Laconic Function Evaluation and ABE for RAMs from (Ring-)LWE

Fangqi Dong
IIIS, Tsinghua University

Zihan Hao
IIIS, Tsinghua University

Ethan Mook
Northeastern University

Hoeteck Wee
ENS, Paris & NTT Research

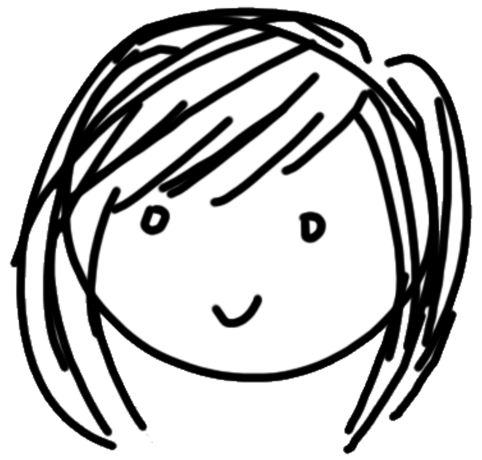
Daniel Wichs
Northeastern University & NTT Research

Crypto 2024

Laconic Function Evaluation (LFE)

Laconic Function Evaluation (LFE)

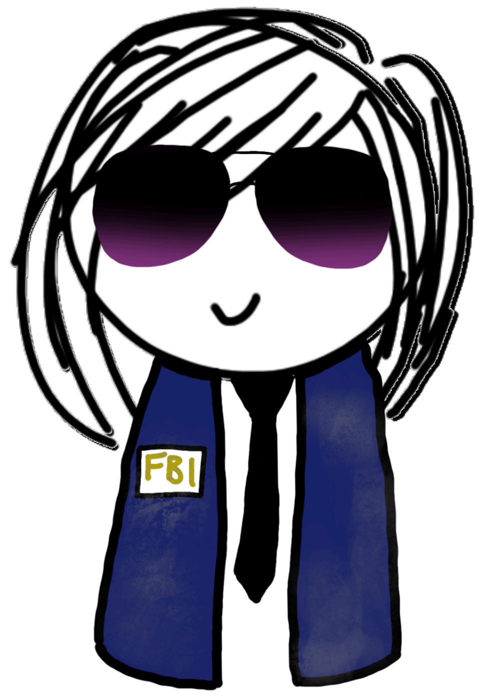
C



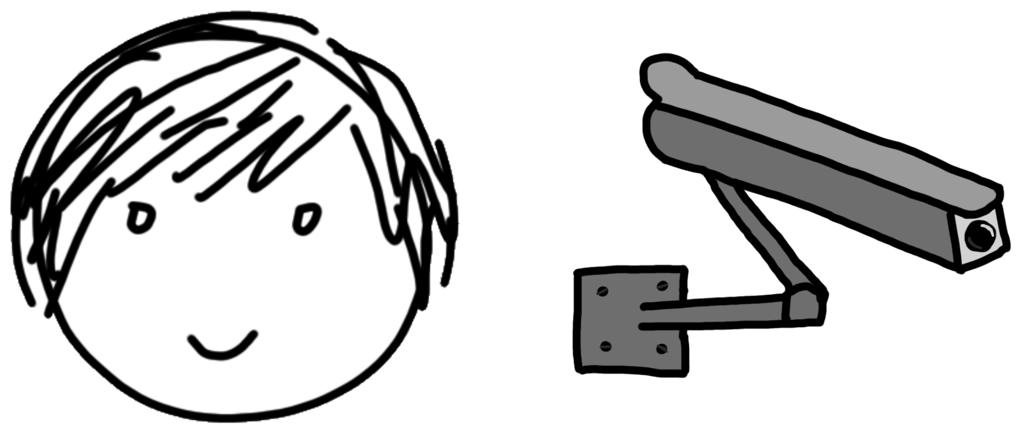
x

Laconic Function Evaluation (LFE)

C



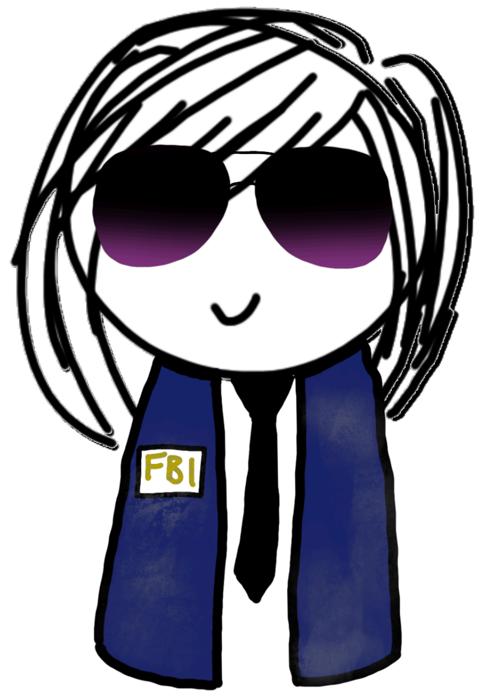
x



Laconic Function Evaluation (LFE)

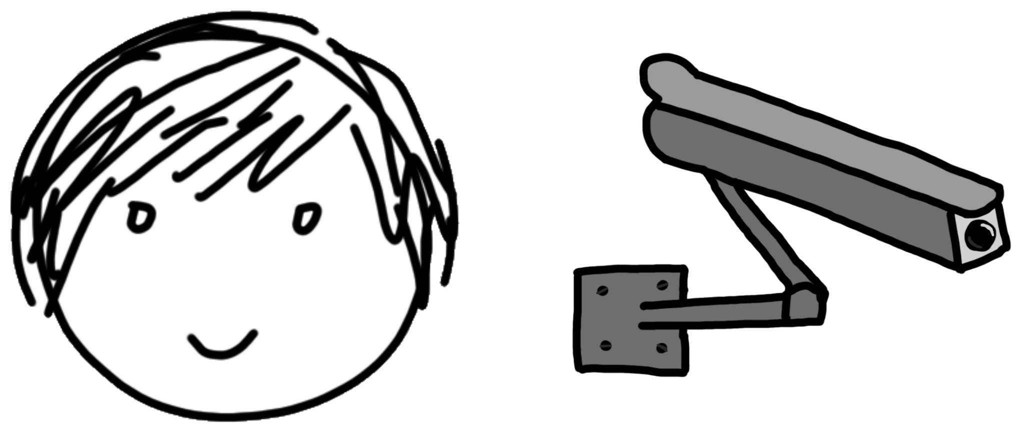
* in CRS model, CRS hidden

C



$\mathrm{dig} = \mathrm{Hash}(C)$ →

x

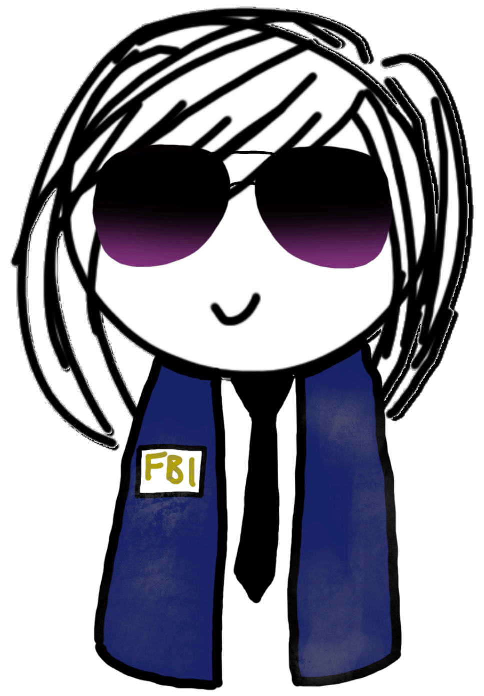


Laconic Function Evaluation (LFE)

* in CRS model, CRS hidden

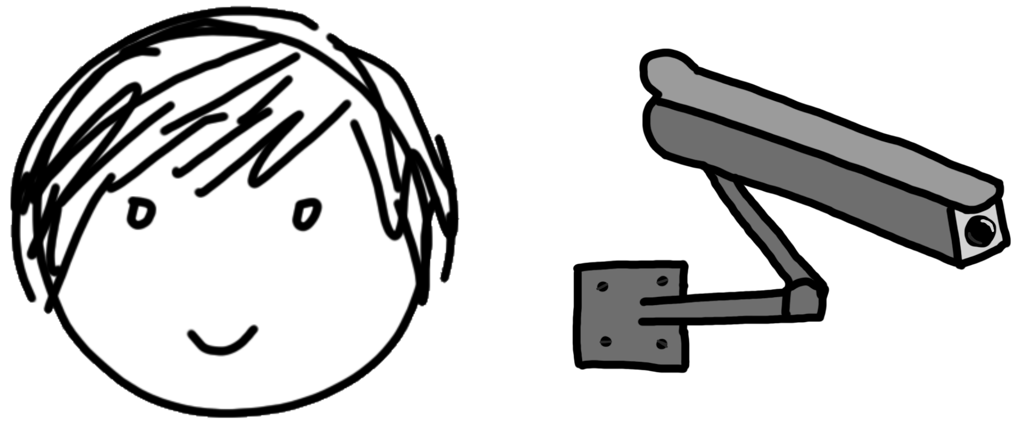
digest very short compared to $|C|$

C



$\mathsf{dig} = \mathsf{Hash}(C)$ →

x

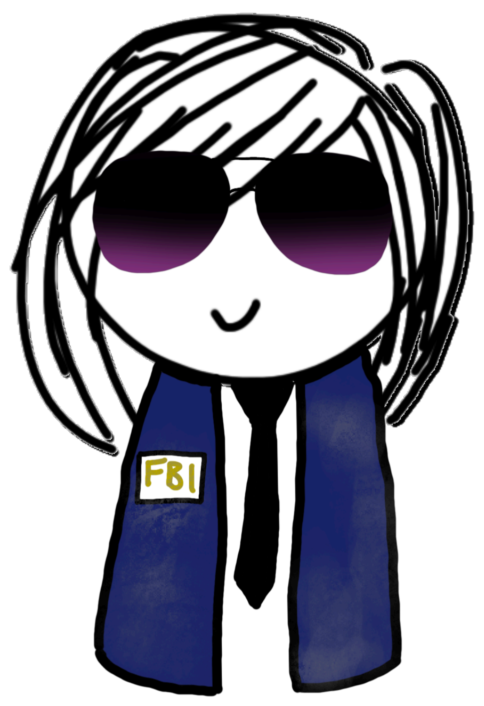


Laconic Function Evaluation (LFE)

* in CRS model, CRS hidden

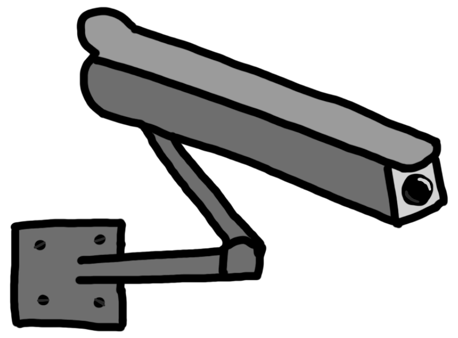
digest very short compared to $|C|$

C



$\mathsf{dig} = \mathsf{Hash}(C)$

x



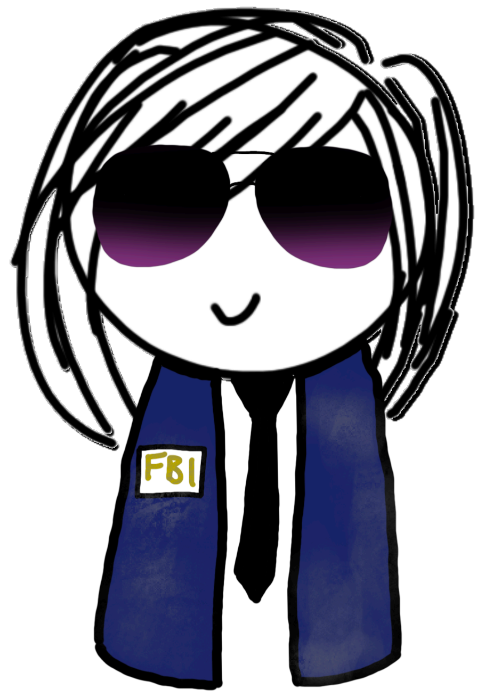
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{dig}, x)$

Laconic Function Evaluation (LFE)

* in CRS model, CRS hidden

digest very short compared to $|C|$

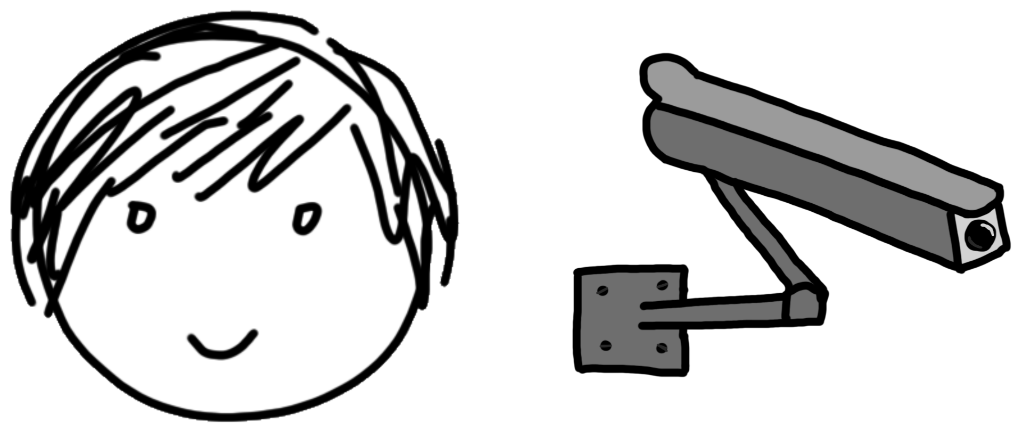
C



$\mathsf{dig} = \mathsf{Hash}(C)$ →

← ct

x



$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{dig}, x)$

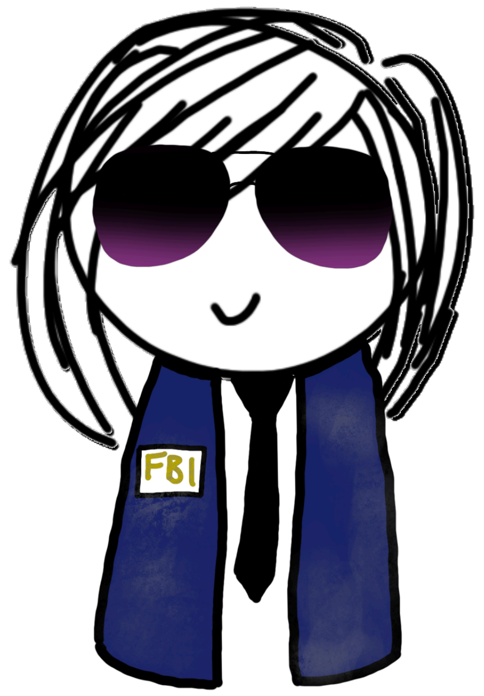
$\mathsf{Dec}(C, \mathsf{ct}) = C(x)$

Laconic Function Evaluation (LFE)

* in CRS model, CRS hidden

digest very short compared to $|C|$

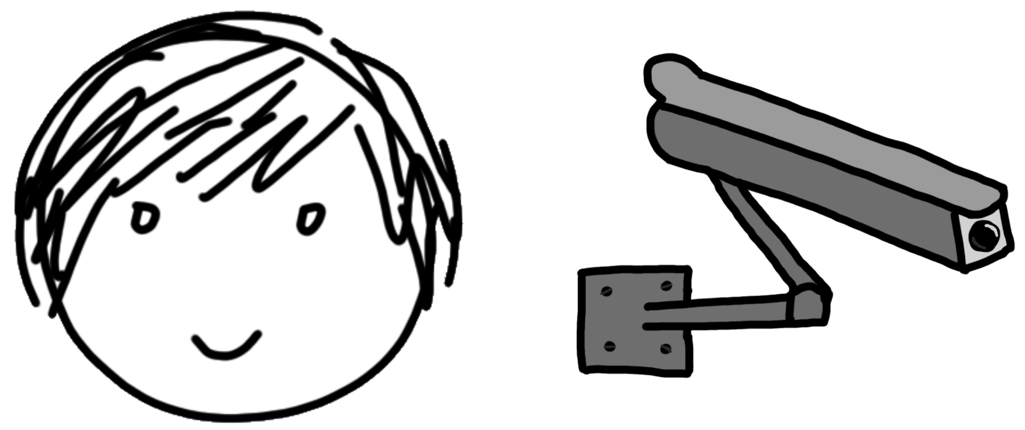
C



$\mathsf{dig} = \mathsf{Hash}(C)$

ct

x



$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{dig}, x)$

$\mathsf{Dec}(C, \mathsf{ct}) = C(x)$

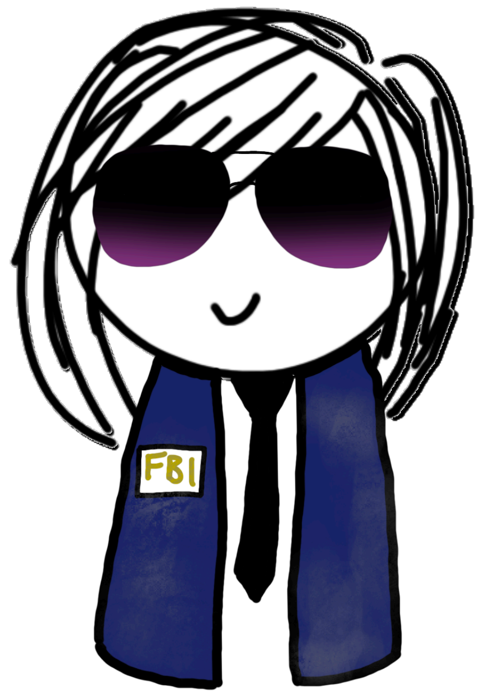
Security: Server learns nothing more than $C(x)$

