

(Fine-Grained) Unbounded Inner-Product Functional Encryption from LWE

Valerio Cini

Erkan Tairi

Bocconi



SCN 2026

(Inner-Product) Functional Encryption [Abd+15]

Public-key encryption with **fine-grained access to computation**: a functional key sk_f reveals only $f(x)$ from a ciphertext of x — and nothing else.

- $\text{Setup}(1^\lambda) \rightarrow (\text{mpk}, \text{msk})$ master keys
- $\text{KeyGen}(\text{msk}, f) \rightarrow sk_f$ key for a function f
- $\text{Enc}(\text{mpk}, x) \rightarrow ct_x$ encrypt a message x
- $\text{Dec}(sk_f, ct_x) \rightarrow f(x)$ learn only $f(x)$

(Inner-Product) Functional Encryption [Abd+15]

Public-key encryption with **fine-grained access to computation**: a functional key sk_f reveals only $f(x)$ from a ciphertext of x — and nothing else.

- $\text{Setup}(1^\lambda) \rightarrow (\text{mpk}, \text{msk})$ master keys
- $\text{KeyGen}(\text{msk}, f) \rightarrow sk_f$ key for a function f
- $\text{Enc}(\text{mpk}, x) \rightarrow ct_x$ encrypt a message x
- $\text{Dec}(sk_f, ct_x) \rightarrow f(x)$ learn only $f(x)$

Inner-product FE (IPFE). keys and messages are vectors; decryption reveals only $\langle \mathbf{x}, \mathbf{y} \rangle$.

(Inner-Product) Functional Encryption [Abd+15]

Public-key encryption with **fine-grained access to computation**: a functional key sk_f reveals only $f(x)$ from a ciphertext of x — and nothing else.

- $\text{Setup}(1^\lambda) \rightarrow (\text{mpk}, \text{msk})$ master keys
- $\text{KeyGen}(\text{msk}, f) \rightarrow sk_f$ key for a function f
- $\text{Enc}(\text{mpk}, x) \rightarrow ct_x$ encrypt a message x
- $\text{Dec}(sk_f, ct_x) \rightarrow f(x)$ learn only $f(x)$

Inner-product FE (IPFE). keys and messages are vectors; decryption reveals only $\langle \mathbf{x}, \mathbf{y} \rangle$.
Applications: privacy-preserving ML [Mar+19], money-laundering detection [PP22], search on encrypted streams [BCS23], data marketplaces [Kou+20], ...

From Bounded to Unbounded IPFE

Problem: almost all IPFE schemes fix the vector length ℓ at setup — so mpk and msk depend on ℓ .

From Bounded to Unbounded IPFE

Problem: almost all IPFE schemes **fix the vector length ℓ at setup** — so mpk and msk depend on ℓ .

Unbounded IPFE (UIPFE) [TT18; DP19]: no a-priori bound on ℓ — keys and ciphertexts for *any* length.

From Bounded to Unbounded IPFE

Problem: almost all IPFE schemes **fix the vector length ℓ at setup** — so mpk and msk depend on ℓ .

Unbounded IPFE (UIPFE) [TT18; DP19]: no a-priori bound on ℓ — keys and ciphertexts for *any* length.

For a key of length ℓ' and a ciphertext of length ℓ :

Strict

decrypt iff $\ell' = \ell$,
recover $\langle \mathbf{x}, \mathbf{y} \rangle$.

Permissive

decrypt iff $\ell' \leq \ell$,
recover $\langle \mathbf{x}|_{[\ell']}, \mathbf{y} \rangle$.

From Bounded to Unbounded IPFE

Problem: almost all IPFE schemes **fix the vector length ℓ at setup** — so mpk and msk depend on ℓ .

Unbounded IPFE (UIPFE) [TT18; DP19]: no a-priori bound on ℓ — keys and ciphertexts for *any* length.

For a key of length ℓ' and a ciphertext of length ℓ :

Strict

decrypt iff $\ell' = \ell$,
recover $\langle \mathbf{x}, \mathbf{y} \rangle$.

Permissive

decrypt iff $\ell' \leq \ell$,
recover $\langle \mathbf{x}|_{[\ell']}, \mathbf{y} \rangle$.

Fine-grained: add identity-/attribute-based access control — decrypt only if the policy is satisfied, $C(w) = 0$.

This work: lattice-based (fine-grained) UIPFE

Prior UIPFE is **pairing-based** [TT18; DP19; DP23] — not post-quantum. The only lattice route [GKW16] is **non-black-box** and only semi-adaptive.

Main result

The first UIPFE from **plain LWE** that is
standard-model, adaptive, black-box, and fine-grained (IB & AB).

This work: lattice-based (fine-grained) UIPFE

Prior UIPFE is **pairing-based** [TT18; DP19; DP23] — not post-quantum. The only lattice route [GKW16] is **non-black-box** and only semi-adaptive.

Main result

The first UIPFE from **plain LWE** that is
standard-model, adaptive, black-box, and fine-grained (IB & AB).

Four schemes, all reducing to the (bounded) ALS IPFE [ALS16]:

- strict UIPFE • permissive UIPFE
- identity-based UIPFE • attribute-based UIPFE

State of the art

Scheme	Fine-grained	Security	Assumption	Model	Black-box
[GKW16] [*]	IB/AB	semi-adaptive	IPFE + OWF	SM	✗
[TT18]	✗	adaptive	SXDH	SM	✓
[DP19]	IB	selective	DBDH	ROM	✓
[DP23] [†]	IB/AB	selective	DBDH	ROM	✓
This work	IB/AB	adaptive [‡]	LWE	SM	✓

^{*} generic, strict only. [†] LSSS policies only. [‡] our AB-UIPFE is selective, depth- d circuits.

