frac{G}{V_0}, \frac{z_j^{t'}}{G}, \frac{U_0}{V_0}, \kappa_{2,j} \right), \tau_j' \right)$ returns $\forall j \in P, \forall t \left(\text{cm} \left(\frac{G}{V_0}, \frac{z_j^t}{G}, \frac{U_0}{V_0}, \kappa_{2,j} \right) \right)$ returns $\forall j \in P, \forall t' \left(\text{cm} \left(\frac{G}{V_0}, \frac{z_j^{t'}}{G}, \frac{U_0}{V_0}, \kappa_{2,j} \right) \right)$ $(\tilde{U}, \tilde{V}) = \left(\prod_{i \in P, j \in [t]} U_0^i \cdot \prod_{i \in P, j \in [t']} U_0^{i'} \right)$ $(\tilde{U}', \tilde{V}') = \left(\prod_{i \in P, j \in [t]} U_0^i \cdot \prod_{i \in P, j \in [t']} U_0^{i'} \right)$ Joint Decryption $\tilde{\kappa}_0 = G^{\tilde{\kappa}_0}, \tilde{\kappa}_0' = G^{\tilde{\kappa}_0'}$ $K_j = \prod_{i \in P} z_i^{K_i^{\tilde{\kappa}_0}}, \forall j \in P$ $\kappa_{2,j} \leftarrow \text{Prave} \left(\text{cm} \left(\frac{G}{V_0}, \frac{z_j^t}{G}, \frac{U_0}{V_0}, \kappa_{2,j} \right) \right)$ $\kappa_{2,j} \leftarrow \text{Prave} \left(\text{cm} \left(\frac{G}{V_0}, \frac{z_j^{t'}}{G}, \frac{U_0}{V_0}, \kappa_{2,j} \right) \right)$ $T = \prod_{i \in P, j \in [t]} V_0^i \cdot \prod_{i \in P, j \in [t']} V_0^{i'}$ if $(\tilde{V} \neq T)$ then return \perp $M = G^{\tilde{V} / (T \cdot \tau^t)}$ return M		$\tilde{\kappa}_0, \tilde{\kappa}_0' = \text{KDF}$ $(\tilde{U}_0, \tilde{V}_0) = (\text{cm}(\tilde{\kappa}_0, \tilde{V}_0), \text{dec}(\tilde{\kappa}_0, \tilde{V}_0)) = (G^{\tilde{\kappa}_0}, V_{K_0}^{\tilde{\kappa}_0}, H_1(\text{pw}, \kappa))$ $\kappa_{2,t} \leftarrow \text{Prave}(\text{pr}_2, (G, \tilde{U}_0, \tilde{\kappa}_0))$ $\kappa_{2,t'} \leftarrow \text{Prave}(\text{pr}_2, (G, \tilde{U}_0, \tilde{\kappa}_0'))$ $\kappa_{2,j} \leftarrow \text{Prave} \left(\text{cm} \left(\frac{G}{V_0}, \frac{z_j^{t'}}{G}, \frac{U_0}{V_0}, \frac{U_0}{V_0} \right), \kappa_{2,j} \right)$ $\kappa_{2,j} \leftarrow \text{Prave} \left(\text{cm} \left(\frac{G}{V_0}, \frac{z_j^t}{G}, \frac{U_0}{V_0}, \frac{U_0}{V_0} \right), \kappa_{2,j} \right)$ $(\tilde{U}, \tilde{V}) = \left(\prod_{i \in P, j \in [t]} U_0^i \cdot \prod_{i \in P, j \in [t']} U_0^{i'} \right)$ $(\tilde{U}', \tilde{V}') = \left(\prod_{i \in P, j \in [t]} U_0^i \cdot \prod_{i \in P, j \in [t']} U_0^{i'} \right)$ $\tau_t = G^{\tau^t}, \tau_{t'} = G^{\tau^{t'}}$ $K_j = G^{K_j}$ $\kappa_{2,j} \leftarrow \text{Prave} \left(\text{cm} \left(\frac{G}{V_0}, \frac{K_j}{V_0}, \kappa_{2,j} \right) \right)$ $\kappa_{2,j} \leftarrow \text{Prave} \left(\text{cm} \left(\frac{G}{V_0}, \frac{K_j}{V_0}, \kappa_{2,j} \right) \right)$