Laconic Function Evaluation (LFE)

* in CRS model, CRS hidden

digest very short compared to $|C|$

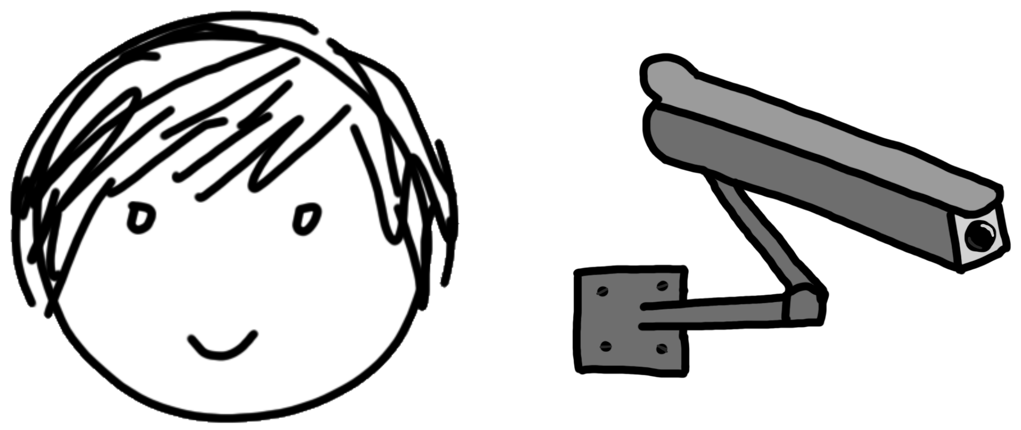
C



$\text{dig} = \text{Hash}(C)$

ct

x



$\text{ct} \leftarrow \text{Enc}(\text{dig}, x)$

$\text{Dec}(C, \text{ct}) = C(x)$

Security: Server learns nothing more than $C(x)$

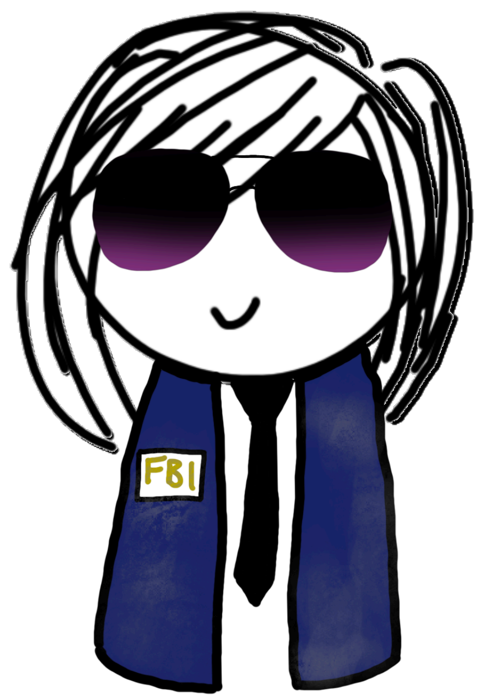
Like FHE: 2-round 2PC where Server does the computational work

Laconic Function Evaluation (LFE)

* in CRS model, CRS hidden

digest very short compared to $|C|$

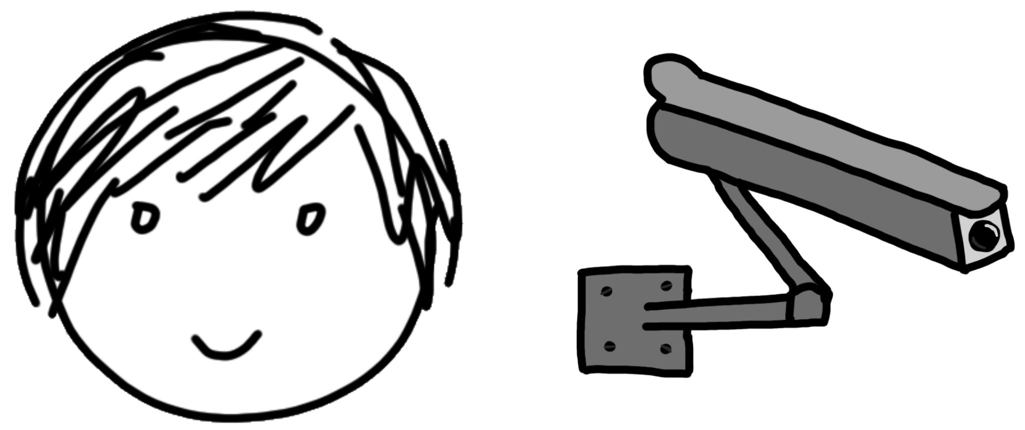
C



$\text{dig} = \text{Hash}(C)$ →

← ct

x



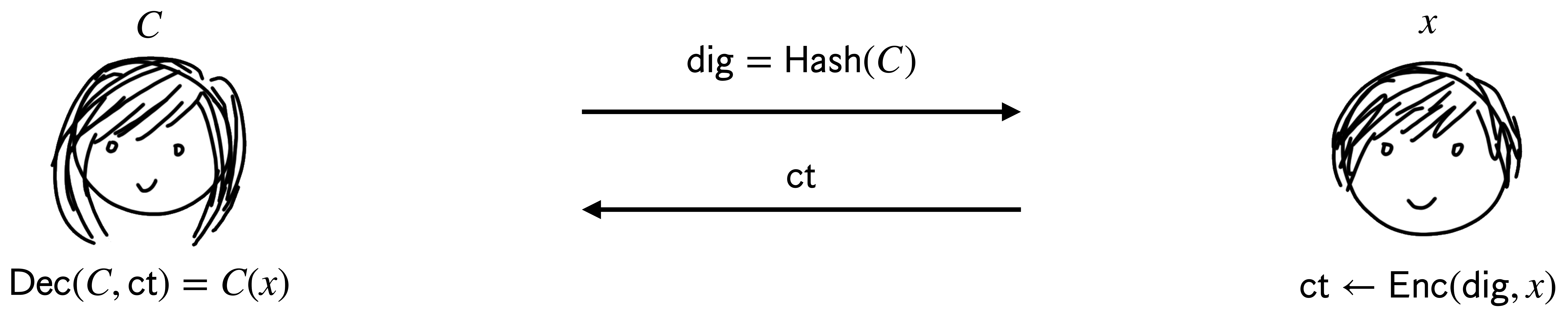
$\text{ct} \leftarrow \text{Enc}(\text{dig}, x)$

$\text{Dec}(C, \text{ct}) = C(x)$

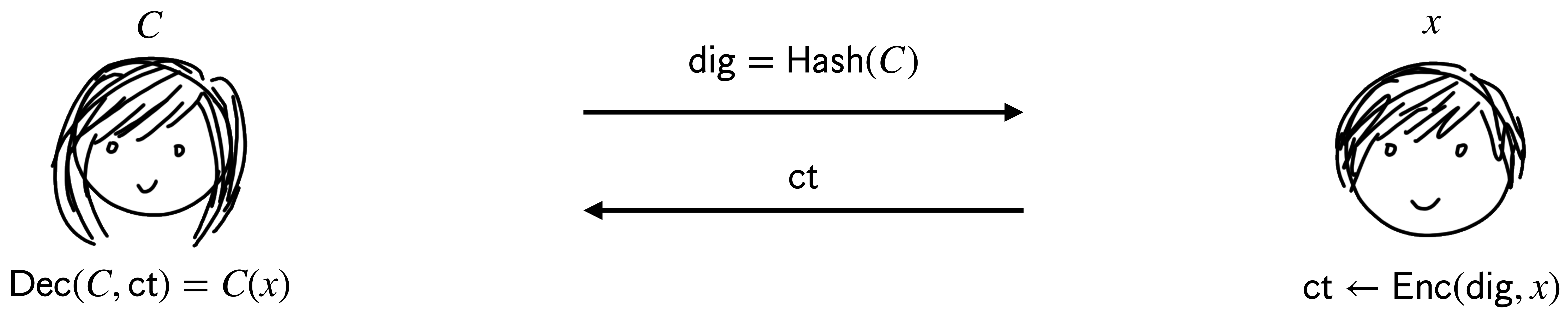
Security: Server learns nothing more than $C(x)$

Like FHE: 2-round 2PC where Server does the computational work
But "flipped": Server learns the output (instead of Client)

Laconic Function Evaluation (LFE)



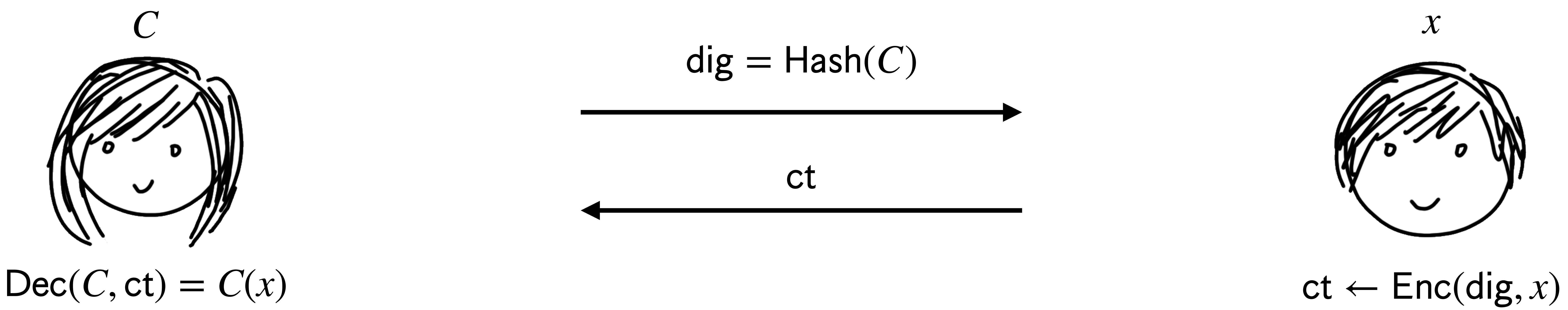
Laconic Function Evaluation (LFE)



Prior work:

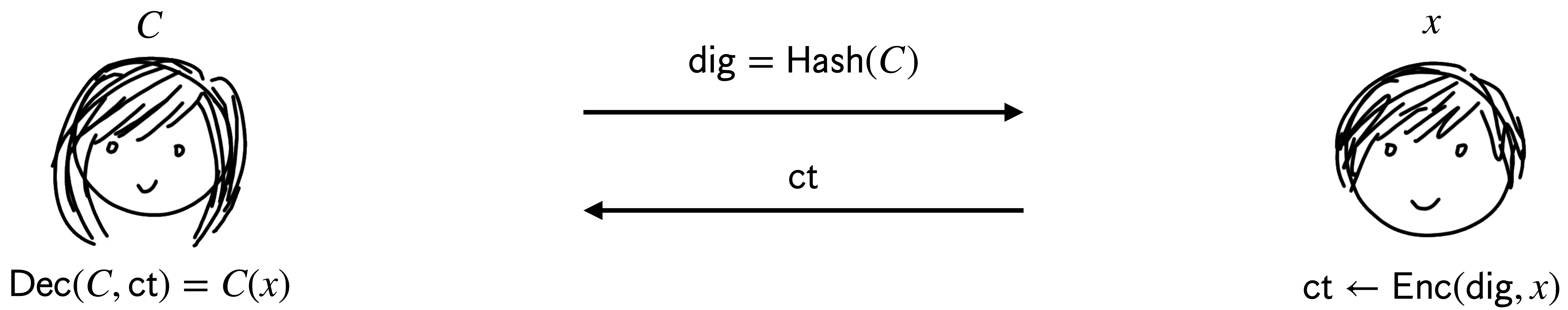
- [Quach-Wee-Wichs'17]: LFE for circuits from LWE

Laconic Function Evaluation (LFE)



Prior work:
- [Quach-Wee-Wichs'17]: LFE for circuits from LWE
- [Döttling-Gajland-Malavolta'23]: LFE for TM from iO + SSB

Laconic Function Evaluation (LFE)

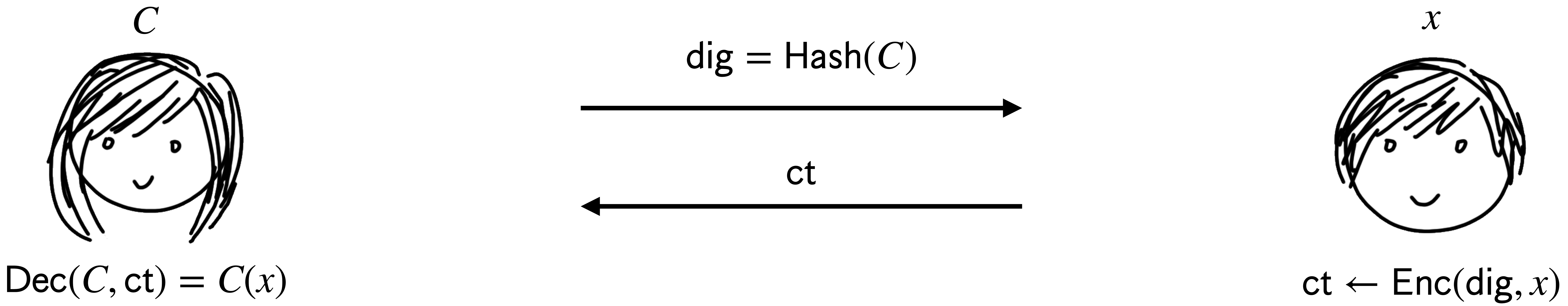


Prior work:

- [Quach-Wee-Wichs'17]: LFE for circuits from LWE
- [Döttling-Gajland-Malavolta'23]: LFE for TM from iO + SSB

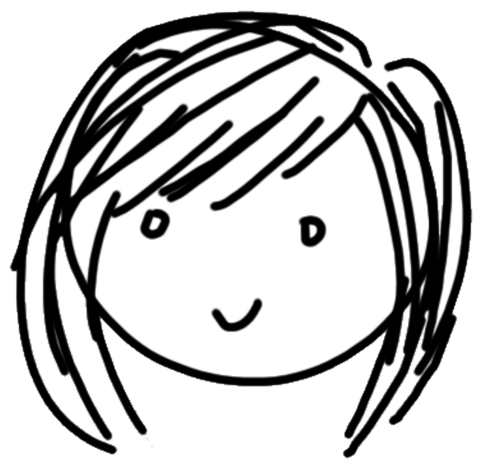
Problem: Server computation is at least linear in inputs!