Security: adaptive IND-CPA

- Adversary gets mpk and **adaptive** access to $\text{KeyGen}(\text{msk}, \cdot)$.
- Outputs a challenge $(\mathbf{x}_0^*, \mathbf{x}_1^*)$; receives $\text{ct}^* \leftarrow \text{Enc}(\text{mpk}, \mathbf{x}_b^*)$ for a hidden bit b .
- Wins if it guesses b .

Validity of the adversary

every queried key $\mathbf{y}$ satisfies $\langle \mathbf{x}_0^*, \mathbf{y} \rangle = \langle \mathbf{x}_1^*, \mathbf{y} \rangle$
(and, in the fine-grained setting, is not authorized to decrypt the challenge).

Security: adaptive IND-CPA

- Adversary gets mpk and **adaptive** access to $\text{KeyGen}(\text{msk}, \cdot)$.
- Outputs a challenge $(\mathbf{x}_0^*, \mathbf{x}_1^*)$; receives $\text{ct}^* \leftarrow \text{Enc}(\text{mpk}, \mathbf{x}_b^*)$ for a hidden bit b .
- Wins if it guesses b .

Validity of the adversary

every queried key $\mathbf{y}$ satisfies $\langle \mathbf{x}_0^*, \mathbf{y} \rangle = \langle \mathbf{x}_1^*, \mathbf{y} \rangle$
(and, in the fine-grained setting, is not authorized to decrypt the challenge).

Selective / semi-adaptive: commit to $(\mathbf{x}_0^*, \mathbf{x}_1^*)$ before mpk / before oracle queries.

Starting point: the ALS bounded IPFE [ALS16]

ALS[ℓ] (error terms omitted)

Setup($1^\lambda, 1^\ell$): $\mathbf{A} \leftarrow \$ \mathbb{Z}_q^{n \times m}$, $\mathbf{T} \leftarrow \$ \mathcal{D}_{\mathbb{Z}^{m \times \ell}, \tau}$, $\mathbf{D} := \mathbf{A}\mathbf{T}$
mpk = $(\mathbf{A}, \mathbf{D})$, msk = $\mathbf{T}$

Enc($\mathbf{x}$): $\mathbf{c}_0 := \mathbf{s}\mathbf{A}$, $\mathbf{c}_1 := \mathbf{s}\mathbf{D} + \lfloor q/K \rfloor \cdot \mathbf{x}$

KeyGen($\mathbf{y}$): $\mathbf{t}_y^\top := \mathbf{T}\mathbf{y}^\top$

Dec: $\mathbf{c}_1\mathbf{y}^\top - \mathbf{c}_0\mathbf{t}_y^\top \approx \lfloor q/K \rfloor \cdot \langle \mathbf{x}, \mathbf{y} \rangle$

Starting point: the ALS bounded IPFE [ALS16]

ALS[ℓ] (error terms omitted)

Setup($1^\lambda, 1^\ell$): $\mathbf{A} \leftarrow \$ \mathbb{Z}_q^{n \times m}$, $\mathbf{T} \leftarrow \$ \mathcal{D}_{\mathbb{Z}^{m \times \ell}, \tau}$, $\mathbf{D} := \mathbf{A}\mathbf{T}$

mpk = $(\mathbf{A}, \mathbf{D})$, msk = $\mathbf{T}$

Enc($\mathbf{x}$): $\mathbf{c}_0 := \mathbf{s}\mathbf{A}$, $\mathbf{c}_1 := \mathbf{s}\mathbf{D} + \lfloor q/K \rfloor \cdot \mathbf{x}$

KeyGen($\mathbf{y}$): $\mathbf{t}_y^\top := \mathbf{T}\mathbf{y}^\top$

Dec: $\mathbf{c}_1\mathbf{y}^\top - \mathbf{c}_0\mathbf{t}_y^\top \approx \lfloor q/K \rfloor \cdot \langle \mathbf{x}, \mathbf{y} \rangle$

Obstacle: both $\mathbf{D} \in \mathbb{Z}_q^{n \times \ell}$ and $\mathbf{T} \in \mathbb{Z}^{m \times \ell}$ depend on ℓ .

Starting point: the ALS bounded IPFE [ALS16]

ALS[ℓ] (error terms omitted)

Setup($1^\lambda, 1^\ell$): $\mathbf{A} \leftarrow \$ \mathbb{Z}_q^{n \times m}$, $\mathbf{T} \leftarrow \$ \mathcal{D}_{\mathbb{Z}^{m \times \ell}, \tau}$, $\mathbf{D} := \mathbf{A}\mathbf{T}$

mpk = $(\mathbf{A}, \mathbf{D})$, msk = $\mathbf{T}$

Enc($\mathbf{x}$): $\mathbf{c}_0 := \mathbf{s}\mathbf{A}$, $\mathbf{c}_1 := \mathbf{s}\mathbf{D} + \lfloor q/K \rfloor \cdot \mathbf{x}$

KeyGen($\mathbf{y}$): $\mathbf{t}_y^\top := \mathbf{T}\mathbf{y}^\top$

Dec: $\mathbf{c}_1\mathbf{y}^\top - \mathbf{c}_0\mathbf{t}_y^\top \approx \lfloor q/K \rfloor \cdot \langle \mathbf{x}, \mathbf{y} \rangle$

Obstacle: both $\mathbf{D} \in \mathbb{Z}_q^{n \times \ell}$ and $\mathbf{T} \in \mathbb{Z}^{m \times \ell}$ depend on ℓ .

(All other parameters can be made independent of ℓ [ALS16].)