Figure 4: Decryption Protocol (Procedures for fine-grained rate-limiting in Figure 8)

$$T_0 := \tilde{U}^{k_0}, T_0' := \tilde{U}'^{k_0}$$

\mathcal{B} computes the views of all parties honestly except for the values U_0, V_0, T_0 and T_0' . For the q -th query, \mathcal{B} sets $U_0 := G^{\beta}$ and $V_0 := G^Y \cdot G^{\beta k_0} \cdot Y_{0,0}^{-1}$. For other queries, \mathcal{B} computes U_0 and V_0 honestly. For all queries, to compute T_0 and T_0' , \mathcal{B} runs the extractor of the NIZKPoK to extract the discrete logarithm \tilde{u} and \tilde{u}' such that $\tilde{U} = G^{\tilde{u}}$ and $\tilde{U}' = G^{\tilde{u}'}$. It then compute $T_0 := G^{\alpha \tilde{u}}$ and $T_0' := G^{\alpha \tilde{u}'}$.

Clearly, if $(G, G^\alpha, G^\beta, G^Y)$ is a DH tuple, \mathcal{B} simulates $\text{Hyb}_{b,3,q-1}$ perfectly. Else, if $(G, G^\alpha, G^\beta, G^Y)$ is a \mathcal{R} tuple, \mathcal{B} simulates $\text{Hyb}'_{b,3,q}$ perfectly. The claim then follows.