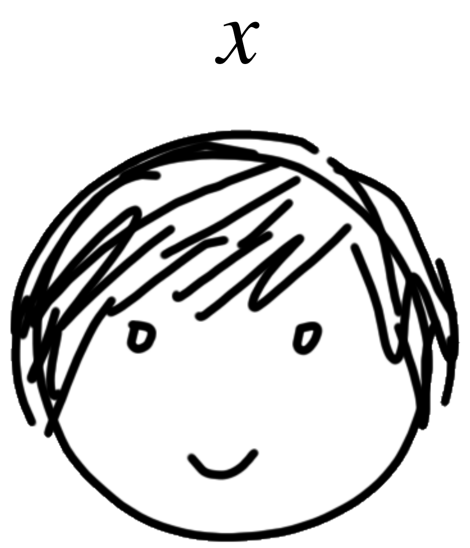
Laconic Function Evaluation (LFE)



Prior work:
- [Quach-Wee-Wichs'17]: LFE for circuits from LWE
- [Döttling-Gajland-Malavolta'23]: LFE for TM from iO + SSB
- [Dong-Hao-**M**-Wichs'24]: LFE for RAM from RingLWE (+iO)

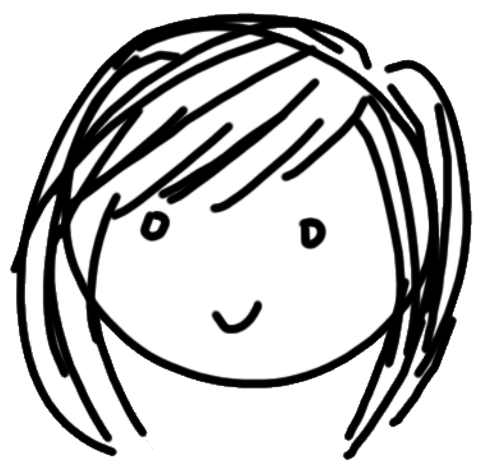
LFE for RAMs





LFE for RAMs

Goal: output RAM computation $f_{\mathsf{DB}}(x)$

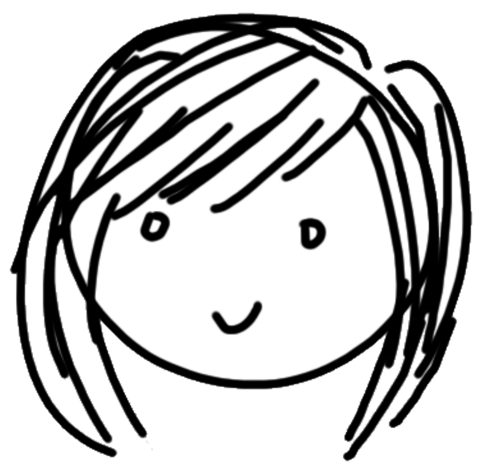


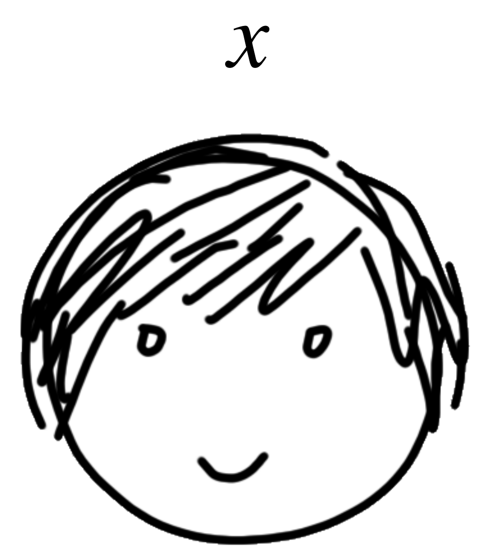
x

LFE for RAMs

Some fixed RAM program (e.g. universal)

Goal: output RAM computation $f_{\mathsf{DB}}(x)$



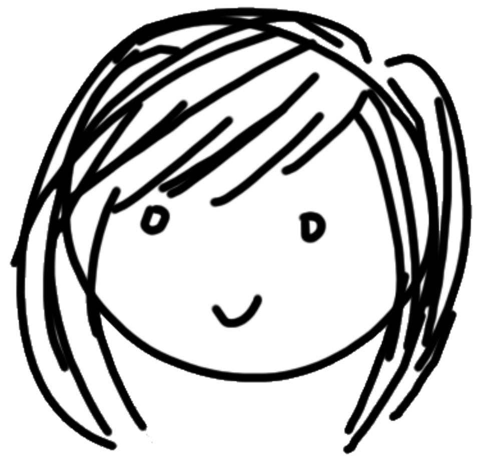


LFE for RAMs

Some fixed RAM program (e.g. universal)

Goal: output RAM computation $f_{\mathsf{DB}}(x)$

$f_{\mathsf{DB}}(x)$ has RAM runtime T



x

Server holds some arbitrary public database

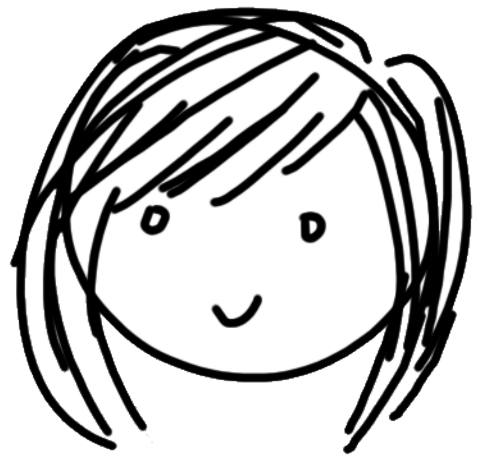
LFE for RAMs

Some fixed RAM program (e.g. universal)

Goal: output RAM computation $f_{\text{DB}}(x)$

$f_{\text{DB}}(x)$ has RAM runtime T

DB



x

LFE for RAMs

Server holds some arbitrary public database

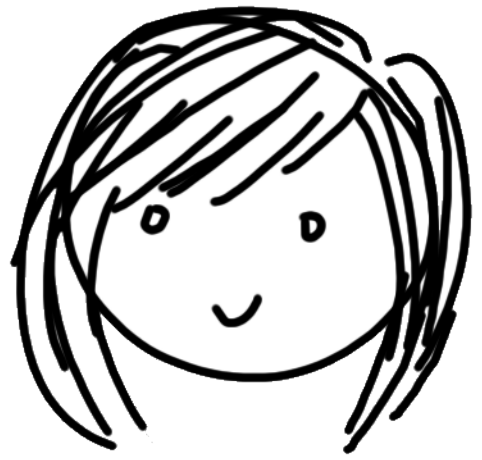
And preprocesses it

Some fixed RAM program (e.g. universal)

Goal: output RAM computation $f_{\mathsf{DB}}(x)$

$f_{\mathsf{DB}}(x)$ has RAM runtime T

$\mathsf{DB} \xrightarrow{\mathsf{Prep}} \widetilde{\mathsf{DB}}$



x

LFE for RAMs

Server holds some arbitrary public database

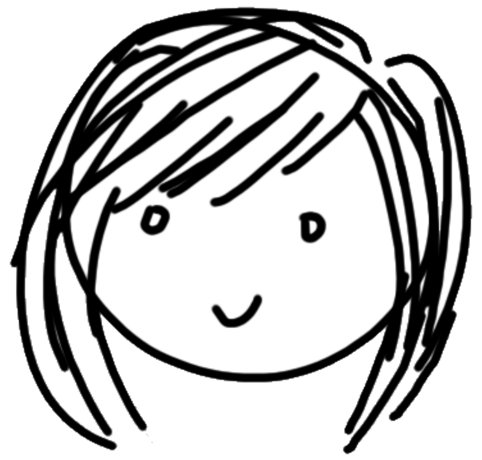
And preprocesses it

Some fixed RAM program (e.g. universal)

Goal: output RAM computation $f_{\mathsf{DB}}(x)$

$f_{\mathsf{DB}}(x)$ has RAM runtime T

$\mathsf{DB} \xrightarrow{\mathsf{Prep}} \widetilde{\mathsf{DB}}$



$\mathsf{dig} = \mathsf{Hash}(\widetilde{\mathsf{DB}})$ →

x

LFE for RAMs

Server holds some arbitrary public database

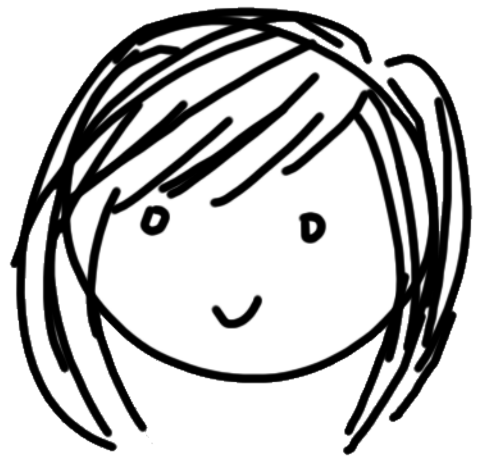
And preprocesses it

Some fixed RAM program (e.g. universal)

Goal: output RAM computation $f_{\mathsf{DB}}(x)$

$f_{\mathsf{DB}}(x)$ has RAM runtime T

$\mathsf{DB} \xrightarrow{\mathsf{Prep}} \widetilde{\mathsf{DB}}$



x

$\mathsf{dig} = \mathsf{Hash}(\widetilde{\mathsf{DB}})$ →

← ct

$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{dig}, x)$

LFE for RAMs

Server holds some arbitrary public database

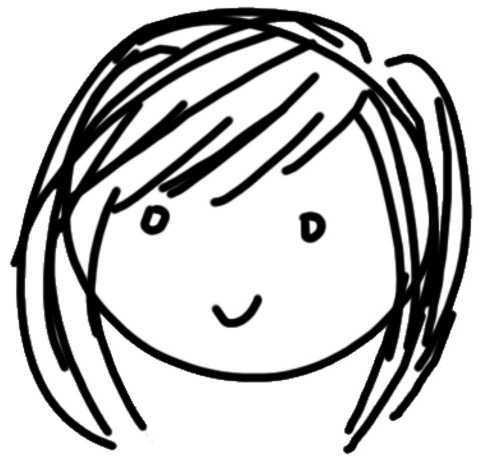
And preprocesses it

Some fixed RAM program (e.g. universal)

Goal: output RAM computation $f_{\mathsf{DB}}(x)$

$f_{\mathsf{DB}}(x)$ has RAM runtime T

$\mathsf{DB} \xrightarrow{\mathsf{Prep}} \widetilde{\mathsf{DB}}$



$\mathsf{Dec}(\widetilde{\mathsf{DB}}, \mathsf{ct}) = f_{\mathsf{DB}}(x)$

$\mathsf{dig} = \mathsf{Hash}(\widetilde{\mathsf{DB}})$ →

← ct

x

$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{dig}, x)$

LFE for RAMs

Server holds some arbitrary public database

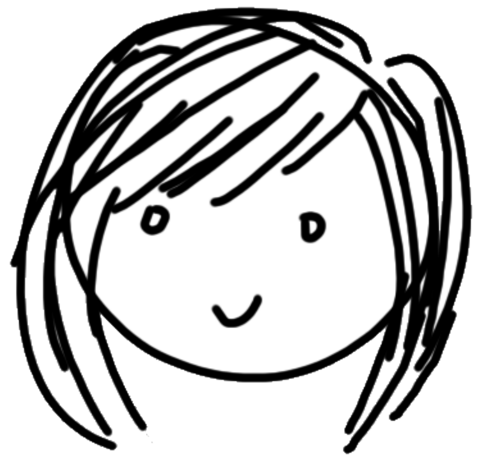
And preprocesses it

Some fixed RAM program (e.g. universal)

Goal: output RAM computation $f_{\mathsf{DB}}(x)$

$f_{\mathsf{DB}}(x)$ has RAM runtime T

$\mathsf{DB} \xrightarrow{\mathsf{Prep}} \widetilde{\mathsf{DB}}$



x

$\mathsf{dig} = \mathsf{Hash}(\widetilde{\mathsf{DB}})$ →

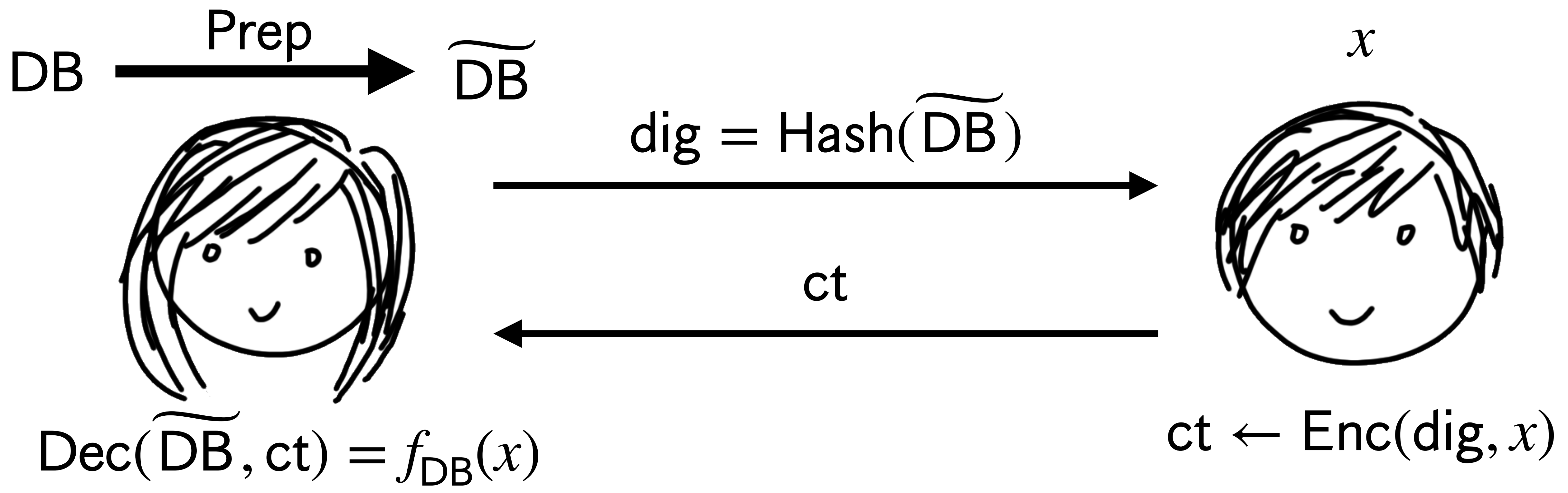
← ct

$\mathsf{Dec}(\widetilde{\mathsf{DB}}, \mathsf{ct}) = f_{\mathsf{DB}}(x)$

$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{dig}, x)$

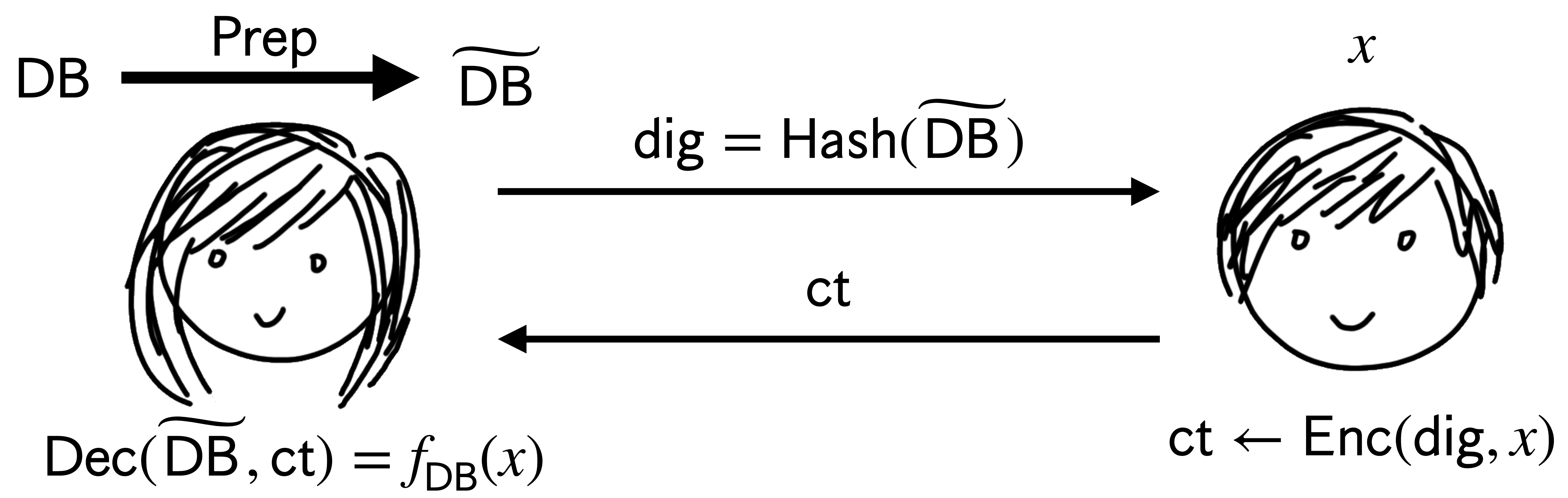
Want Dec to run in time $\approx T$

LFE for RAMs



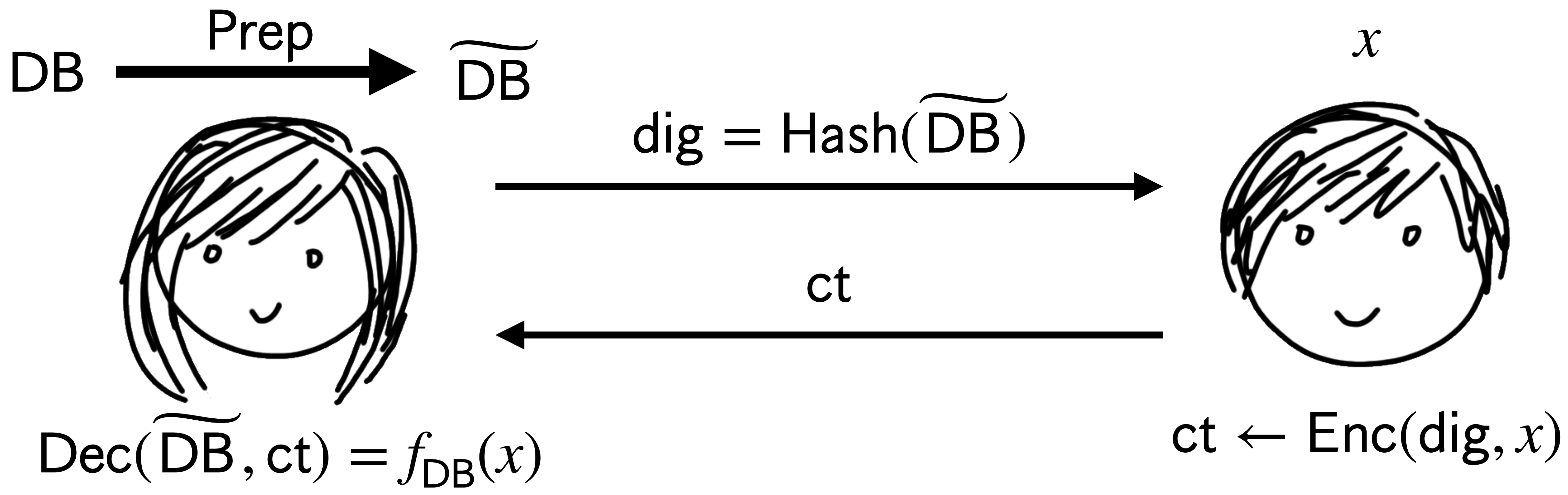
	[DH**M**W'24] (weak efficiency)	[DH**M**W'24] (strong efficiency)
Prep:		
Enc:		
Dec:		
Assumption:		