Step 1: remove the msk dependence on ℓ

Idea: sample $\mathbf{A}$ with a trapdoor $\mathbf{T}_\mathbf{A}$, sample $\mathbf{D} \leftarrow \mathbb{Z}_q^{n \times \ell}$ uniformly, and derive $\mathbf{T}$ *on the fly* as a Gaussian preimage of $\mathbf{D}$ under $\mathbf{A}$.

modified algorithms

Setup: $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \text{TrapGen}$, $\mathbf{D} \leftarrow \mathbb{Z}_q^{n \times \ell}$, $k \leftarrow \{0, 1\}^\lambda$

$\text{mpk} = (\mathbf{A}, \mathbf{D})$, $\text{msk} = (\mathbf{T}_\mathbf{A}, k)$

KeyGen($\mathbf{y}$): $\mathbf{T} \leftarrow \text{SamplePre}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{D}, \tau; \text{PRF}(k, \ell))$, $\mathbf{t}_\mathbf{y}^\top := \mathbf{T}\mathbf{y}^\top$

Step 1: remove the msk dependence on ℓ

Idea: sample $\mathbf{A}$ with a trapdoor $\mathbf{T}_\mathbf{A}$, sample $\mathbf{D} \leftarrow \$ \mathbb{Z}_q^{n \times \ell}$ uniformly, and derive $\mathbf{T}$ *on the fly* as a Gaussian preimage of $\mathbf{D}$ under $\mathbf{A}$.

modified algorithms

Setup: $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \text{TrapGen}$, $\mathbf{D} \leftarrow \$ \mathbb{Z}_q^{n \times \ell}$, $k \leftarrow \$ \{0, 1\}^\lambda$

$\text{mpk} = (\mathbf{A}, \mathbf{D})$, $\text{msk} = (\mathbf{T}_\mathbf{A}, k)$

KeyGen($\mathbf{y}$): $\mathbf{T} \leftarrow \text{SamplePre}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{D}, \tau; \text{PRF}(k, \ell))$, $\mathbf{t}_\mathbf{y}^\top := \mathbf{T}\mathbf{y}^\top$

- the PRF *derandomizes* preimage sampling — KeyGen stays deterministic (crucial for security).

Step 1: remove the msk dependence on ℓ

Idea: sample $\mathbf{A}$ with a trapdoor $\mathbf{T}_\mathbf{A}$, sample $\mathbf{D} \leftarrow \mathbb{Z}_q^{n \times \ell}$ uniformly, and derive $\mathbf{T}$ *on the fly* as a Gaussian preimage of $\mathbf{D}$ under $\mathbf{A}$.

modified algorithms

Setup: $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \text{TrapGen}$, $\mathbf{D} \leftarrow \mathbb{Z}_q^{n \times \ell}$, $k \leftarrow \{0, 1\}^\lambda$

$\text{mpk} = (\mathbf{A}, \mathbf{D})$, $\text{msk} = (\mathbf{T}_\mathbf{A}, k)$

KeyGen($\mathbf{y}$): $\mathbf{T} \leftarrow \text{SamplePre}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{D}, \tau; \text{PRF}(k, \ell))$, $\mathbf{t}_\mathbf{y}^\top := \mathbf{T}\mathbf{y}^\top$

- the PRF derandomizes preimage sampling — KeyGen stays deterministic (crucial for security).

Observation

the only remaining dependence on ℓ is now $\mathbf{D} \in \mathbb{Z}_q^{n \times \ell}$.

Step 2: “delay” the sampling of $\mathbf{D}$ to KeyGen [CW24]

Add $\mathbf{B}, \mathbf{W}, \mathbf{Y} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{v} \in \mathbb{Z}_q^n$ to mpk, and $\mathbf{T}_\mathbf{B}$ to msk.

- Ciphertext carries, for $j \in [\ell]$:

$$\mathbf{s}_j \mathbf{v}^\top + x_j \lfloor q/K \rfloor, \quad \mathbf{s}_j \mathbf{B}, \quad \mathbf{s}_j (\mathbf{W} + j\mathbf{G}) - \mathbf{s} \mathbf{G}, \quad \mathbf{s}_j (\mathbf{Y} + \ell \mathbf{G}).$$

Step 2: “delay” the sampling of $\mathbf{D}$ to KeyGen [CW24]

Add $\mathbf{B}, \mathbf{W}, \mathbf{Y} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{v} \in \mathbb{Z}_q^n$ to mpk, and $\mathbf{T}_\mathbf{B}$ to msk.

- Ciphertext carries, for $j \in [\ell]$:

$$\mathbf{s}_j \mathbf{v}^\top + x_j \lfloor q/K \rfloor, \quad \mathbf{s}_j \mathbf{B}, \quad \mathbf{s}_j (\mathbf{W} + j\mathbf{G}) - \mathbf{s}\mathbf{G}, \quad \mathbf{s}_j (\mathbf{Y} + \ell\mathbf{G}).$$

- Secret key carries Gaussian preimages $(\mathbf{z}_j, \mathbf{r}_j, \mathbf{p}_j)$ of $\mathbf{v}^\top$ under $[\mathbf{B} \mid \mathbf{W} + j\mathbf{G} \mid \mathbf{Y} + \ell\mathbf{G}]$, with $\mathbf{d}_j^\top := \mathbf{G}\mathbf{r}_j^\top$ and $\mathbf{D} = [\mathbf{d}_1^\top \mid \cdots \mid \mathbf{d}_\ell^\top]$.

Step 2: “delay” the sampling of $\mathbf{D}$ to KeyGen [CW24]

Add $\mathbf{B}, \mathbf{W}, \mathbf{Y} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{v} \in \mathbb{Z}_q^n$ to mpk, and $\mathbf{T}_\mathbf{B}$ to msk.

- **Ciphertext** carries, for $j \in [\ell]$:

$$\mathbf{s}_j \mathbf{v}^\top + x_j \lfloor q/K \rfloor, \quad \mathbf{s}_j \mathbf{B}, \quad \mathbf{s}_j (\mathbf{W} + j\mathbf{G}) - \mathbf{s} \mathbf{G}, \quad \mathbf{s}_j (\mathbf{Y} + \ell \mathbf{G}).$$

- **Secret key** carries Gaussian preimages $(\mathbf{z}_j, \mathbf{r}_j, \mathbf{p}_j)$ of $\mathbf{v}^\top$ under $[\mathbf{B} \mid \mathbf{W} + j\mathbf{G} \mid \mathbf{Y} + \ell \mathbf{G}]$, with $\mathbf{d}_j^\top := \mathbf{G} \mathbf{r}_j^\top$ and $\mathbf{D} = [\mathbf{d}_1^\top \mid \cdots \mid \mathbf{d}_\ell^\top]$.
- **Decryption** recombines them into $\mathbf{s} \mathbf{d}_j^\top + x_j \lfloor q/K \rfloor$ — exactly an **ALS ciphertext** — then runs ALS decryption.