A Gap in BELSSZ'20

Dec(\dots)	Rate-Limiter $\mathcal{R}_t(\text{rate}, \text{sk}_t, \text{pk}, \text{C})$	Rate-Limiter $\mathcal{R}_t(\text{rate}, \text{sk}_t, \text{pk}, \text{C})$
$\text{return } \mathcal{N}(\mathbb{G}, \text{sk}_t)$ $(\mathbb{G}, \text{C}) \leftarrow \text{SKE.Dec}(\text{sk}_t, \text{C})$ $\text{return } (\mathbb{G}, \text{C}, \text{C}) \neq \perp$ $X_0 \leftarrow \mathcal{H}_1(\text{sk}_t), X_1 \leftarrow \mathcal{H}_2(\text{sk}_t)$ $X_2 \leftarrow \mathcal{C}_2^{-1} \cdot \text{rk}_t(\text{pk}, \text{sk}_t)$	*	$X_0 \leftarrow \mathcal{H}_1(\text{sk}_t), X_1 \leftarrow \mathcal{H}_2(\text{sk}_t)$ $X_2 \leftarrow \mathcal{C}_2^{-1}$
Computing encryption of $Z = \mathcal{C}_2^{-1} \cdot \prod_{i \in P} v_i^{x_i}$ for some t -subset $P \subseteq [n]$...		
$K \leftarrow \mathcal{K}_t$ $\pi_1 \leftarrow \text{Exp}(\mathbb{G}, \text{vk}_t) = (G^{\pi_1}, K^{\pi_1}, V_{K_1}^{\pi_1})$ $S_j \leftarrow \prod_{i \in [n]} \mathcal{C}_1^{x_i} \cdot v_j \in [m]$ $\pi_{1,2} \leftarrow \text{Prove}(\text{com}, (\mathbb{G}, \mathbb{G}_1, \pi_1))$ $\mathcal{P} \leftarrow \left\{ i \in [n] : \text{Ver} \left(\text{com} \left(\frac{\mathbb{G}_1}{\mathbb{G}_1}, \frac{\mathbb{G}_1}{K}, \frac{\mathbb{G}_1}{V_j} \right), \pi_{1,2} \right) = 1 \right\}$ $\text{return } \mathcal{P} \geq t$ $\mathcal{P} \leftarrow \text{Subset}(\mathcal{P})$	$\mathbb{G}_1, V_0, \pi_{1,2}$	$K \leftarrow \mathcal{K}_t$ $\pi_1 \leftarrow \text{Exp}(\mathbb{G}, \text{vk}_t) = (G^{\pi_1}, K^{\pi_1}, V_{K_1}^{\pi_1})$ $S_j \leftarrow G^{\prod_{i \in [n]} x_i} \cdot v_j \in [m] \setminus \{t\}$ $\delta_1 \leftarrow G^{\pi_1}$ $\pi_{1,2} \leftarrow \text{Prove} \left(\text{com} \left(\frac{\mathbb{G}_1}{\mathbb{G}_1}, \frac{\mathbb{G}_1}{K}, \frac{\mathbb{G}_1}{V_j} \right), \left(\frac{\pi_1}{\delta_1} \right) \right)$
Computing encryption of $(Z^t, Z^{t'}) \cdot \mathcal{H}_1(\text{pk}, \pi) \cdot \mathcal{H}_2(\text{sk})^{\text{SKE.P} \setminus \{t, t'\}}$ for some random t and t' ...		
$(\mathbb{G}, \mathcal{V}) \leftarrow \left(\prod_{i \in P, i \neq t} \mathcal{C}_1^{x_i}, \prod_{i \in P, i \neq t'} \mathcal{C}_1^{x_i} \right)$ $\tilde{u}, \tilde{u}' \leftarrow \text{Exp}$ $(\mathbb{G}, \tilde{u}) \leftarrow (\mathcal{C}_1^{\tilde{u}}, V^{\tilde{u}}), (\mathbb{G}, \tilde{u}') \leftarrow (\mathcal{C}_1^{\tilde{u}'}, V^{\tilde{u}'}) = (G^{\tilde{u}}, V^{\tilde{u}}, X_0^{\tilde{u}}, X_1^{\tilde{u}}, X_2^{\tilde{u}})$ $\pi_{2,2} \leftarrow \text{Prove}(\text{com}, (\mathbb{G}, \mathbb{G}_1, \tilde{u}))$ $\pi_{2,2}' \leftarrow \text{Prove}(\text{com}, (\mathbb{G}, \mathbb{G}_1, \tilde{u}'))$	$\mathbb{G}_1, \tilde{u}, \pi_{2,2}, \tilde{u}', \pi_{2,2}'$	$\text{return } \mathcal{V} \in \mathcal{P} : \text{Ver} \left(\text{com} \left(\frac{\mathbb{G}_1}{\mathbb{G}_1}, \frac{\mathbb{G}_1}{V_j} \right), \pi_{2,2} \right)$ $(\mathbb{G}, \mathcal{V}) \leftarrow \left(\prod_{i \in P, i \neq t} \mathcal{C}_1^{x_i}, \prod_{i \in P, i \neq t'} \mathcal{C}_1^{x_i} \right)$ $\tilde{u}, \tilde{u}' \leftarrow \text{Exp}$ $(\mathbb{G}, \tilde{u}) \leftarrow (\mathcal{C}_1^{\tilde{u}}, V^{\tilde{u}}), (\mathbb{G}, \tilde{u}') \leftarrow (\mathcal{C}_1^{\tilde{u}'}, V^{\tilde{u}'}) = (G^{\tilde{u}}, V^{\tilde{u}}, X_0^{\tilde{u}}, X_1^{\tilde{u}}, X_2^{\tilde{u}})$ $\pi_{2,2} \leftarrow \text{Prove} \left(\text{com} \left(\frac{\mathbb{G}_1}{\mathbb{G}_1}, \frac{\mathbb{G}_1}{V_j}, X_0^{\tilde{u}}, X_1^{\tilde{u}}, X_2^{\tilde{u}} \right), \left(\frac{\tilde{u}}{\tilde{u}'} \right) \right)$
Join decryption		
$\tilde{u} \leftarrow \mathcal{C}_1^{-1} \cdot \tilde{u}' = \tilde{u}$ $K_j \leftarrow \prod_{i \in [n]} \mathcal{C}_1^{x_i} \cdot v_j \in \mathcal{P}$	$\tilde{u}, \pi_{2,2}, \tilde{u}', \pi_{2,2}'$	$\pi_{2,2} \leftarrow \text{Prove} \left(\text{com} \left(\frac{\mathbb{G}_1}{\mathbb{G}_1}, \frac{\mathbb{G}_1}{K_j} \right), \tilde{u} \right)$ $\pi_{2,2}' \leftarrow \text{Prove} \left(\text{com} \left(\frac{\mathbb{G}_1}{\mathbb{G}_1}, \frac{\mathbb{G}_1}{K_j} \right), \tilde{u}' \right)$
$\text{return } \mathcal{V} \in \mathcal{P} : \text{Ver} \left(\text{com} \left(\frac{\mathbb{G}_1}{\mathbb{G}_1}, \frac{\mathbb{G}_1}{V_j} \right), \pi_{2,2} \right)$ $\text{return } \mathcal{V} \in \mathcal{P} : \text{Ver} \left(\text{com} \left(\frac{\mathbb{G}_1}{\mathbb{G}_1}, \frac{\mathbb{G}_1}{V_j} \right), \pi_{2,2}' \right)$ $\mathcal{T} \leftarrow \prod_{i \in P, i \neq t} \mathcal{C}_1^{x_i}, \mathcal{T}' \leftarrow \prod_{i \in P, i \neq t'} \mathcal{C}_1^{x_i}$ $\text{if } (\tilde{\mathcal{P}} \neq \mathcal{T}) \text{ then return } \epsilon$ $M \leftarrow \mathcal{C}_2^{-1}(\tilde{u}^t \cdot \mathcal{T}^{-t})$ $\text{return } M$		$\pi_{2,2} \leftarrow \text{Prove} \left(\text{com} \left(\frac{\mathbb{G}_1}{\mathbb{G}_1}, \frac{\mathbb{G}_1}{K_j} \right), \tilde{u} \right)$ $\pi_{2,2}' \leftarrow \text{Prove} \left(\text{com} \left(\frac{\mathbb{G}_1}{\mathbb{G}_1}, \frac{\mathbb{G}_1}{K_j} \right), \tilde{u}' \right)$ $\text{return } \epsilon$