LFE for RAMs



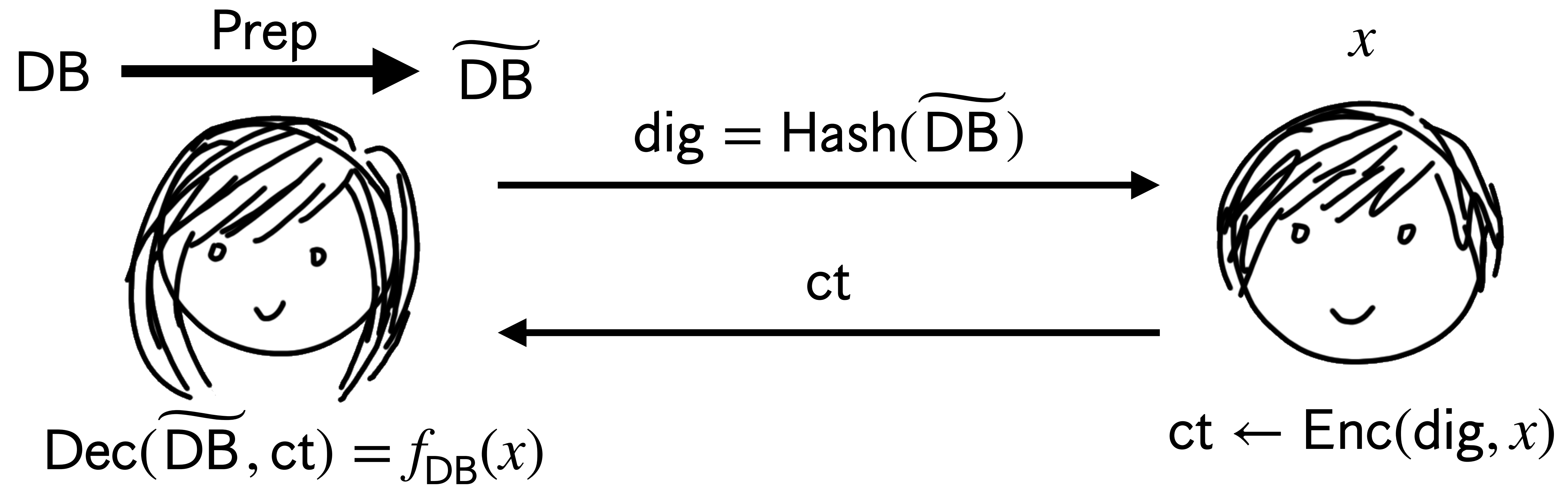
	[DH**M**W'24] (weak efficiency)	[DH**M**W'24] (strong efficiency)
Prep:	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$
Enc:		
Dec:		
Assumption:		

LFE for RAMs



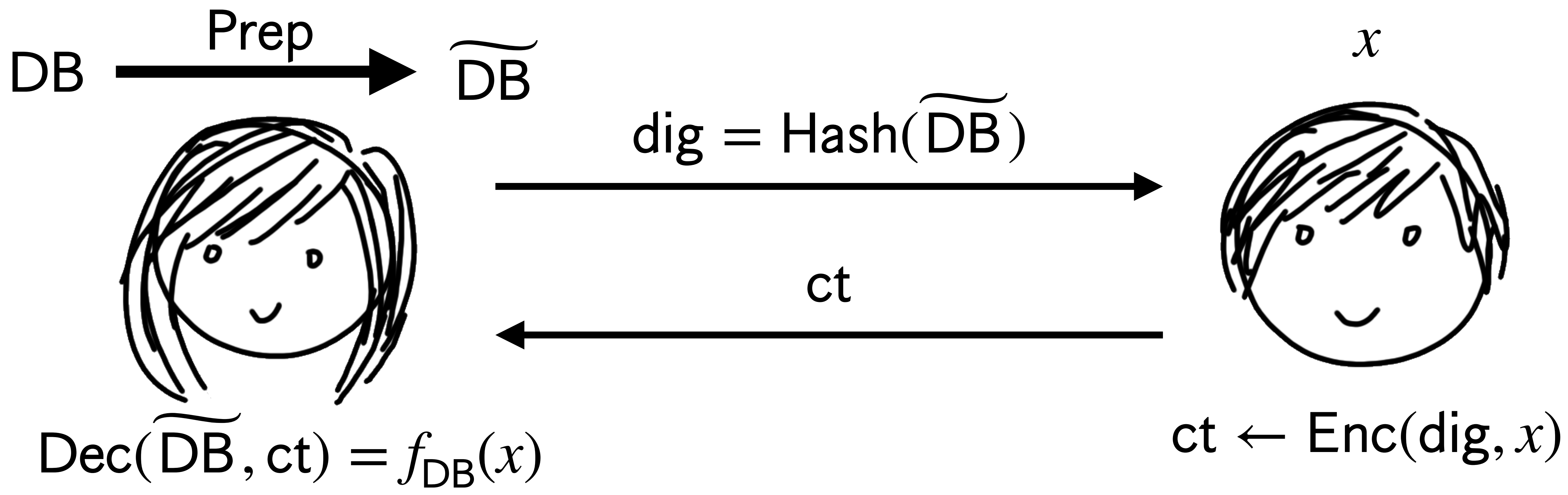
	[DH**M**W'24] (weak efficiency)	[DH**M**W'24] (strong efficiency)
Prep:	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$
Enc:	$\lvert x \rvert + T$	$\lvert x \rvert$
Dec:		
Assumption:		

LFE for RAMs



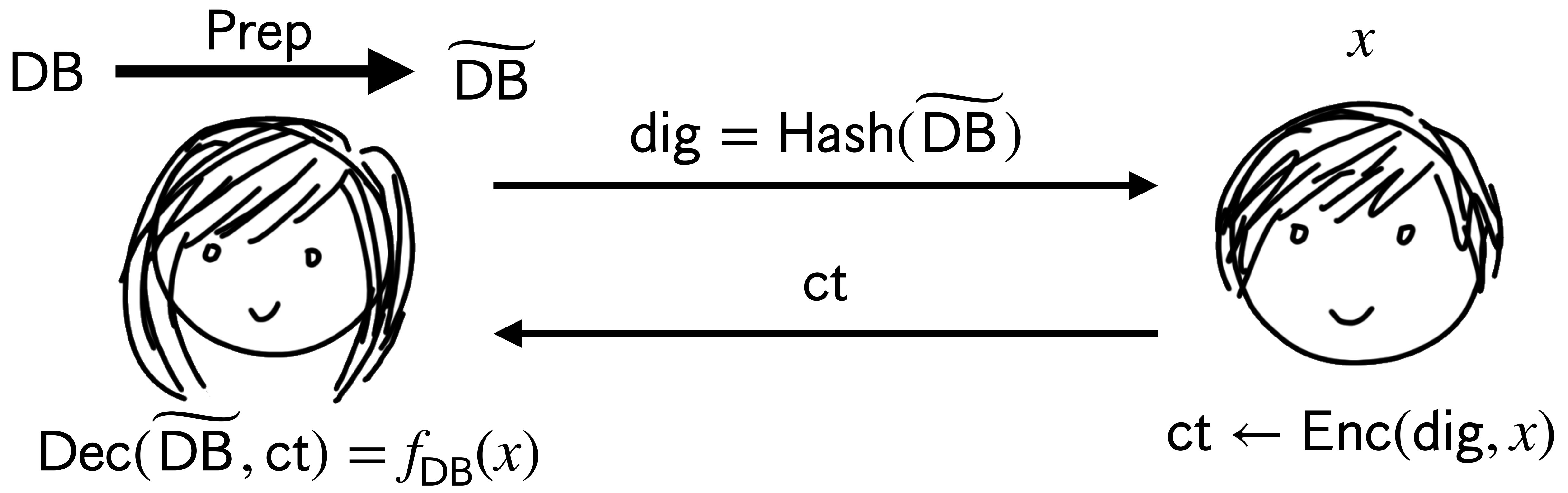
	[DH**M**W'24] (weak efficiency)	[DH**M**W'24] (strong efficiency)
Prep:	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$
Enc:	$\lvert x \rvert + T$	$\lvert x \rvert$
Dec:	T	T
Assumption:		

LFE for RAMs



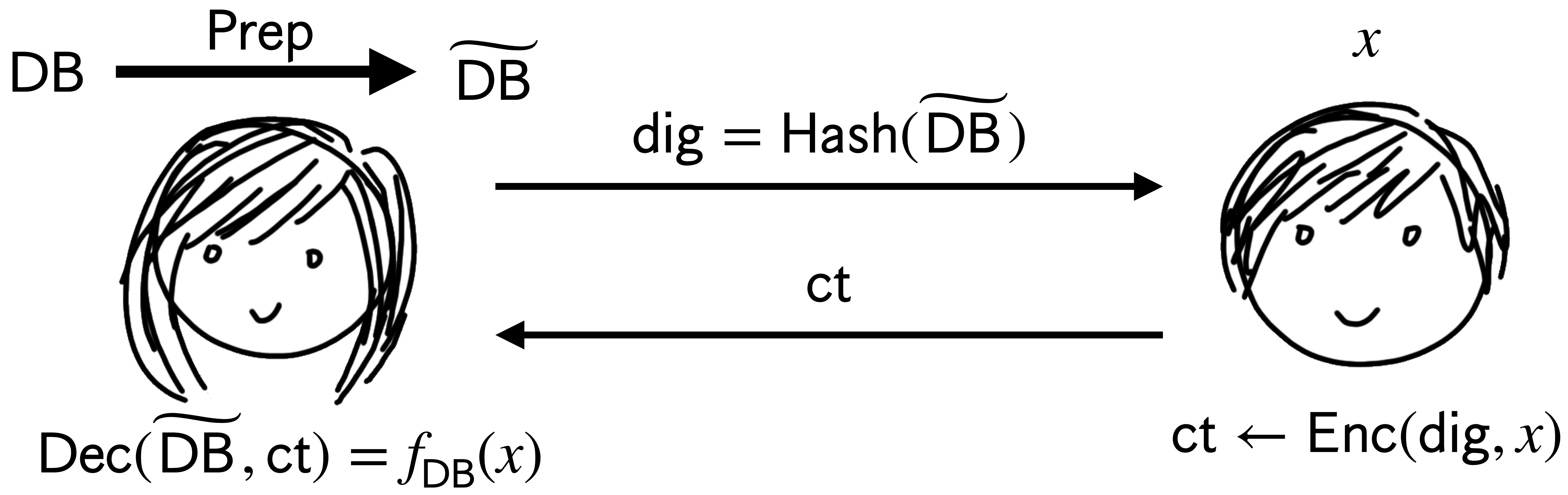
	[DHMW'24] (weak efficiency)	[DHMW'24] (strong efficiency)
Prep:	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$
Enc:	$\lvert x \rvert + T$	$\lvert x \rvert$
Dec:	T	T
Assumption:	RingLWE	RingLWE + iO

LFE for RAMs



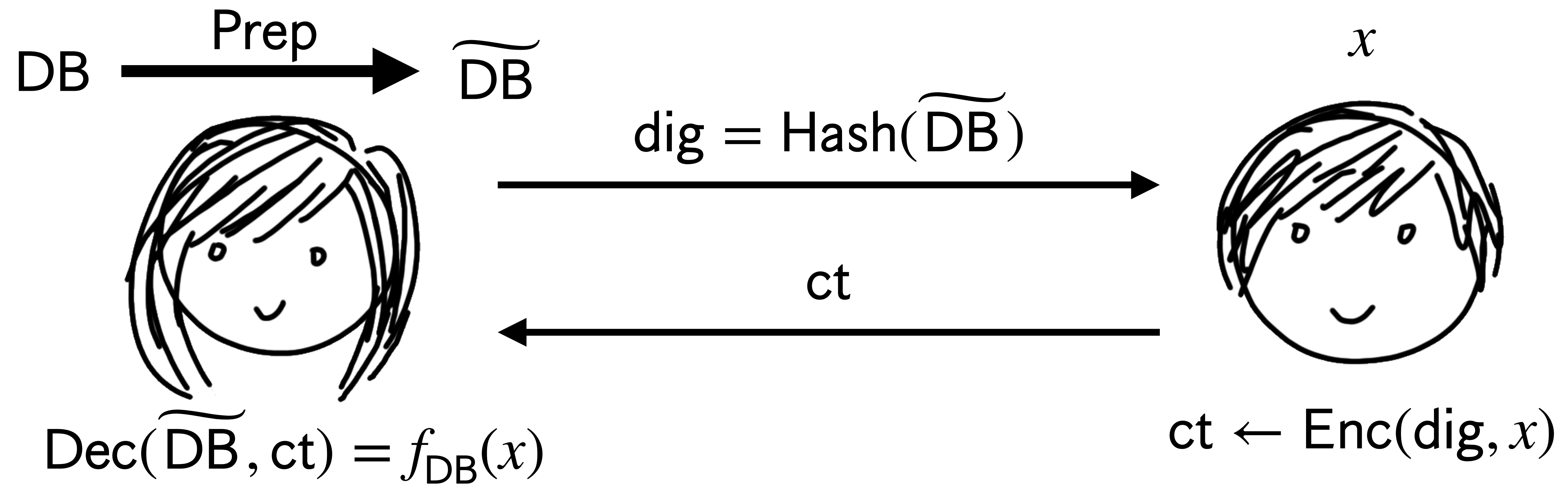
	[DH**M**W'24] (weak efficiency)	[DH**M**W'24] (strong efficiency)	This work
Prep:	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$	
Enc:	$\lvert x \rvert + T$	$\lvert x \rvert$	
Dec:	T	T	
Assumption:	RingLWE	RingLWE + iO	

LFE for RAMs



	[DHMW'24] (weak efficiency)	[DHMW'24] (strong efficiency)	This work
Prep:	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$	$\lvert \mathsf{DB} \rvert^{1+\varepsilon}$	$s^{1+o(1)}$
Enc:	$\lvert x \rvert + T$	$\lvert x \rvert$	
Dec:	T	T	
Assumption:	RingLWE	RingLWE + iO	

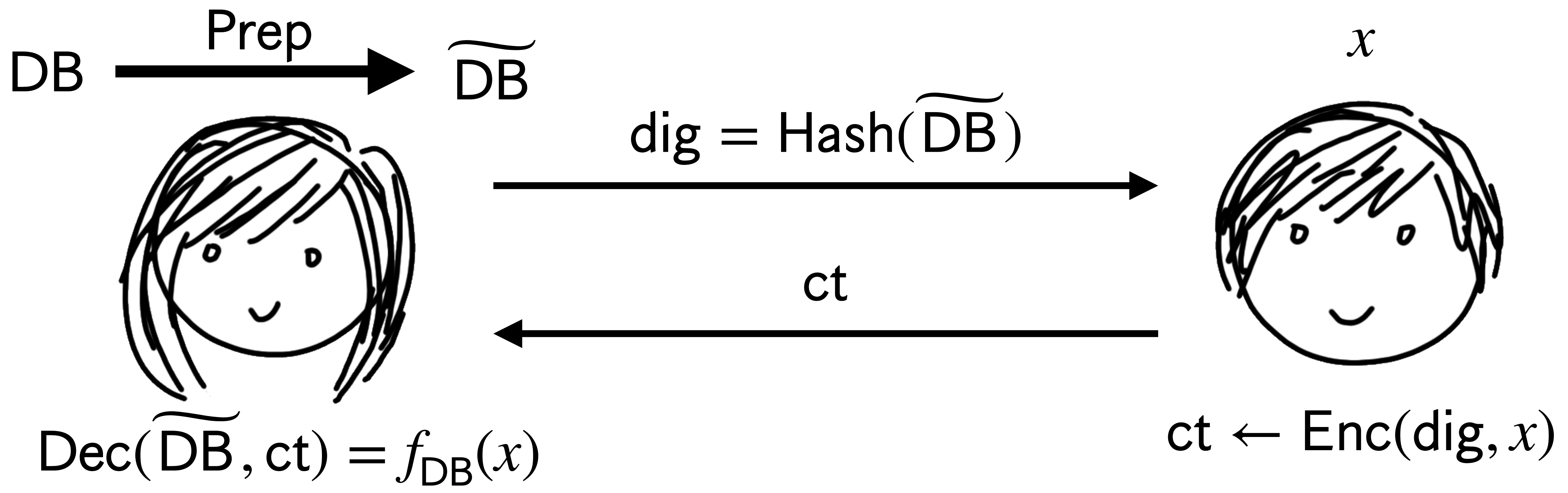
LFE for RAMs



s = ***circuit size*** of $f_{\text{DB}} \approx T \cdot |\text{DB}|$

	[DH**M**W'24] (weak efficiency)	[DH**M**W'24] (strong efficiency)	This work
Prep:	$\|\text{DB}\|^{1+\varepsilon}$	$\|\text{DB}\|^{1+\varepsilon}$	$s^{1+o(1)}$
Enc:	$\|x\| + T$	$\|x\|$	
Dec:	T	T	
Assumption:	RingLWE	RingLWE + iO	