Step 2: “delay” the sampling of $\mathbf{D}$ to KeyGen [CW24]

Add $\mathbf{B}, \mathbf{W}, \mathbf{Y} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{v} \in \mathbb{Z}_q^n$ to mpk, and $\mathbf{T}_\mathbf{B}$ to msk.

- **Ciphertext** carries, for $j \in [\ell]$:

$$\mathbf{s}_j \mathbf{v}^\top + x_j \lfloor q/K \rfloor, \quad \mathbf{s}_j \mathbf{B}, \quad \mathbf{s}_j (\mathbf{W} + j\mathbf{G}) - \mathbf{s} \mathbf{G}, \quad \mathbf{s}_j (\mathbf{Y} + \ell \mathbf{G}).$$

- **Secret key** carries Gaussian preimages $(\mathbf{z}_j, \mathbf{r}_j, \mathbf{p}_j)$ of $\mathbf{v}^\top$ under $[\mathbf{B} \mid \mathbf{W} + j\mathbf{G} \mid \mathbf{Y} + \ell \mathbf{G}]$, with $\mathbf{d}_j^\top := \mathbf{G} \mathbf{r}_j^\top$ and $\mathbf{D} = [\mathbf{d}_1^\top \mid \cdots \mid \mathbf{d}_\ell^\top]$.
- **Decryption** recombines them into $\mathbf{s} \mathbf{d}_j^\top + x_j \lfloor q/K \rfloor$ — exactly an **ALS ciphertext** — then runs ALS decryption.

The term $\mathbf{Y} + \ell \mathbf{G}$ **binds the length** ℓ : a different $\mathbf{D}_\ell$ per length, preventing prefix attacks.

The strict UIPFE scheme

Π_1 (error terms omitted)

Setup: $(\mathbf{A}, \mathbf{T}_\mathbf{A}), (\mathbf{B}, \mathbf{T}_\mathbf{B}) \leftarrow \text{TrapGen}; \mathbf{W}, \mathbf{Y} \leftarrow \$ \mathbb{Z}_q^{n \times m}; \mathbf{v} \leftarrow \$ \mathbb{Z}_q^n; k \leftarrow \$ \{0, 1\}^\lambda$

Enc($\mathbf{x}$): $\mathbf{c}_0 := \mathbf{s}\mathbf{A},$ for $j \in [\ell]:$

$$\mathbf{c}_{1,j} := \mathbf{s}_j \mathbf{v}^\top + x_j \lfloor q/K \rfloor, \quad \mathbf{c}_{2,j} := \mathbf{s}_j \mathbf{B},$$

$$\mathbf{c}_{3,j} := \mathbf{s}_j(\mathbf{W} + j\mathbf{G}) - \mathbf{s}\mathbf{G}, \quad \mathbf{c}_{4,j} := \mathbf{s}_j(\mathbf{Y} + \ell\mathbf{G})$$

KeyGen($\mathbf{y}$): (randomness $\text{PRF}(k, \cdot)$) $\begin{bmatrix} \mathbf{z}_j^\top \\ \mathbf{r}_j^\top \\ \mathbf{p}_j^\top \end{bmatrix} \leftarrow [\mathbf{B} \mid \mathbf{W} + j\mathbf{G} \mid \mathbf{Y} + \ell\mathbf{G}]^{-1}(\mathbf{v}^\top)$

$$\mathbf{D} = [\mathbf{G}\mathbf{r}_1^\top \mid \cdots \mid \mathbf{G}\mathbf{r}_\ell^\top], \quad \mathbf{T} \leftarrow \mathbf{A}^{-1}(\mathbf{D}), \quad \mathbf{t}_\mathbf{y}^\top := \mathbf{T}\mathbf{y}^\top$$

Correctness

Using the secret-key preimages, each ciphertext block **collapses to an ALS ciphertext**:

$$\mathbf{c}_{1,j} - [\mathbf{c}_{2,j} \mid \mathbf{c}_{3,j} \mid \mathbf{c}_{4,j}] \cdot [\mathbf{z}_j \mid \mathbf{r}_j \mid \mathbf{p}_j]^\top \approx \mathbf{s} \mathbf{d}_j^\top + x_j \lfloor q/K \rfloor.$$

Correctness

Using the secret-key preimages, each ciphertext block **collapses to an ALS ciphertext**:

$$\mathbf{c}_{1,j} - [\mathbf{c}_{2,j} \mid \mathbf{c}_{3,j} \mid \mathbf{c}_{4,j}] \cdot [\mathbf{z}_j \mid \mathbf{r}_j \mid \mathbf{p}_j]^\top \approx \mathbf{s} \mathbf{d}_j^\top + x_j \lfloor q/K \rfloor.$$

Concatenating over j yields $\mathbf{s} \mathbf{D} + \lfloor q/K \rfloor \cdot \mathbf{x}$. Together with $\mathbf{c}_0 = \mathbf{s} \mathbf{A}$ and $\mathbf{t}_y^\top = \mathbf{T} \mathbf{y}^\top$ (where $\mathbf{A} \mathbf{T} = \mathbf{D}$), this is exactly an **ALS ciphertext/key pair**, so ALS decryption recovers $\langle \mathbf{x}, \mathbf{y} \rangle$.

Security proof: two steps

Let ℓ^* be the challenge length (guessed up front, losing a factor T).