Figure 4: Decryption Protocol (Procedures for fine-grained rate-limiting in Figure 8)

$$T_0 := \tilde{U}^{k_0}, T'_0 := \tilde{U}'^{k_0}$$

\mathcal{B} computes the views of all parties honestly except for the values U_0, V_0, T_0 and T'_0 . For the q -th query, \mathcal{B} sets $U_0 := G^\beta$ and $V_0 := G^Y \cdot G^{\beta k_0} \cdot Y_{0,0}^{-1}$. For other queries, \mathcal{B} computes U_0 and V_0 honestly. For all queries, to compute T_0 and T'_0 , \mathcal{B} runs the extractor of the NIZKPoK to extract the discrete logarithm \tilde{u} and \tilde{u}' such that $\tilde{U} = G^{\tilde{u}}$ and $\tilde{U}' = G^{\tilde{u}'}$. It then compute $T_0 := G^{\alpha \tilde{u}}$ and $T'_0 := G^{\alpha \tilde{u}'}$.

Clearly, if $(G, G^\alpha, G^\beta, G^Y)$ is a DH tuple, \mathcal{B} simulates $\text{Hyb}_{b,3,q-1}$ perfectly. Else, if $(G, G^\alpha, G^\beta, G^Y)$ is a random tuple, \mathcal{B} simulates $\text{Hyb}'_{b,3,q}$ perfectly. The claim then follows.