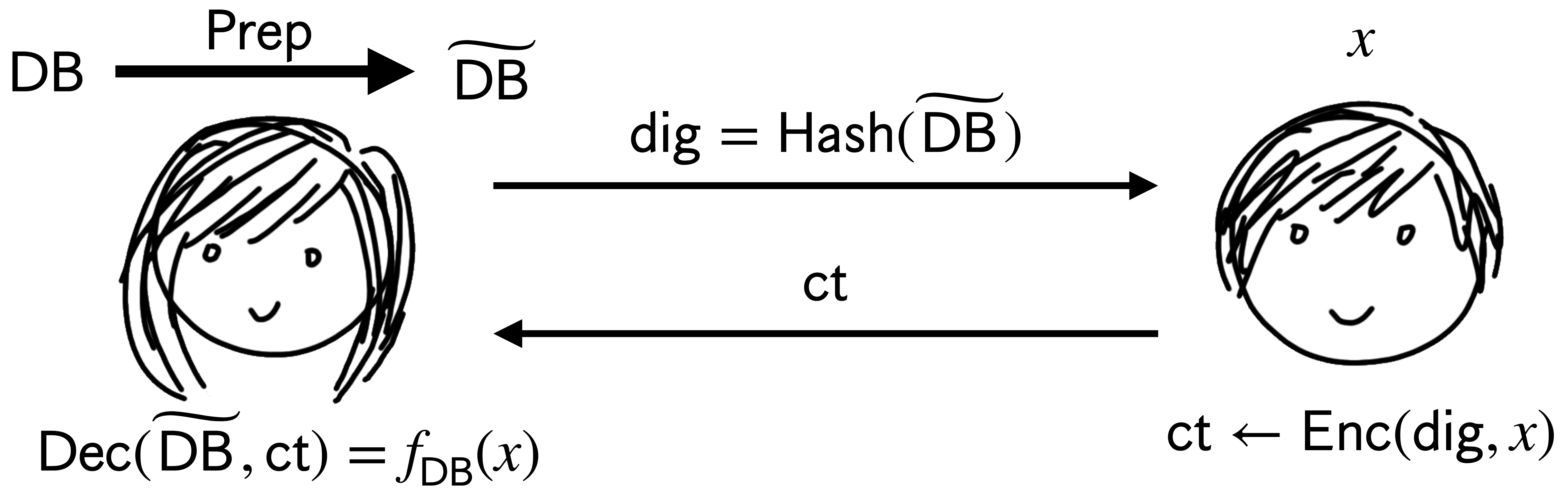
LFE for RAMs



s = ***circuit size*** of $f_{\text{DB}} \approx T \cdot |\text{DB}|$

	[DH**M**W'24] (weak efficiency)	[DH**M**W'24] (strong efficiency)	This work
Prep:	$\|\text{DB}\|^{1+\varepsilon}$	$\|\text{DB}\|^{1+\varepsilon}$	$s^{1+o(1)}$
Enc:	$\|x\| + T$	$\|x\|$	$\|x\|^{1+o(1)}$
Dec:	T	T	$T^{1+o(1)}$
Assumption:	RingLWE	RingLWE + iO	RingLWE (+ circular security)

LFE for RAMs



s = ***circuit size*** of $f_{\text{DB}} \approx T \cdot |\text{DB}|$

	[DHMW'24] (weak efficiency)	[DHMW'24] (strong efficiency)	This work
Prep:	$\|\text{DB}\|^{1+\varepsilon}$	$\|\text{DB}\|^{1+\varepsilon}$	$s^{1+o(1)}$
Enc:	$\|x\| + T$	$\|x\|$	$\|x\|^{1+o(1)}$
Dec:	T	T	$T^{1+o(1)}$
Assumption:	RingLWE	RingLWE + iO	RingLWE (+ circular security)

About as efficient online, worse offline

LFE Construction Template

[Quach-Wee-Wichs'17]

Attribute-based LFE

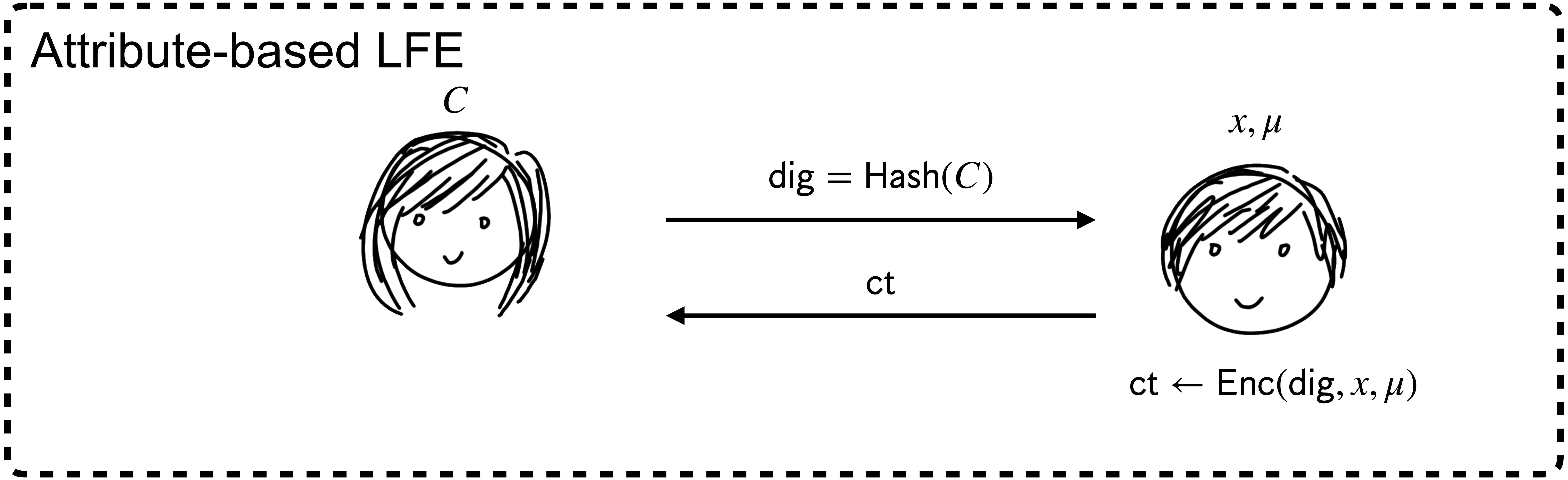
LFE Construction Template

[Quach-Wee-Wichs'17]

Attribute-based LFE

LFE Construction Template

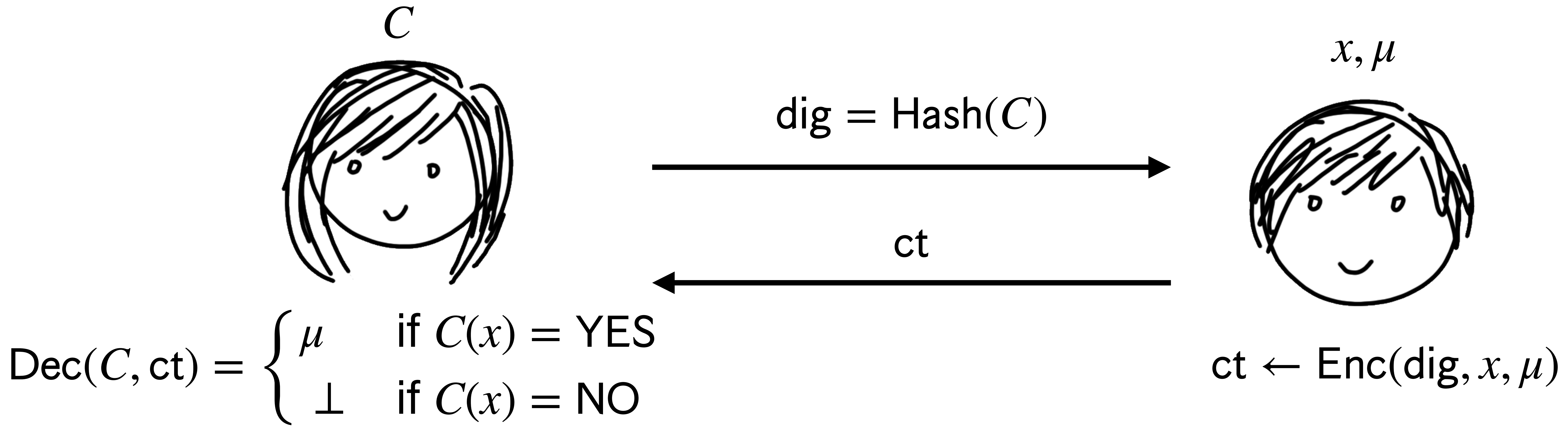
[Quach-Wee-Wichs'17]



LFE Construction Template

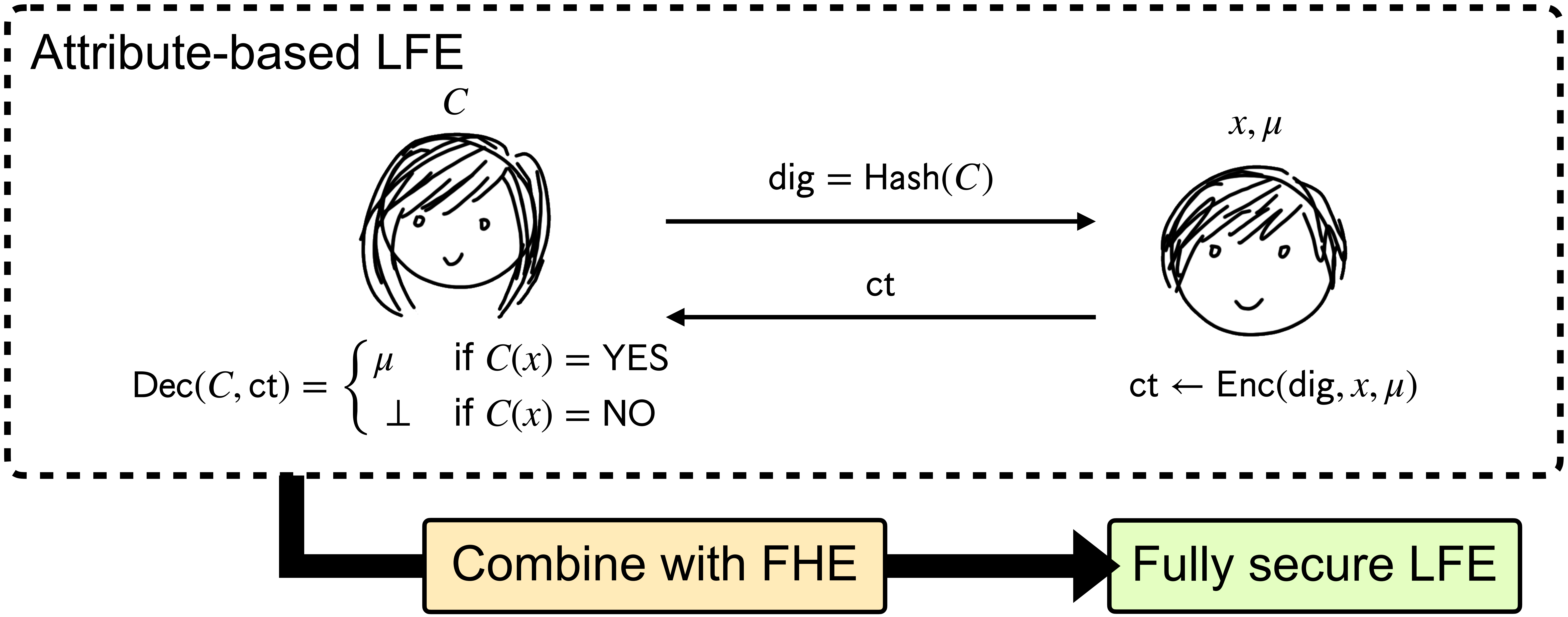
[Quach-Wee-Wichs'17]

Attribute-based LFE



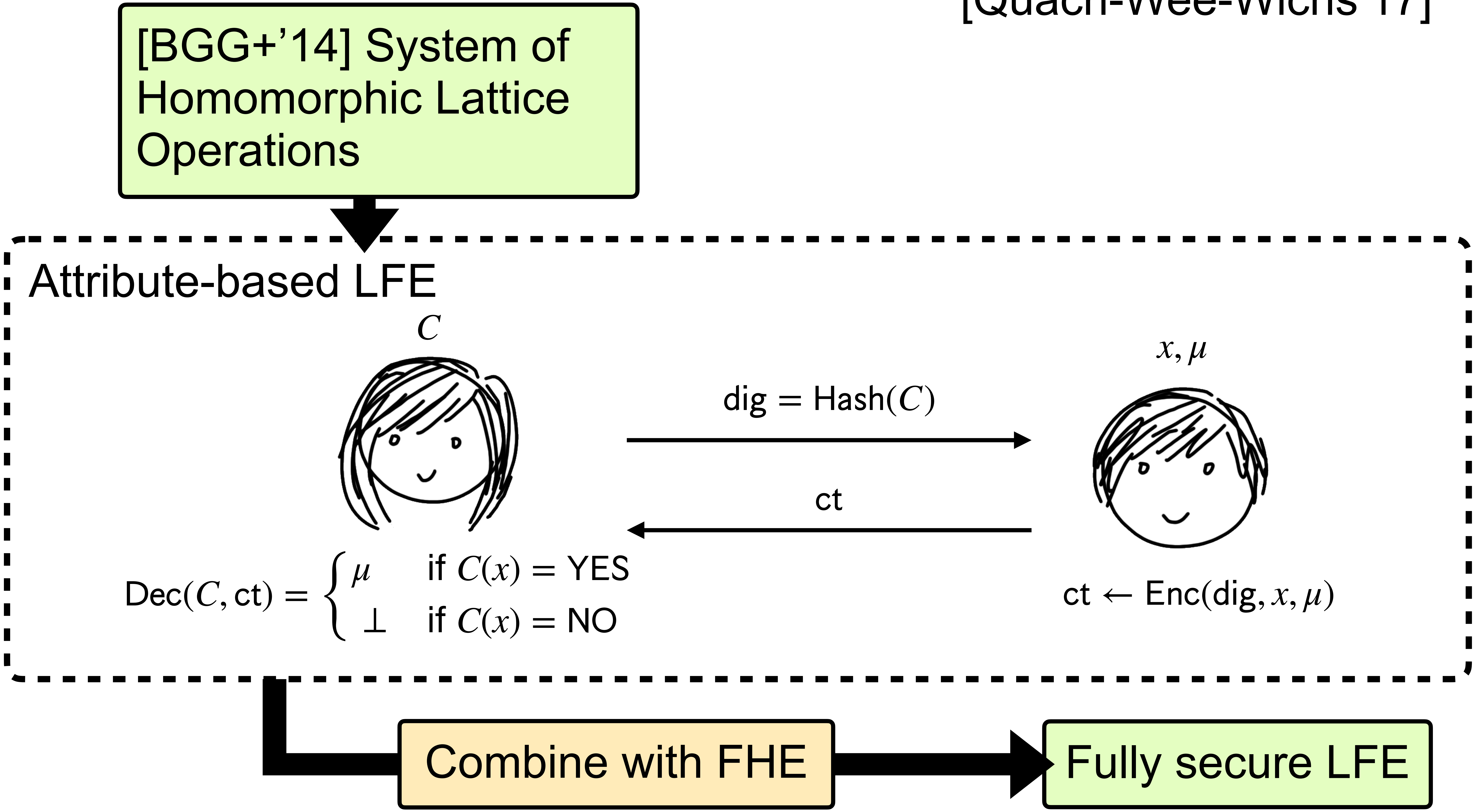
LFE Construction Template

[Quach-Wee-Wichs'17]