Step 1 — randomize the challenge ciphertext

For $k = 1, \dots, \ell^*$: program $\mathbf{W} = \mathbf{B}\tilde{\mathbf{W}}_k - k\mathbf{G}$ and $\mathbf{Y} = \mathbf{B}\tilde{\mathbf{Y}} - \ell^*\mathbf{G}$ (leftover hash), rewrite $\mathbf{v}$ through the preimages, then invoke **LWE** to replace $\mathbf{c}_{2,k}, \mathbf{c}_{3,k}, \mathbf{c}_{4,k}$ with **uniformly random**.

Step 2 — reduce to ALS[ℓ^*]

Sample $\mathbf{T}_\ell$ and set $\mathbf{D}_\ell = \mathbf{A}\mathbf{T}_\ell$ for every $\ell \neq \ell^*$; embed the ALS challenge at $\ell = \ell^*$, i.e. $\mathbf{A} = \mathbf{A}_{\text{ALS}}$ and $\mathbf{D}_{\ell^*} = \mathbf{D}_{\text{ALS}}$.

Security proof sketch (Step 1, fix $k \in [\ell^*]$)

$$\mathbf{c}_0 := \mathbf{s}\mathbf{A}$$

$$\mathbf{c}_{1,k} := \mathbf{s}_k \mathbf{v}^\top + x_k \lfloor q/K \rfloor$$

$$\mathbf{c}_{2,k} := \mathbf{s}_k \mathbf{B}$$

$$\mathbf{c}_{3,k} := \mathbf{s}_k (\mathbf{W} + k\mathbf{G}) - \mathbf{s}\mathbf{G}$$

$$\mathbf{c}_{4,k} := \mathbf{s}_k (\mathbf{Y} + \ell^* \mathbf{G})$$

Security proof sketch (Step 1, fix $k \in [\ell^*]$)

- Hybrids over $k \in [\ell^*]$: $\mathbf{W} = \mathbf{B}\tilde{\mathbf{W}}_k - k\mathbf{G}$, $\mathbf{Y} = \mathbf{B}\tilde{\mathbf{Y}} - \ell^*\mathbf{G}$

$$\mathbf{c}_0 := \mathbf{s}\mathbf{A}$$

$$\mathbf{c}_{1,k} := \mathbf{s}_k \mathbf{v}^\top + x_k \lfloor q/K \rfloor$$

$$\mathbf{c}_{2,k} := \mathbf{s}_k \mathbf{B}$$

$$\mathbf{c}_{3,k} := \mathbf{s}_k (\mathbf{W} + k\mathbf{G}) - \mathbf{s}\mathbf{G}$$

$$\mathbf{c}_{4,k} := \mathbf{s}_k (\mathbf{Y} + \ell^*\mathbf{G})$$

program $\mathbf{W}, \mathbf{Y}$ (leftover hash lemma)

Security proof sketch (Step 1, fix $k \in [\ell^*]$)

- Hybrids over $k \in [\ell^*]$: $\mathbf{W} = \mathbf{B}\tilde{\mathbf{W}}_k - k\mathbf{G}$, $\mathbf{Y} = \mathbf{B}\tilde{\mathbf{Y}} - \ell^*\mathbf{G}$
 $\mathbf{v}^\top = [\mathbf{B} \mid \mathbf{W} + k\mathbf{G} \mid \mathbf{Y} + \ell^*\mathbf{G}] \mathbf{k}_k^\top$

$$\mathbf{c}_0 := \mathbf{s}\mathbf{A}$$

$$\mathbf{c}_{1,k} := \mathbf{s}_k \mathbf{v}^\top + x_k \lfloor q/K \rfloor$$

$$\mathbf{c}_{2,k} := \mathbf{s}_k \mathbf{B}$$

$$\mathbf{c}_{3,k} := \mathbf{s}_k (\mathbf{W} + k\mathbf{G}) - \mathbf{s}\mathbf{G}$$

$$\mathbf{c}_{4,k} := \mathbf{s}_k (\mathbf{Y} + \ell^*\mathbf{G})$$

program $\mathbf{v}$ (SamplePre)

Security proof sketch (Step 1, fix $k \in [\ell^*]$)

- Hybrids over $k \in [\ell^*]$: $\mathbf{W} = \mathbf{B}\tilde{\mathbf{W}}_k - k\mathbf{G}$, $\mathbf{Y} = \mathbf{B}\tilde{\mathbf{Y}} - \ell^*\mathbf{G}$
 $\mathbf{v}^\top = [\mathbf{B} \mid \mathbf{W} + k\mathbf{G} \mid \mathbf{Y} + \ell^*\mathbf{G}] \mathbf{k}_k^\top$

$$\mathbf{c}_0 := \mathbf{s}\mathbf{A}$$

$$\mathbf{c}_{1,k} := \mathbf{s}_k \mathbf{v}^\top + x_k \lfloor q/K \rfloor$$

$$\mathbf{c}_{2,k} := \mathbf{s}_k \mathbf{B}$$

$$\mathbf{c}_{3,k} := \mathbf{s}_k (\mathbf{W} + k\mathbf{G}) - \mathbf{s}\mathbf{G}$$

$$\mathbf{c}_{4,k} := \mathbf{s}_k (\mathbf{Y} + \ell^*\mathbf{G})$$

$\mathbf{B}$ uniform; answer other keys via trapdoor

Security proof sketch (Step 1, fix $k \in [\ell^*]$)