- $\text{Hyb}_{b,1}$ is mostly identical to $\text{Hyb}_{b,0}$, except that all zero-knowledge proofs are simulated by running the simulator of the NIZKPoK scheme. It is straightforward to show that, for all $b \in \{0, 1\}$,

$$\left| \Pr[\text{Hyb}_{b,0} = 1] - \Pr[\text{Hyb}_{b,1} = 1] \right| \leq \text{negl}(\lambda)$$

using the zero-knowledge property of the NIZKPoK scheme.

A Gap in BELSSZ'20

Dec(\dots)		Rate-Limiter $\mathcal{R}_t(\text{rate}, \text{sk}_t, \forall t \in [m])$
Server $\mathcal{S}(1^{\lambda}, 1^{\epsilon}, \text{pk}, \text{pw}, \kappa, C)$ returns (U, V, π) $(G, C) \leftarrow \text{SKE.Dec}(\text{sk}, C)$ outputs $(G, C, \kappa) \neq \perp$ $X_0 \leftarrow H_0(\kappa), X_1 \leftarrow H_1(\kappa)$ $X_{\text{enc}} \leftarrow G^{\kappa} \cdot H_0(\text{pw}, \kappa)$		returns (U, V, π) $X_0 \leftarrow H_0(\kappa), X_1 \leftarrow H_1(\kappa)$ $X_{\text{enc}} \leftarrow G^{\kappa}$
Computing encryption of $Z = X_{\text{enc}}^{-1} \cdot \prod_{i \in P} V_i^{x_i}$ for some t -subset $P \subseteq [m]$		
$K \leftarrow K_0 \cdot K_1$ $\kappa_0 \leftarrow \text{Exp}_G(U_0, V_0) \in (G^{\kappa}, K^{\kappa}, V_{\text{enc}}^{\kappa})$ $S_j \leftarrow \prod_{i \in P} V_i^{x_i}, \forall j \in [m]$ $\pi_{1,j} \leftarrow \text{Proof}(\text{enc}, (G, U_0, \kappa))$ $\mathcal{P} \leftarrow \left\{ j \in [m] : \text{Ver} \left(\text{enc} \left(\frac{G}{K_0}, \frac{G}{K}, \frac{G}{V_j}, \pi_{1,j} \right) \right) = 1 \right\}$ outputs $(\mathcal{P}) \geq t$ $\mathcal{P} \leftarrow \text{Subset}(\mathcal{P})$		$K \leftarrow K_0 \cdot K_1$ $\kappa_0 \leftarrow \text{Exp}_G(U_0, V_0) \in (G^{\kappa}, K^{\kappa}, V_{\text{enc}}^{\kappa})$ $S_j \leftarrow G^{\text{Exp}_G(U_j, V_j)}, \forall j \in [m] \setminus \{t\}$ $S_t \leftarrow G^{\kappa}$ $\pi_{1,t} \leftarrow \text{Proof} \left(\text{enc} \left(\frac{G}{K_0}, \frac{G}{K}, \frac{G}{V_t}, \pi_{1,t} \right) \right)$ outputs $\forall j \in P : (j) : \text{Ver} \left(\text{enc} \left(\frac{G}{K_0}, \frac{G}{K}, \frac{G}{V_j}, \pi_{1,j} \right) \right)$ outputs $\forall j \in [m] \setminus P : (j)$ $(U, V) \leftarrow \left(\prod_{i \in P} U_i^{x_i}, \prod_{i \in P} V_i^{x_i} \right)$
Computing encryption of $(Z^t, Z^{t'}) \cdot H_1(\text{pw}, \kappa) \cdot H_2(\kappa)^{\text{Exp}_G(V_{\text{enc}}^{x_{\text{enc}}})}$ for some random t and t'		
$\tilde{\kappa}_0, \tilde{\kappa}_1 \leftarrow \text{Exp}_G$ $(\tilde{U}_0, \tilde{V}_0) \leftarrow (\text{enc}, \tilde{U}_0^{\tilde{\kappa}_0}, \tilde{V}_0^{\tilde{\kappa}_0}) \in (G^{\tilde{\kappa}_0}, V_{\text{enc}}^{\tilde{\kappa}_0}, H_1(\text{pw}, \kappa))$ $\pi_{2,t} \leftarrow \text{Proof}(\text{enc}, (G, \tilde{U}_0, \tilde{\kappa}_0))$ $\pi_{2,t'} \leftarrow \text{Proof}(\text{enc}, (G, \tilde{U}_0, \tilde{\kappa}_0))$		$\tilde{\kappa}_0, \tilde{\kappa}_1 \leftarrow \text{Exp}_G$ $(\tilde{U}_0, \tilde{V}_0) \leftarrow (\text{enc}, \tilde{U}_0^{\tilde{\kappa}_0}, \tilde{V}_0^{\tilde{\kappa}_0}) \in (G^{\tilde{\kappa}_0}, V_{\text{enc}}^{\tilde{\kappa}_0}, H_1(\text{pw}, \kappa))$ $\pi_{2,t} \leftarrow \text{Proof}(\text{enc}, (G, \tilde{U}_0, \tilde{\kappa}_0))$ $\pi_{2,t'} \leftarrow \text{Proof} \left(\text{enc} \left(\frac{G}{V_t}, X_{\text{enc}}^{t'} \cdot \frac{G}{G}, \frac{G}{V_{t'}} \right) \left(\frac{G}{G} \right) \right)$
outputs $\forall j \in P : \text{Ver} \left(\text{enc} \left(\frac{G}{V_j}, \frac{G}{V_j}, \pi_{2,j} \right) \right)$ outputs $\forall j \in P : \text{Ver} \left(\text{enc} \left(\frac{G}{V_j}, X_{\text{enc}}^{t'} \cdot \frac{G}{G}, \frac{G}{V_j}, \pi_{2,j} \right) \right)$		outputs $\forall j \in P : (j) : \text{Ver} \left(\text{enc} \left(\frac{G}{V_j}, \frac{G}{V_j}, \pi_{2,j} \right) \right)$ outputs $\forall j \in P : (j) : \text{Ver} \left(\text{enc} \left(\frac{G}{V_j}, X_{\text{enc}}^{t'} \cdot \frac{G}{G}, \frac{G}{V_j}, \pi_{2,j} \right) \right)$
Join decryption		
$\tilde{\kappa}_0 \leftarrow \text{Exp}_G(U_0, V_0) \in (G^{\tilde{\kappa}_0}, K^{\tilde{\kappa}_0}, V_{\text{enc}}^{\tilde{\kappa}_0})$ $K_2 \leftarrow \prod_{i \in P} V_i^{x_i}, \forall i \in P$ outputs $\forall j \in P : \text{Ver} \left(\text{enc} \left(\frac{G}{V_j}, \frac{G}{V_j}, \pi_{2,j} \right) \right)$ outputs $\forall j \in P : \text{Ver} \left(\text{enc} \left(\frac{G}{V_j}, X_{\text{enc}}^{t'} \cdot \frac{G}{G}, \frac{G}{V_j}, \pi_{2,j} \right) \right)$ $T \leftarrow \prod_{i \in P} U_i^{x_i}, T' \leftarrow \prod_{i \in P} V_i^{x_i}$ if $(\tilde{\mathcal{P}} \neq \mathcal{P})$ then return ϵ $M \leftarrow G^{\text{Exp}_G(U, V \cdot T^{-1})}$ return M		$\tilde{\kappa}_0 \leftarrow \text{Exp}_G, \tilde{\kappa}_1 \leftarrow \text{Exp}_G$ $K_2 \leftarrow G^{\kappa}$ $\pi_{2,t} \leftarrow \text{Proof} \left(\text{enc} \left(\frac{G}{V_t}, \frac{G}{V_t}, \pi_{2,t} \right) \right)$ $\pi_{2,t'} \leftarrow \text{Proof} \left(\text{enc} \left(\frac{G}{V_{t'}}, X_{\text{enc}}^{t'} \cdot \frac{G}{G}, \frac{G}{V_{t'}}, \pi_{2,t'} \right) \right)$ $T_1, \pi_{2,t}, \pi_{2,t'}$ returns ϵ