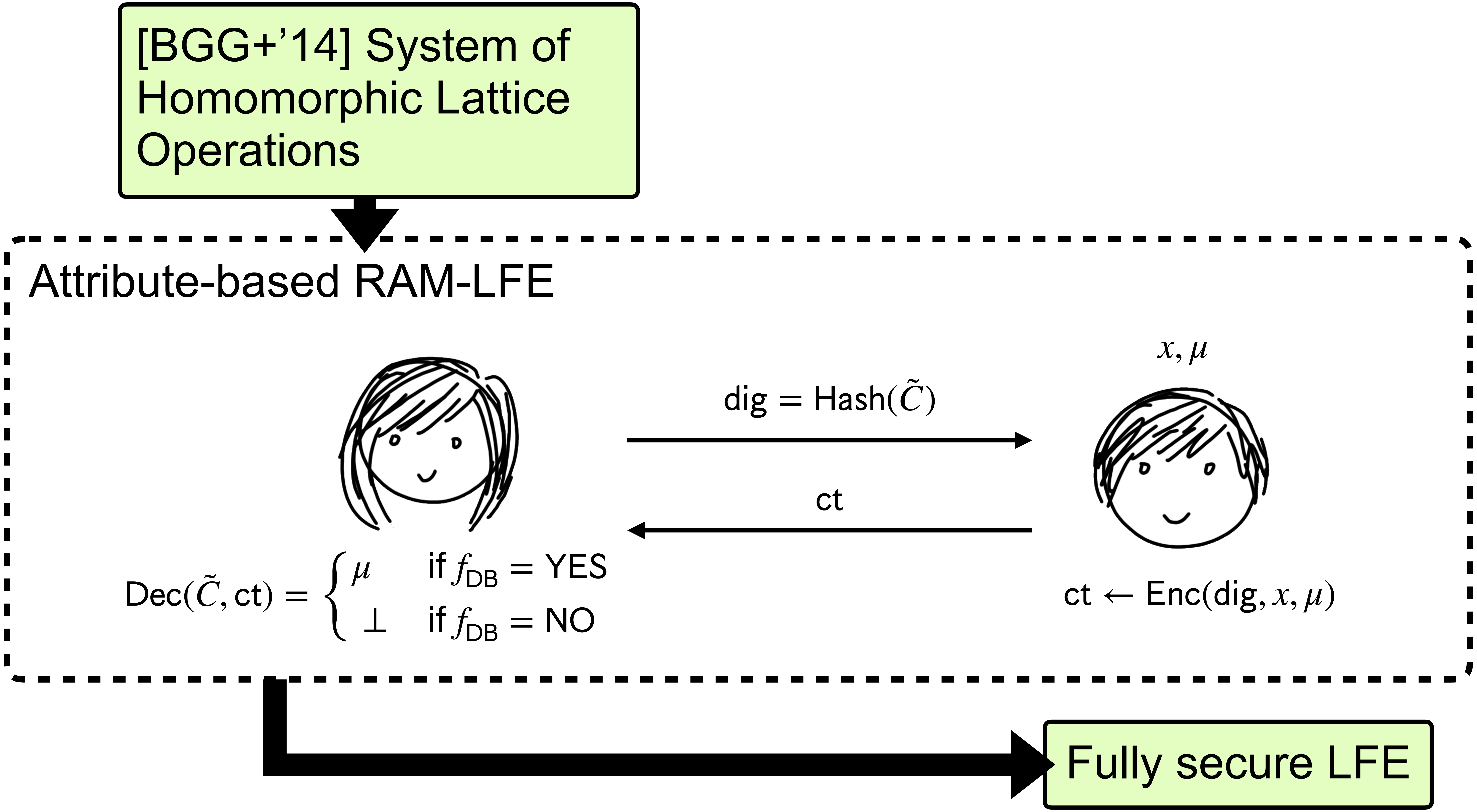


LFE Construction Template

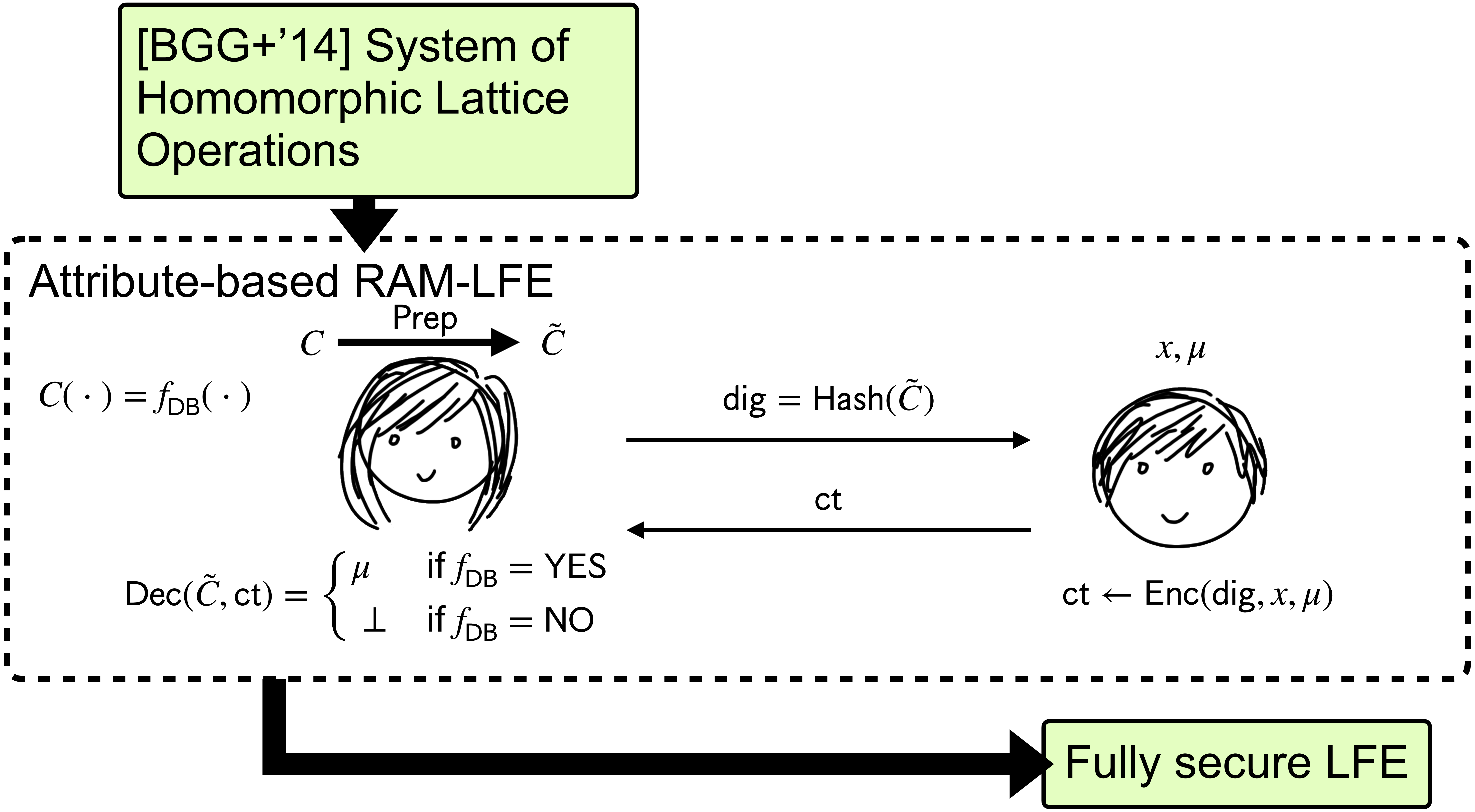
[Quach-Wee-Wichs'17]



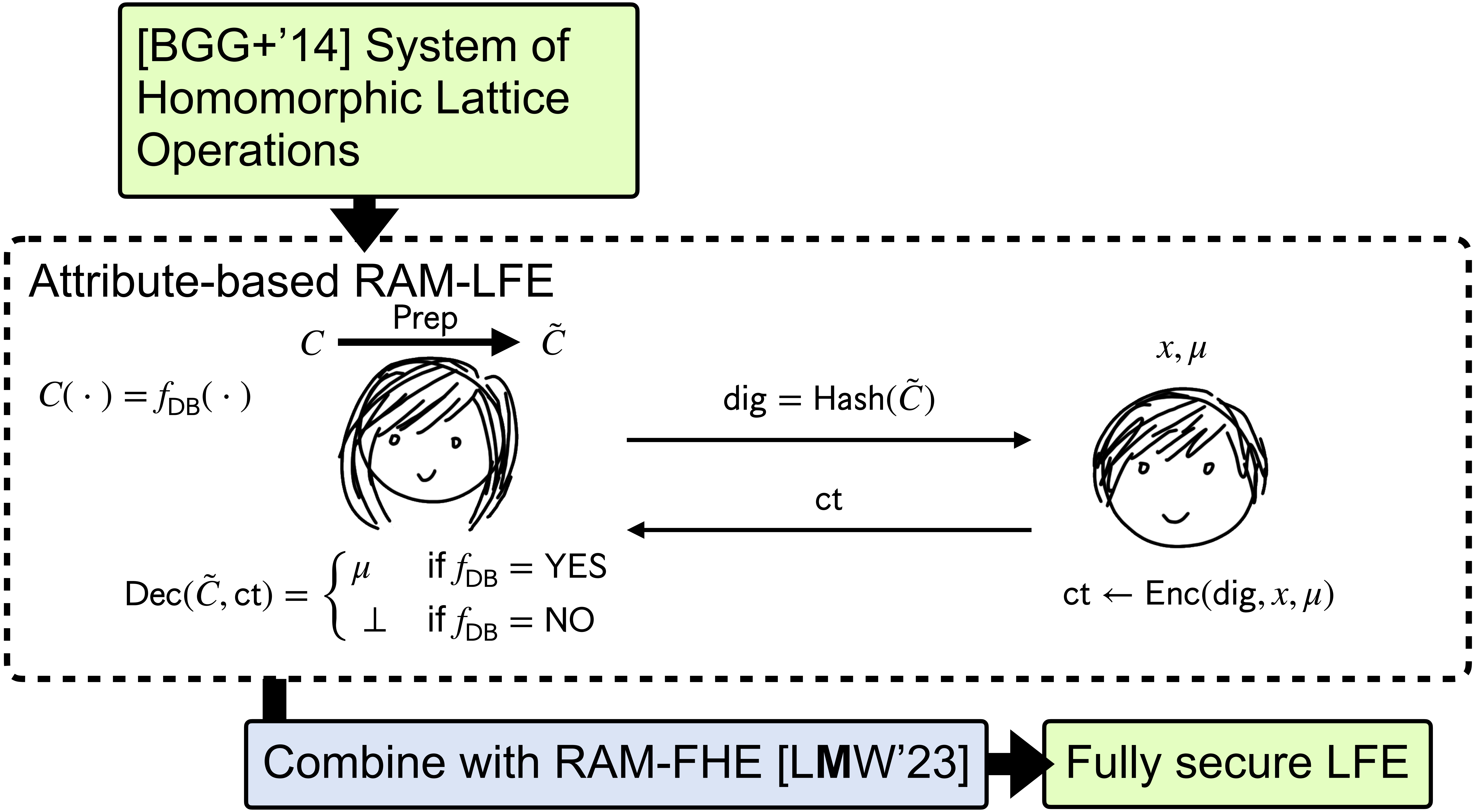
LFE Construction Template



LFE Construction Template



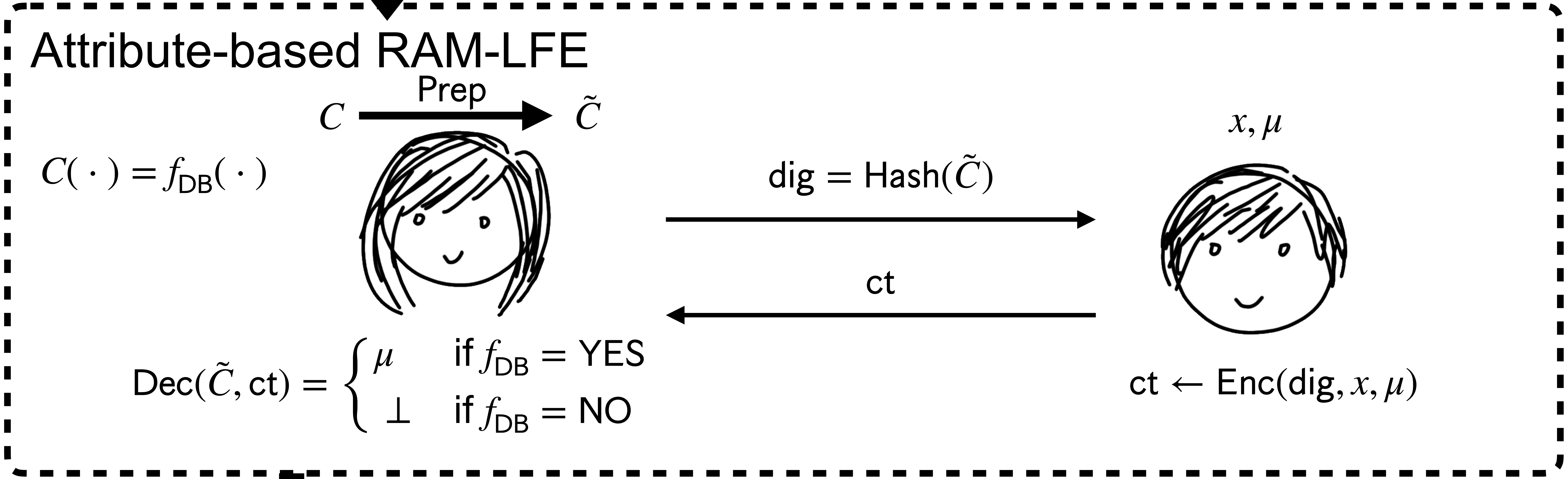
LFE Construction Template



LFE Construction Template

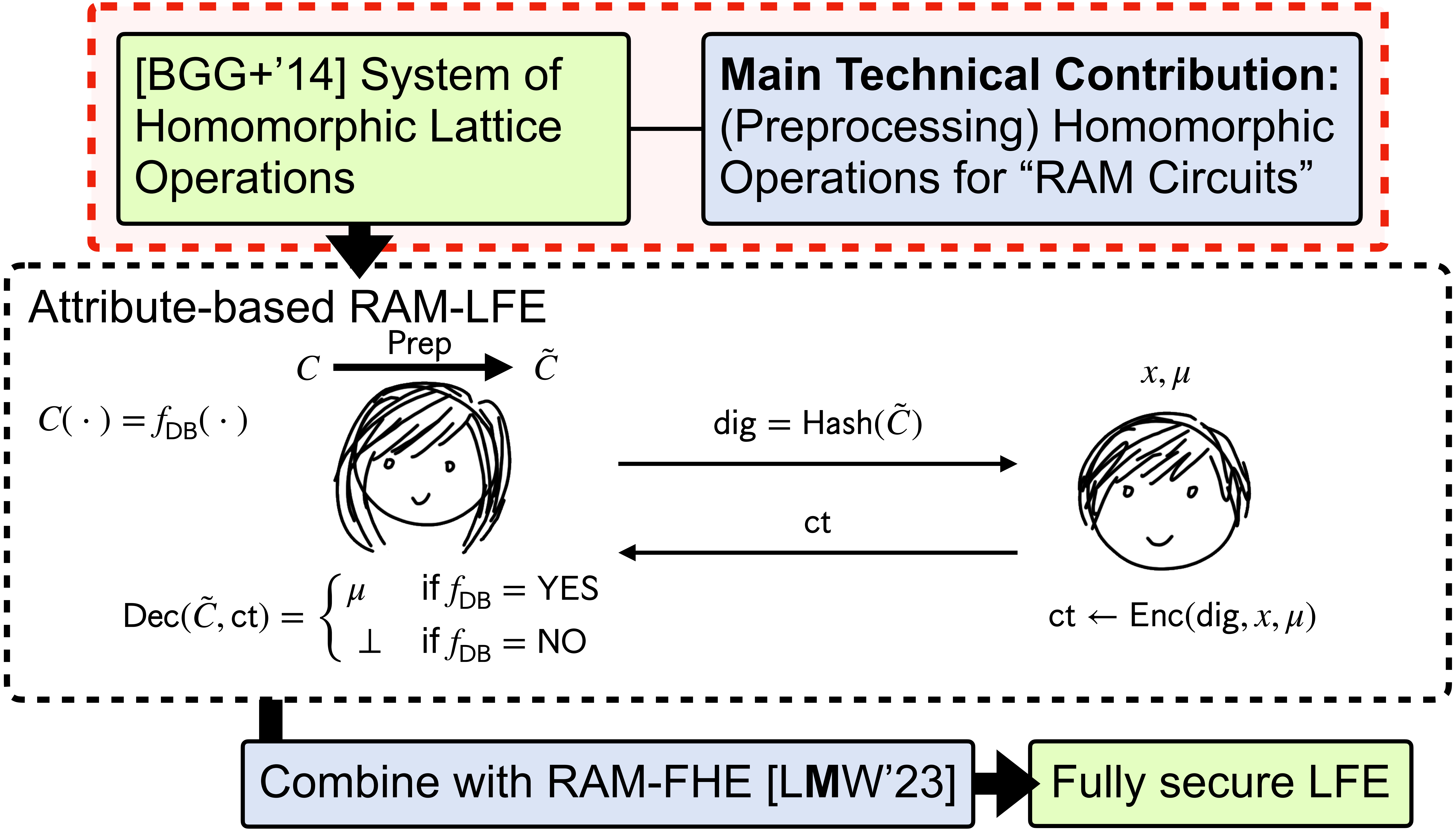
[BGG+'14] System of Homomorphic Lattice Operations