- Hybrids over $k \in [\ell^*]$: $\mathbf{W} = \mathbf{B}\tilde{\mathbf{W}}_k - k\mathbf{G}$, $\mathbf{Y} = \mathbf{B}\tilde{\mathbf{Y}} - \ell^*\mathbf{G}$
 $\mathbf{v}^\top = [\mathbf{B} \mid \mathbf{W} + k\mathbf{G} \mid \mathbf{Y} + \ell^*\mathbf{G}] \mathbf{k}_k^\top$

$$\mathbf{c}_0 := \mathbf{s}\mathbf{A}$$

$$\mathbf{c}_{1,k} := [\mathbf{c}_{2,k} \mid \mathbf{c}_{3,k} \mid \mathbf{c}_{4,k}] \mathbf{k}_k^\top + \mathbf{s}\mathbf{d}_k^\top + x_k \lfloor q/K \rfloor$$

$$\mathbf{c}_{2,k} := \mathbf{s}_k \mathbf{B}$$

$$\mathbf{c}_{3,k} := \mathbf{s}_k (\mathbf{W} + k\mathbf{G}) - \mathbf{s}\mathbf{G}$$

$$\mathbf{c}_{4,k} := \mathbf{s}_k (\mathbf{Y} + \ell^*\mathbf{G})$$

rewrite $\mathbf{c}_{1,k}$ (noise flooding)

Security proof sketch (Step 1, fix $k \in [\ell^*]$)

- Hybrids over $k \in [\ell^*]$: $\mathbf{W} = \mathbf{B}\tilde{\mathbf{W}}_k - k\mathbf{G}$, $\mathbf{Y} = \mathbf{B}\tilde{\mathbf{Y}} - \ell^*\mathbf{G}$
 $\mathbf{v}^\top = [\mathbf{B} \mid \mathbf{W} + k\mathbf{G} \mid \mathbf{Y} + \ell^*\mathbf{G}] \mathbf{k}_k^\top$

$$\mathbf{c}_0 := \mathbf{s}\mathbf{A}$$

$$\mathbf{c}_{1,k} := [\mathbf{c}_{2,k} \mid \mathbf{c}_{3,k} \mid \mathbf{c}_{4,k}] \mathbf{k}_k^\top + \mathbf{s}\mathbf{d}_k^\top + x_k \lfloor q/K \rfloor$$

$$\mathbf{c}_{2,k} := \mathbf{s}_k \mathbf{B}$$

$$\mathbf{c}_{3,k} := \mathbf{c}_{2,k} \tilde{\mathbf{W}}_k - \mathbf{s}\mathbf{G}$$

$$\mathbf{c}_{4,k} := \mathbf{c}_{2,k} \tilde{\mathbf{Y}}$$

fold $\mathbf{c}_{3,k}, \mathbf{c}_{4,k}$ onto $\mathbf{c}_{2,k}$ (noise flooding)

Security proof sketch (Step 1, fix $k \in [\ell^*]$)

- Hybrids over $k \in [\ell^*]$: $\mathbf{W} = \mathbf{B}\tilde{\mathbf{W}}_k - k\mathbf{G}$, $\mathbf{Y} = \mathbf{B}\tilde{\mathbf{Y}} - \ell^*\mathbf{G}$
 $\mathbf{v}^\top = [\mathbf{B} \mid \mathbf{W} + k\mathbf{G} \mid \mathbf{Y} + \ell^*\mathbf{G}] \mathbf{k}_k^\top$

$$\mathbf{c}_0 := \mathbf{s}\mathbf{A}$$

$$\mathbf{c}_{1,k} := [\mathbf{c}_{2,k} \mid \mathbf{c}_{3,k} \mid \mathbf{c}_{4,k}] \mathbf{k}_k^\top + \mathbf{s}\mathbf{d}_k^\top + x_k \lfloor q/K \rfloor$$

$$\mathbf{c}_{2,k} := \leftarrow \$ \mathbb{Z}_q^m$$

$$\mathbf{c}_{3,k} := \mathbf{c}_{2,k}\tilde{\mathbf{W}}_k - \mathbf{s}\mathbf{G}$$

$$\mathbf{c}_{4,k} := \mathbf{c}_{2,k}\tilde{\mathbf{Y}}$$

$$\mathbf{c}_{2,k} \leftarrow \$ \mathbb{Z}_q^m \quad (\text{LWE})$$

Security proof sketch (Step 1, fix $k \in [\ell^*]$)

- Hybrids over $k \in [\ell^*]$: $\mathbf{W} = \mathbf{B}\tilde{\mathbf{W}}_k - k\mathbf{G}$, $\mathbf{Y} = \mathbf{B}\tilde{\mathbf{Y}} - \ell^*\mathbf{G}$
 $\mathbf{v}^\top = [\mathbf{B} \mid \mathbf{W} + k\mathbf{G} \mid \mathbf{Y} + \ell^*\mathbf{G}] \mathbf{k}_k^\top$

$$\mathbf{c}_0 := \mathbf{s}\mathbf{A}$$

$$\mathbf{c}_{1,k} := [\mathbf{c}_{2,k} \mid \mathbf{c}_{3,k} \mid \mathbf{c}_{4,k}] \mathbf{k}_k^\top + \mathbf{s}\mathbf{d}_k^\top + x_k \lfloor q/K \rfloor$$

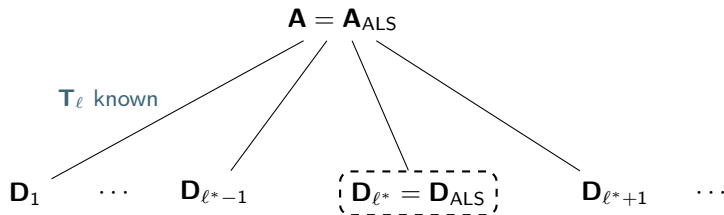
$$\mathbf{c}_{2,k} := \leftarrow \$ \mathbb{Z}_q^m$$

$$\mathbf{c}_{3,k} := \leftarrow \$ \mathbb{Z}_q^m$$

$$\mathbf{c}_{4,k} := \leftarrow \$ \mathbb{Z}_q^m$$

$\mathbf{c}_{3,k}, \mathbf{c}_{4,k} \leftarrow \$ \mathbb{Z}_q^m$ (GLHL) $\Rightarrow$ repeat over k , then reduce to ALS

Security proof: reduction to $\text{ALS}[\ell^*]$



- keys for $\ell \neq \ell^*$ are answered with the known $\mathbf{T}_\ell$;
- keys for $\ell = \ell^*$ are forwarded to the ALS challenger.