Figure 4: Decryption Protocol (Procedures for fine-grained rate-limiting in Figure 8)

$$T_0 := \tilde{U}^{k_0}, T'_0 := \tilde{U}'^{k_0}$$

\mathcal{B} computes the views of all parties honestly except for the values U_0, V_0, T_0 and T'_0 . For the q -th query, \mathcal{B} sets $U_0 := G^\beta$ and $V_0 := G^Y \cdot G^{\beta k_0} \cdot Y_{0,0}^{-1}$. For other queries, \mathcal{B} computes U_0 and V_0 honestly. For all queries, to compute T_0 and T'_0 , \mathcal{B} runs the extractor of the NIZKPoK to extract the discrete logarithm \tilde{u} and \tilde{u}' such that $\tilde{U} = G^{\tilde{u}}$ and $\tilde{U}' = G^{\tilde{u}'}$. It then compute $T_0 := G^{\tilde{u}k_0}$ and $T'_0 := G^{\tilde{u}'k_0}$.

Clearly, if $(G, G^\alpha, G^\beta, G^Y)$ is a DH tuple, \mathcal{B} simulates $\text{Hyb}_{b,3,q-1}$ perfectly. Else, if $(G, G^\alpha, G^\beta, G^Y)$ is a random tuple, \mathcal{B} simulates $\text{Hyb}'_{b,3,q}$ perfectly. The claim then follows.

- $\text{Hyb}_{b,1}$ is mostly identical to $\text{Hyb}_{b,0}$, except that all zero-knowledge proofs are simulated by running the simulator of the NIZKPoK scheme. It is straightforward to show that, for all $b \in \{0, 1\}$,

$$\left| \Pr[\text{Hyb}_{b,0} = 1] - \Pr[\text{Hyb}_{b,1} = 1] \right| \leq \text{negl}(\lambda)$$

using the zero-knowledge property of the NIZKPoK scheme.