Permissive UIPFE

Goal: a key of length ℓ' decrypts *any* ciphertext of length $\ell \geq \ell'$, recovering $\langle \mathbf{x}|_{[\ell']}, \mathbf{y} \rangle$.

- Encryption: encrypt **every prefix** $\mathbf{x}|_{[\ell']}$, $\ell' \in [\ell]$, sharing the same $\mathbf{s}$ and $\mathbf{c}_0$.

Permissive UIPFE

Goal: a key of length ℓ' decrypts *any* ciphertext of length $\ell \geq \ell'$, recovering $\langle \mathbf{x}|_{[\ell']}, \mathbf{y} \rangle$.

- Encryption: encrypt **every prefix** $\mathbf{x}|_{[\ell']}$, $\ell' \in [\ell]$, sharing the same $\mathbf{s}$ and $\mathbf{c}_0$.
- Decryption: use the block matching the key length.

Permissive UIPFE

Goal: a key of length ℓ' decrypts *any* ciphertext of length $\ell \geq \ell'$, recovering $\langle \mathbf{x}|_{[\ell']}, \mathbf{y} \rangle$.

- Encryption: encrypt **every prefix** $\mathbf{x}|_{[\ell']}$, $\ell' \in [\ell]$, sharing the same $\mathbf{s}$ and $\mathbf{c}_0$.
- Decryption: use the block matching the key length.
- Cost: ciphertexts grow to $O(\ell^2)$.

Permissive UIPFE

Goal: a key of length ℓ' decrypts *any* ciphertext of length $\ell \geq \ell'$, recovering $\langle \mathbf{x}|_{[\ell']}, \mathbf{y} \rangle$.

- Encryption: encrypt **every prefix** $\mathbf{x}|_{[\ell']}$, $\ell' \in [\ell]$, sharing the same $\mathbf{s}$ and $\mathbf{c}_0$.
- Decryption: use the block matching the key length.
- Cost: ciphertexts grow to $O(\ell^2)$.

Proof idea: reduce to ALS $[\ell^*(\ell^* + 1)/2]$ by **embedding each length in its own block**,

$$\tilde{\mathbf{x}}_b = [\mathbf{x}_b|_{[1]} \mid \cdots \mid \mathbf{x}_b|_{[\ell^*]}], \quad \tilde{\mathbf{y}} = [\mathbf{0} \mid \cdots \mid \mathbf{y} \mid \cdots \mid \mathbf{0}],$$

so that $\langle \tilde{\mathbf{x}}_b, \tilde{\mathbf{y}} \rangle = \langle \mathbf{x}_b|_{[\ell']}, \mathbf{y} \rangle$.

Fine-grained UIPFE — almost for free

The reduction is **modular**: swap the underlying ALS IPFE for a *richer* bounded IPFE, reducing to the IB-/AB-IPFE of Lai–Liu–Wang [LLW21].

- **Identity-based**: bind $\mathbf{c}_0$ to an identity,

$$\mathbf{c}_0 := \mathbf{s}[\mathbf{A} \mid \mathbf{A}_{\text{id}}], \quad \mathbf{A}_{\text{id}} = \mathbf{G} + \sum_i b_i \mathbf{A}_i.$$

Fine-grained UIPFE — almost for free

The reduction is **modular**: swap the underlying ALS IPFE for a *richer* bounded IPFE, reducing to the IB-/AB-IPFE of Lai–Liu–Wang [LLW21].

- **Identity-based**: bind $\mathbf{c}_0$ to an identity,

$$\mathbf{c}_0 := \mathbf{s}[\mathbf{A} \mid \mathbf{A}_{\text{id}}], \quad \mathbf{A}_{\text{id}} = \mathbf{G} + \sum_i b_i \mathbf{A}_i.$$

- **Attribute-based**: use homomorphic evaluation [GSW13; Bon+14],

$$\mathbf{c}_0 := \mathbf{s}[\mathbf{A} \mid \mathbf{A}_{\mathbf{w}}], \quad \mathbf{A}_C = \text{EvalC}(\{\mathbf{A}_i\}, C);$$

decryption succeeds iff $C(\mathbf{w}) = 0$.

Fine-grained UIPFE — almost for free

The reduction is **modular**: swap the underlying ALS IPFE for a *richer* bounded IPFE, reducing to the IB-/AB-IPFE of Lai–Liu–Wang [LLW21].

- **Identity-based**: bind $\mathbf{c}_0$ to an identity,

$$\mathbf{c}_0 := \mathbf{s}[\mathbf{A} \mid \mathbf{A}_{\text{id}}], \quad \mathbf{A}_{\text{id}} = \mathbf{G} + \sum_i b_i \mathbf{A}_i.$$

- **Attribute-based**: use homomorphic evaluation [GSW13; Bon+14],

$$\mathbf{c}_0 := \mathbf{s}[\mathbf{A} \mid \mathbf{A}_{\mathbf{w}}], \quad \mathbf{A}_C = \text{EvalC}(\{\mathbf{A}_i\}, C);$$

decryption succeeds iff $C(\mathbf{w}) = 0$.

Same proof skeleton — only the *final reduction target* changes. AB-UIPFE is selective and supports depth- d circuits.

The big picture



Every UIPFE variant reduces, in a **black-box** and **modular** way, to a bounded IPFE.

Open problems

- More *efficient* permissive UIPFE — linear (not quadratic) ciphertexts?
- Lattice-based *function-hiding* IPFE? (lower bounds [Üna20])
- Lattice-based compact *quadratic* FE? (lower bounds [TÜ24])
- UIPFE with keys for the *larger* domain (opposite of permissive)?

Open problems

- More **efficient** permissive UIPFE — linear (not quadratic) ciphertexts?
- Lattice-based *function-hiding* IPFE? (lower bounds [Üna20])
- Lattice-based compact *quadratic* FE? (lower bounds [TÜ24])
- UIPFE with keys for the *larger* domain (opposite of permissive)?

Thank you!



ia.cr/2026/062

References I

- [Abd+15] M. Abdalla et al. “Simple Functional Encryption Schemes for Inner Products”. In: *PKC 2015*. Ed. by J. Katz. Vol. 9020. LNCS. Springer, Berlin, Heidelberg, 2015, pp. 733–751. DOI: 10.1007/978-3-662-46447-2_33.
- [ALS16] S. Agrawal, B. Libert, and D. Stehlé. “Fully Secure Functional Encryption for Inner Products, from Standard Assumptions”. In: *CRYPTO 2016, Part III*. Ed. by M. Robshaw and J. Katz. Vol. 9816. LNCS. Springer, Berlin, Heidelberg, Aug. 2016, pp. 333–362. DOI: 10.1007/978-3-662-53015-3_12.
- [BCS23] E. Bouscatié, G. Castagnos, and O. Sanders. “Pattern Matching in Encrypted Stream from Inner Product Encryption”. In: *PKC 2023, Part I*. Ed. by A. Boldyreva and V. Kolesnikov. Vol. 13940. LNCS. Springer, Cham, May 2023, pp. 774–801. DOI: 10.1007/978-3-031-31368-4_27.
- [Bon+14] D. Boneh et al. “Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE and Compact Garbled Circuits”. In: *EUROCRYPT 2014*. Ed. by P. Q. Nguyen and E. Oswald. Vol. 8441. LNCS. Springer, Berlin, Heidelberg, May 2014, pp. 533–556. DOI: 10.1007/978-3-642-55220-5_30.
- [CW24] V. Cini and H. Wee. “Unbounded ABE for Circuits from LWE, Revisited”. In: *ASIACRYPT 2024, Part IV*. Ed. by K.-M. Chung and Y. Sasaki. Vol. 15487. LNCS. Springer, Singapore, Dec. 2024, pp. 238–267. DOI: 10.1007/978-981-96-0894-2_8.
- [DP19] E. Dufour Sans and D. Pointcheval. “Unbounded Inner-Product Functional Encryption with Succinct Keys”. In: *ACNS 2019*. Ed. by R. H. Deng et al. Vol. 11464. LNCS. Springer, Cham, June 2019, pp. 426–441. DOI: 10.1007/978-3-030-21568-2_21.

References II

- [DP23] P. Datta and T. Pal. “Decentralized Multi-Authority Attribute-Based Inner-Product FE: Large Universe and Unbounded”. In: *PKC 2023, Part I*. Ed. by A. Boldyreva and V. Kolesnikov. Vol. 13940. LNCS. Springer, Cham, May 2023, pp. 587–621. DOI: 10.1007/978-3-031-31368-4_21.
- [GKW16] R. Goyal, V. Koppula, and B. Waters. “Semi-adaptive Security and Bundling Functionalities Made Generic and Easy”. In: *TCC 2016-B, Part II*. Ed. by M. Hirt and A. D. Smith. Vol. 9986. LNCS. Springer, Berlin, Heidelberg, 2016, pp. 361–388. DOI: 10.1007/978-3-662-53644-5_14.
- [GSW13] C. Gentry, A. Sahai, and B. Waters. “Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based”. In: *CRYPTO 2013, Part I*. Ed. by R. Canetti and J. A. Garay. Vol. 8042. LNCS. Springer, Berlin, Heidelberg, Aug. 2013, pp. 75–92. DOI: 10.1007/978-3-642-40041-4_5.
- [Kou+20] V. Koutsos et al. *Agora: A Privacy-Aware Data Marketplace*. Cryptology ePrint Archive, Report 2020/865. 2020. URL: <https://eprint.iacr.org/2020/865>.
- [LLW21] Q. Lai, F.-H. Liu, and Z. Wang. “New Lattice Two-Stage Sampling Technique and Its Applications to Functional Encryption - Stronger Security and Smaller Ciphertexts”. In: *EUROCRYPT 2021, Part I*. Ed. by A. Canteaut and F.-X. Standaert. Vol. 12696. LNCS. Springer, Cham, Oct. 2021, pp. 498–527. DOI: 10.1007/978-3-030-77870-5_18.

References III

- [Mar+19] T. Marc et al. “Privacy-Enhanced Machine Learning with Functional Encryption”. In: *ESORICS 2019, Part I*. Ed. by K. Sako, S. Schneider, and P. Y. A. Ryan. Vol. 11735. LNCS. Springer, Cham, Sept. 2019, pp. 3–21. DOI: 10.1007/978-3-030-29959-0_1.
- [PP22] P. de Perthuis and D. Pointcheval. “Two-Client Inner-Product Functional Encryption with an Application to Money-Laundering Detection”. In: *ACM CCS 2022*. Ed. by H. Yin et al. ACM Press, Nov. 2022, pp. 725–737. DOI: 10.1145/3548606.3559374.
- [TT18] J. Tomida and K. Takashima. “Unbounded Inner Product Functional Encryption from Bilinear Maps”. In: *ASIACRYPT 2018, Part II*. Ed. by T. Peyrin and S. Galbraith. Vol. 11273. LNCS. Springer, Cham, Dec. 2018, pp. 609–639. DOI: 10.1007/978-3-030-03329-3_21.
- [TÜ24] E. Tairi and A. Ünal. “Lower Bounds for Lattice-Based Compact Functional Encryption”. In: *EUROCRYPT 2024, Part II*. Ed. by M. Joye and G. Leander. Vol. 14652. LNCS. Springer, Cham, May 2024, pp. 249–279. DOI: 10.1007/978-3-031-58723-8_9.
- [Üna20] A. Ünal. “Impossibility Results for Lattice-Based Functional Encryption Schemes”. In: *EUROCRYPT 2020, Part I*. Ed. by A. Canteaut and Y. Ishai. Vol. 12105. LNCS. Springer, Cham, May 2020, pp. 169–199. DOI: 10.1007/978-3-030-45721-1_7.