A Gap in BELSSZ'20

Dec(, ...,)	Rate-Limiter $R_{i,j} := (r_{i,j}^c, s_{i,j}, \forall i \in [m])$
Server $S(1^{\lambda}, 1^{\epsilon}, \rho_{pub}, pw, s, C)$ returns (U_0, V_0) $(C_0, C) \leftarrow \text{SKE}(\text{Dec}(s_0, C))$ returns $(C_0, C) \neq \perp$ $X_0 = H_0(s), X_1 = H_1(s)$ $Y_{i,j}^0 = G^{s_{i,j}} \cdot H_0(pw, s)$	returns (U_0^c, V_0^c) $X_0 = H_0(s), X_1 = H_1(s)$ $Y_{i,j} = X_{i,j}^0$
Computing encryption of $Z = Y_{i,j}^0 \cdot \prod_{(i,j) \in P} Y_{i,j}^{0,c}$ for some t -subset $P \subseteq [m]$	
$K = K_0 \cdot K_1$ $s_1 = s \cdot \alpha_p, (U_0, V_0) = (G^{s_1}, K^s \cdot Y_{i,j}^0)$ $S_j = \prod_{(i,j) \in P} G^{s_{i,j}^c}, \forall j \in [m]$ $\pi_{1,2} \leftarrow \text{Prove}(\text{com}, (G, U_0, s_1))$ $P = \{j \in [m] : \text{Ver}(\text{com}, (G, U_0, \frac{U_0}{K}, \frac{U_0}{V_0}, \pi_{1,2})) = 1\}$ returns $(P) \geq t$ $P \leftarrow \text{Subset}(P)$ $(U, V) = \left(\prod_{(i,j) \in P} U_{i,j}^{0,c}, \prod_{(i,j) \in P} V_{i,j}^{0,c} \right)$	$K = K_0 \cdot K_1$ $r_1 = s \cdot \alpha_p, (U_0, V_0) = (G^{r_1}, K^s \cdot Y_{i,j})$ $S_j = G^{\prod_{(i,j) \in P} s_{i,j}^c}, \forall j \in [m] \setminus \{1\}$ $S_1 = G^{s^c}$ $\pi_{1,2} \leftarrow \text{Prove}(\text{com}, (G, U_0, \frac{U_0}{K}, \frac{U_0}{V_0}, \pi_{1,2}))$ returns $\forall j \in P : (1) : \text{Ver}(\text{com}, (G, U_0, \frac{U_0}{K}, \frac{U_0}{V_0}, \pi_{1,2}))$ returns $\forall j \notin P : (G, U_0, s_1)$ $(U, V) = \left(\prod_{(i,j) \in P} U_{i,j}^{0,c}, \prod_{(i,j) \in P} V_{i,j}^{0,c} \right)$
Computing encryption of $(Z^t, Z^{t'}) = H_1(pw, s) \cdot H_2(s) \cdot \prod_{(i,j) \in P} Y_{i,j}^{0,c}$ for some random t and t'	
$\tilde{r}_1, \tilde{r}_2 = s \cdot \alpha_p$ $(U, V) = (G^{\tilde{r}_1}, G^{\tilde{r}_2} \cdot H_1(pw, s) \cdot H_2(s) \cdot \prod_{(i,j) \in P} Y_{i,j}^{0,c})$	$\tilde{r}_1, \tilde{r}_2 = s \cdot \alpha_p$ $(U, V) = (G^{\tilde{r}_1}, G^{\tilde{r}_2} \cdot H_1(pw, s) \cdot H_2(s) \cdot \prod_{(i,j) \in P} Y_{i,j}^{0,c})$

$$T_0 := \tilde{U}^{k_0}, T_0' := \tilde{U}'^{k_0}$$

\mathcal{B} computes the views of all parties honestly except for the values U_0, V_0, T_0 and T_0' . For the q -th query, \mathcal{B} sets $U_0 := G^{\beta}$ and $V_0 := G^{\gamma} \cdot G^{\beta k_0} \cdot Y_{0,0}^{-1}$. For other queries, \mathcal{B} computes U_0 and V_0 honestly. For all queries, to compute T_0 and T_0' , \mathcal{B} runs the extractor of the NIZKPoK to extract the discrete logarithm \tilde{u} and \tilde{u}' such that $\tilde{U} = G^{\tilde{u}}$ and $\tilde{U}' = G^{\tilde{u}'}$. It then compute $T_0 := G^{\alpha \tilde{u}}$ and $T_0' := G^{\alpha \tilde{u}'}$.

Extracting from simulated proofs is impossible!

- Hyb_{b,1} is mostly identical to Hyb_{b,0}, except that all zero-knowledge proofs are simulated by running the simulator of the NIZKPoK scheme. It is straightforward to show that, for all $b \in \{0, 1\}$,

$$\left| \Pr[\text{Hyb}_{b,0} = 1] - \Pr[\text{Hyb}_{b,1} = 1] \right| \leq \text{negl}(\lambda)$$

using the zero-knowledge property of the NIZKPoK scheme.

Figure 4: Decryption Protocol (Procedures for fine-grained rate-limiting in Figure 8)



Thanks for your attention!



ruben-baecker.de

Password-Hardened Encryption Revisited

Ruben Baecker, Paul Gerhart, and Dominique Schröder

ia.cr/2025/1453

Universally Composable Password-Hardened
Encryption

Behzad Abdolmaleki, Ruben Baecker, Paul Gerhart, Mike
Graf, Mojtaba Khalili, Daniel Rausch, and Dominique
Schröder

ia.cr/2025/1647

UCPY: A Hybrid Protocol