Main Technical Contribution: (Preprocessing) Homomorphic Operations for "RAM Circuits"



Combine with RAM-FHE [L**M**W'23] → Fully secure LFE

LFE Construction Template



Homomorphic Operations

[BGG+'14]

Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$

Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$

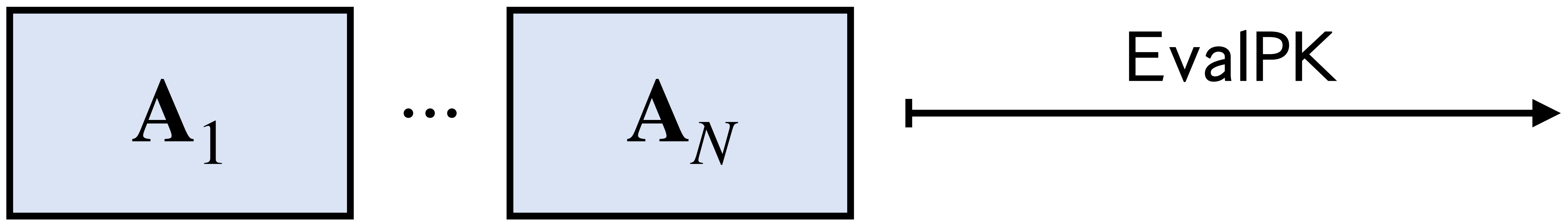
$$\mathbf{A}_1 \quad \cdots \quad \mathbf{A}_N$$

Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$

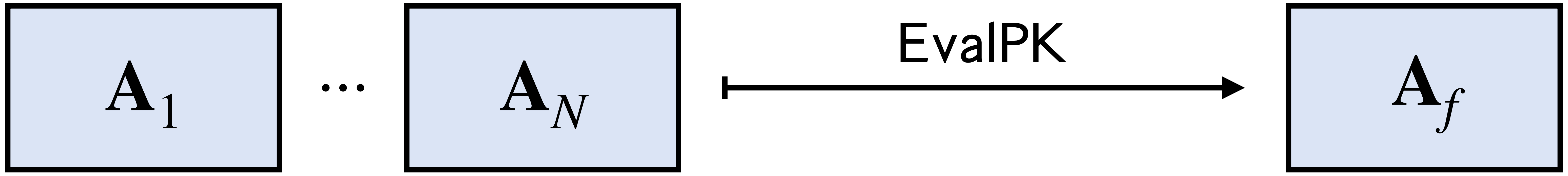


Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$

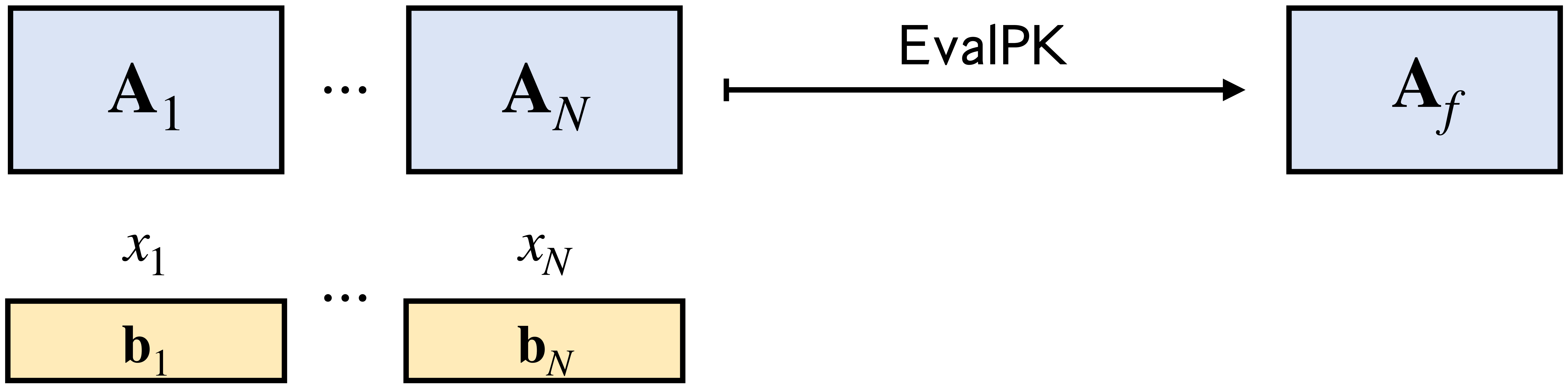


Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f: \{0,1\}^N \to \{0,1\}$

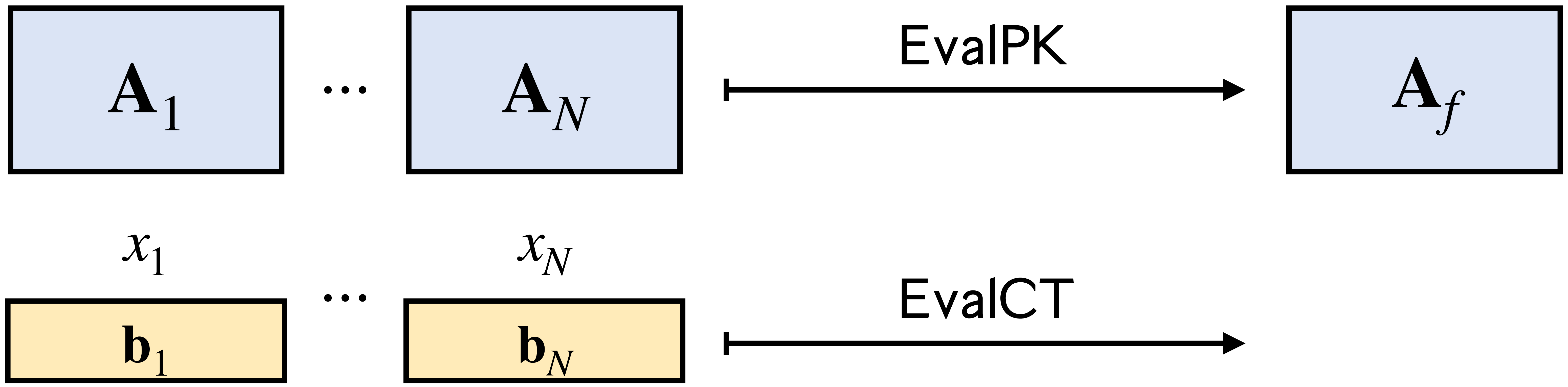


Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$

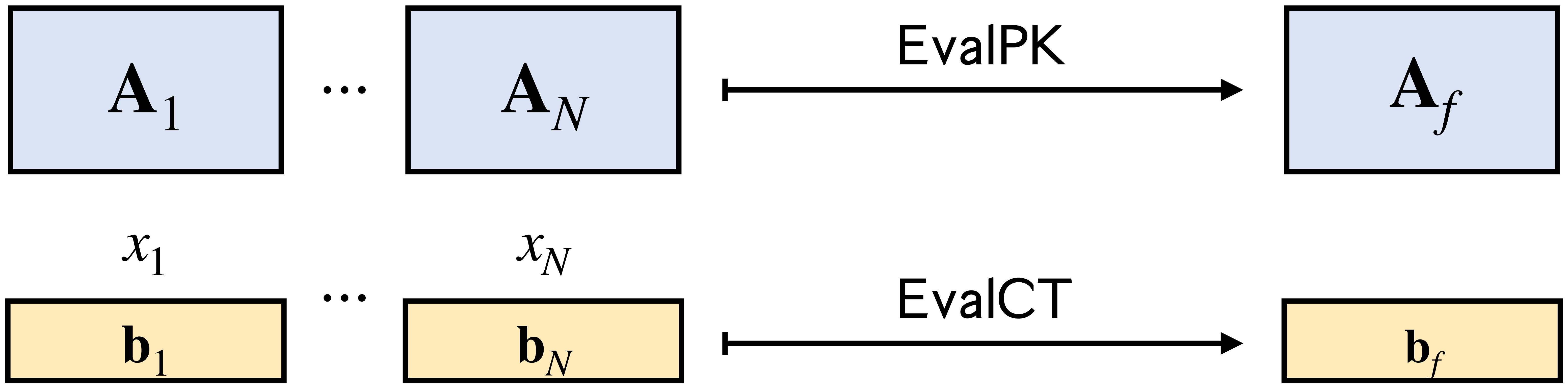


Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$

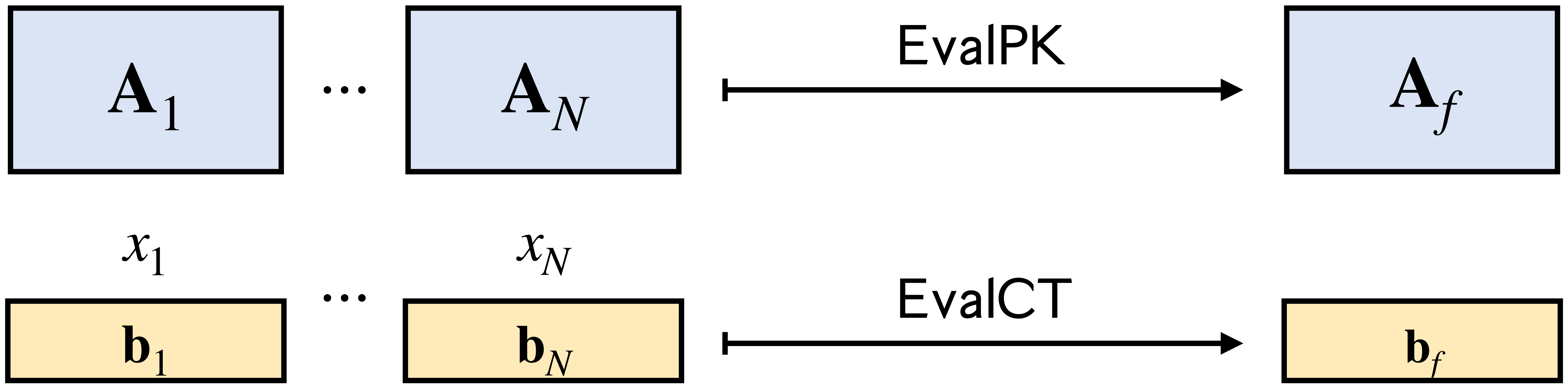


Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$



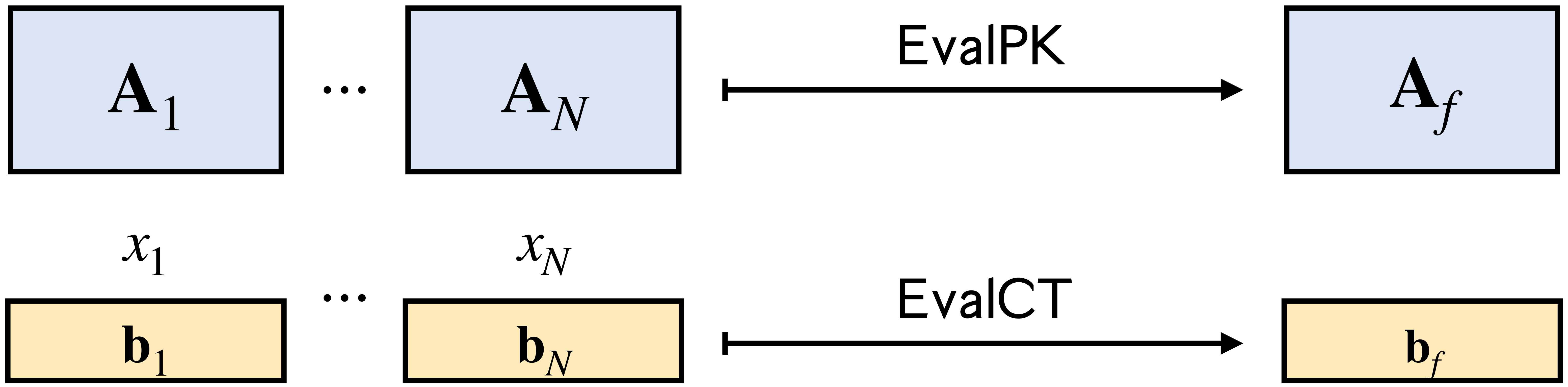
$\mathbf{b}_i \approx \mathbf{s}^\top(\mathbf{A}_i - x_i\mathbf{G})$

Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f: \{0,1\}^N \to \{0,1\}$



$\mathbf{b}_i \approx \mathbf{s}^\top(\mathbf{A}_i - x_i\mathbf{G})$

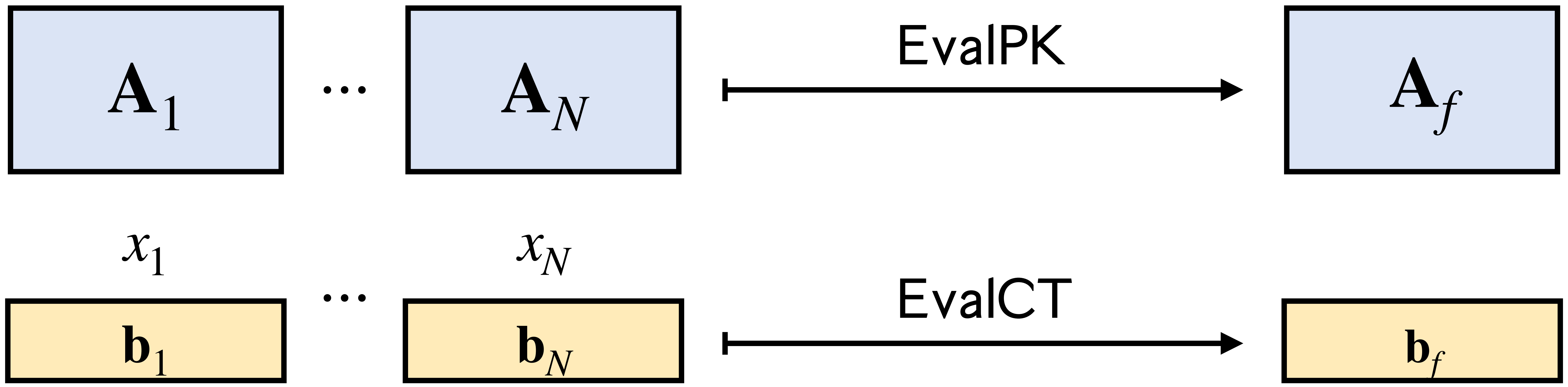
$\mathbf{b}_f \approx \mathbf{s}^\top(\mathbf{A}_f - f(x)\mathbf{G})$

Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$



Error grows with depth of f

$\mathbf{b}_i \approx \mathbf{s}^\top(\mathbf{A}_i - x_i\mathbf{G})$

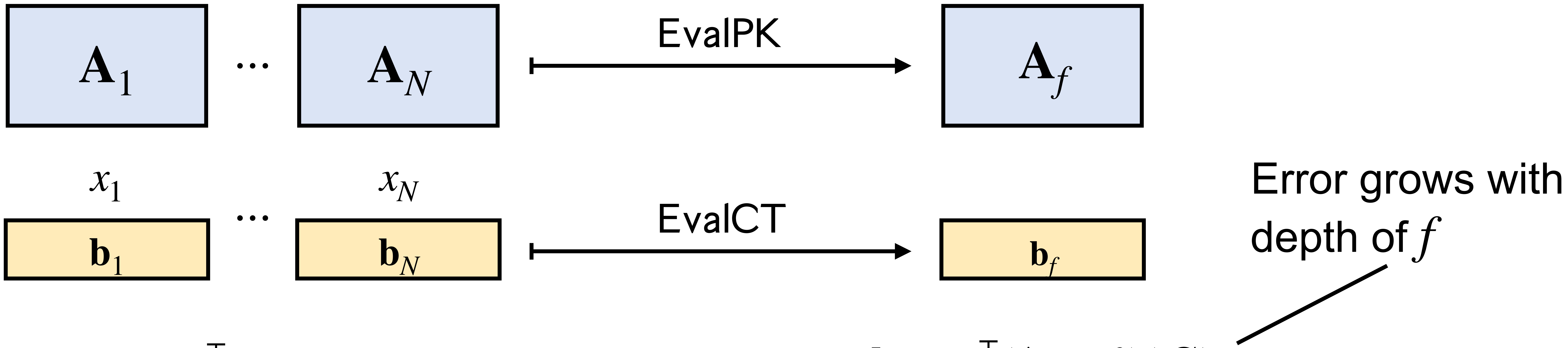
$\mathbf{b}_f \approx \mathbf{s}^\top(\mathbf{A}_f - f(x)\mathbf{G})$

Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$



$\mathbf{b}_i \approx \mathbf{s}^\top(\mathbf{A}_i - x_i\mathbf{G})$

$\mathbf{b}_f \approx \mathbf{s}^\top(\mathbf{A}_f - f(x)\mathbf{G})$

[BGG+'14]: EvalPK, EvalCT run in time proportional to the circuit size of f

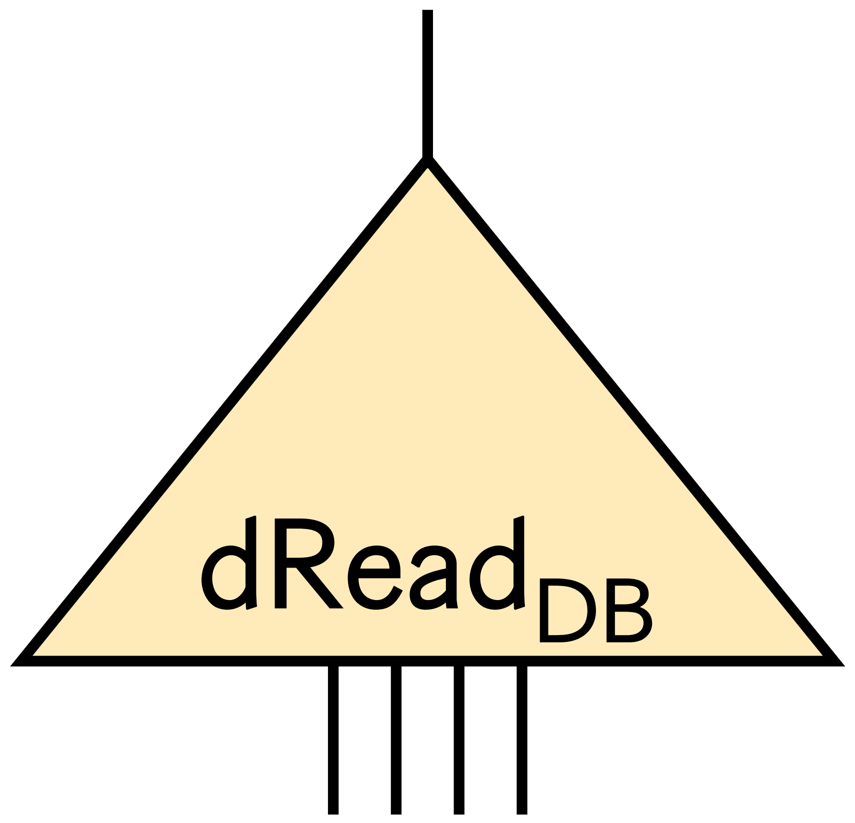
RAM Circuits

Boolean circuits + two new gates:

RAM Circuits

Boolean circuits + two new gates:

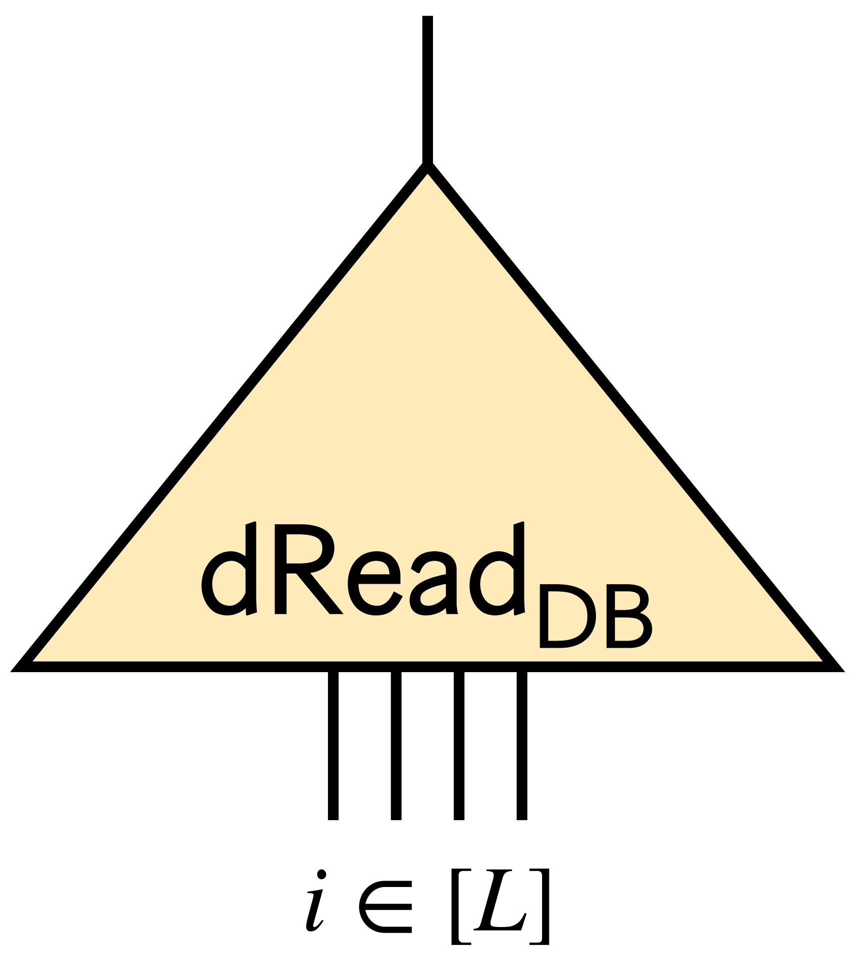
Data-Read Gates: For any DB $\in \{0,1\}^L$



RAM Circuits

Boolean circuits + two new gates:

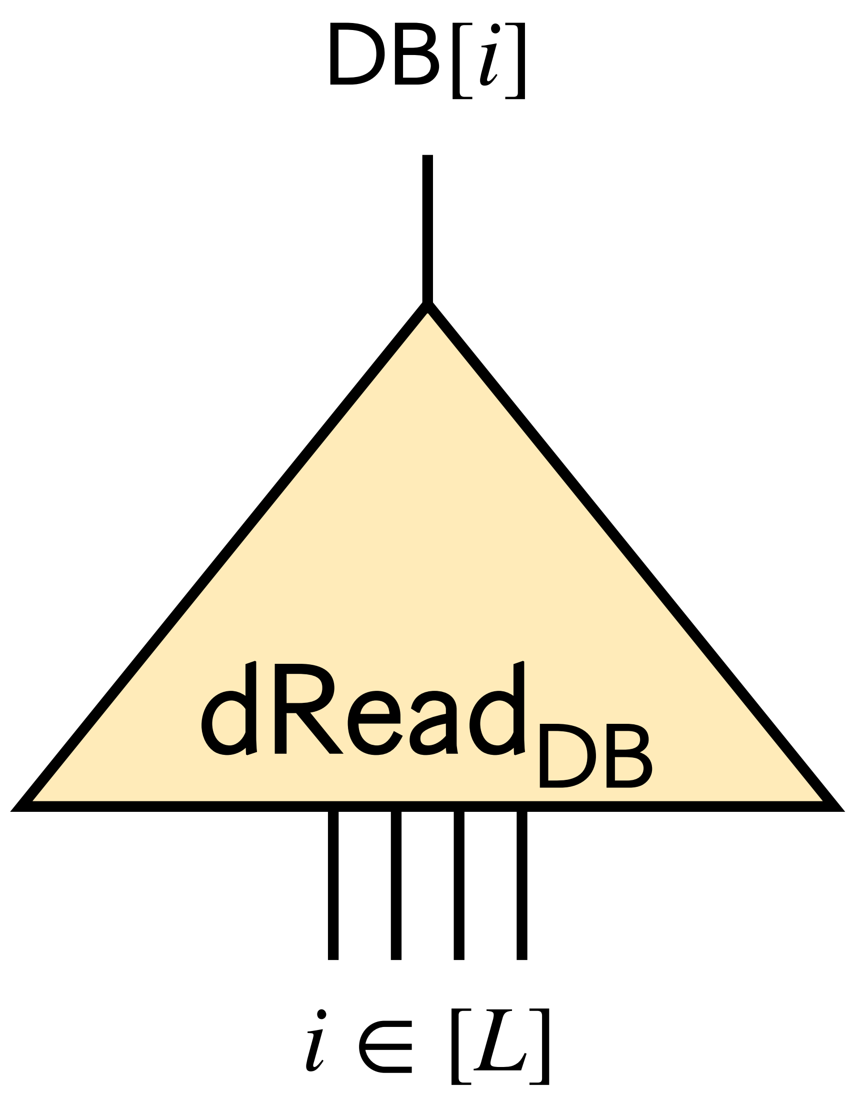
Data-Read Gates: For any DB $\in \{0,1\}^L$



RAM Circuits

Boolean circuits + two new gates:

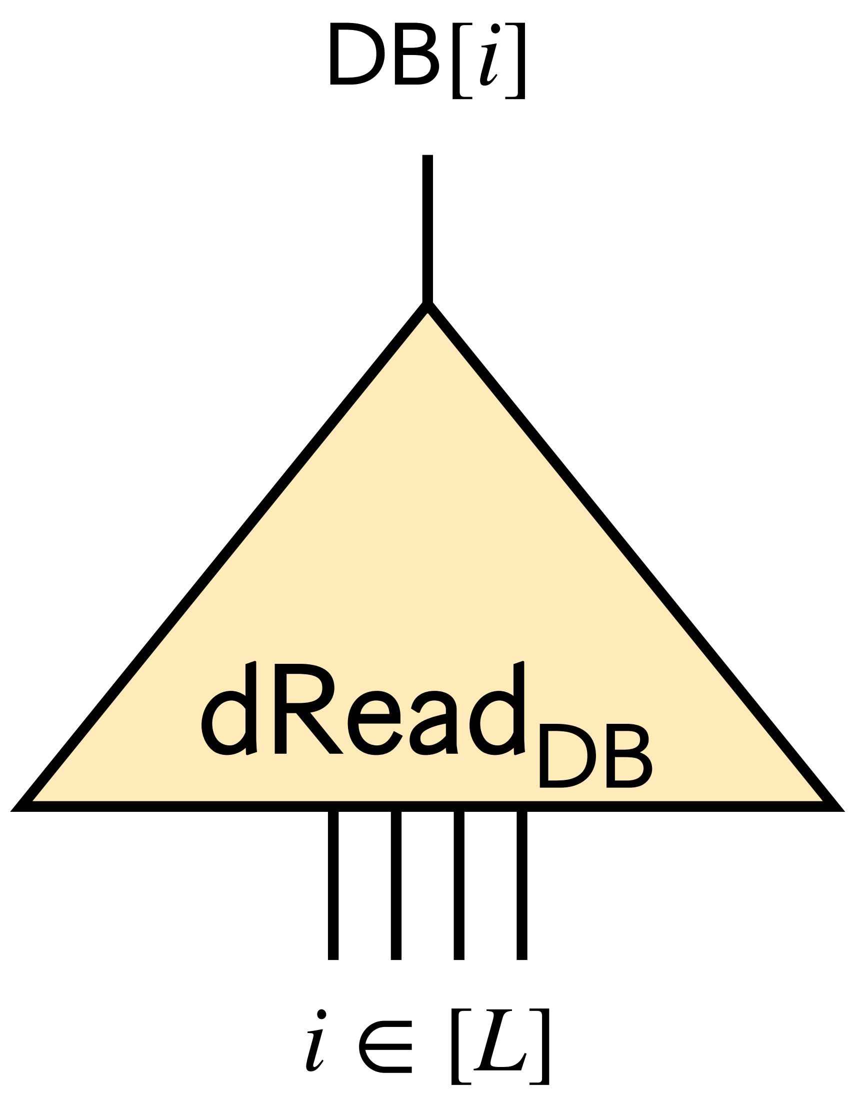
Data-Read Gates: For any DB $\in \{0,1\}^L$



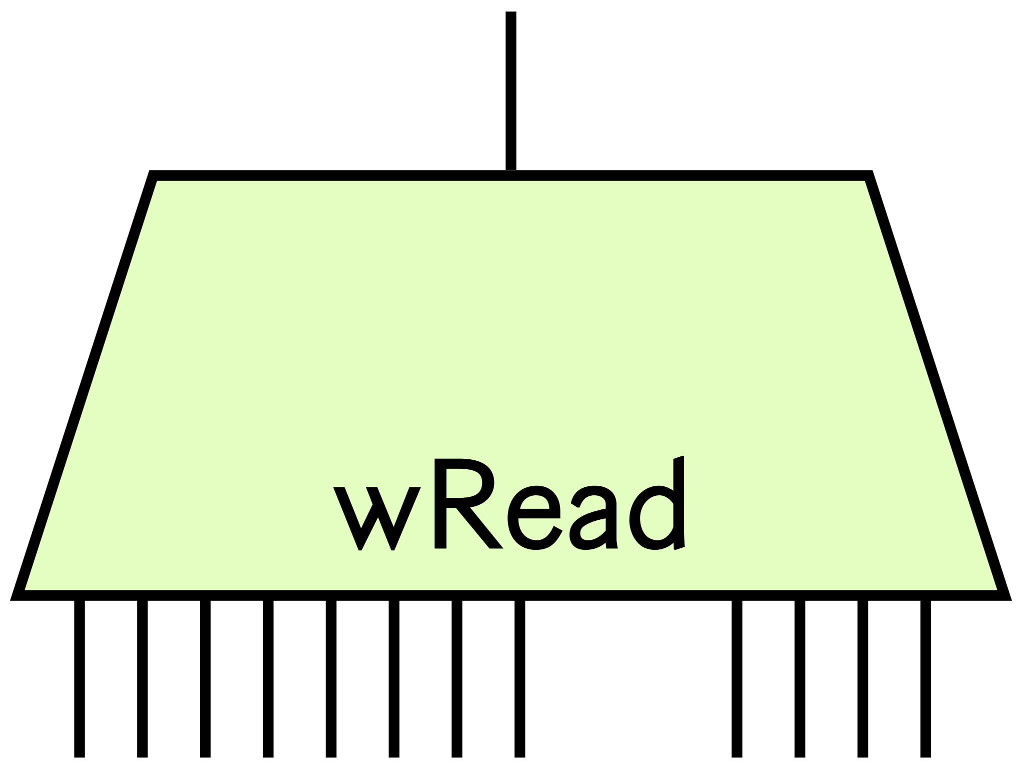
RAM Circuits

Boolean circuits + two new gates:

Data-Read Gates: For any DB $\in \{0,1\}^L$



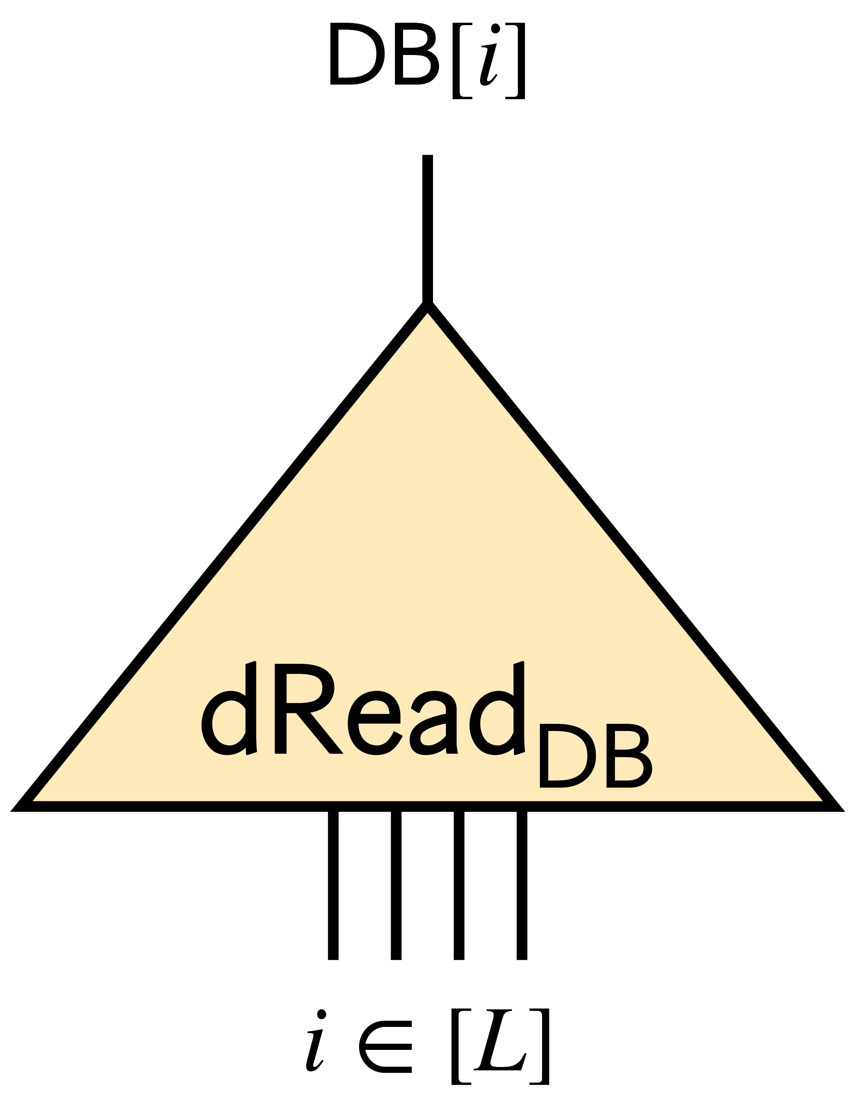
Wire-Read Gates:



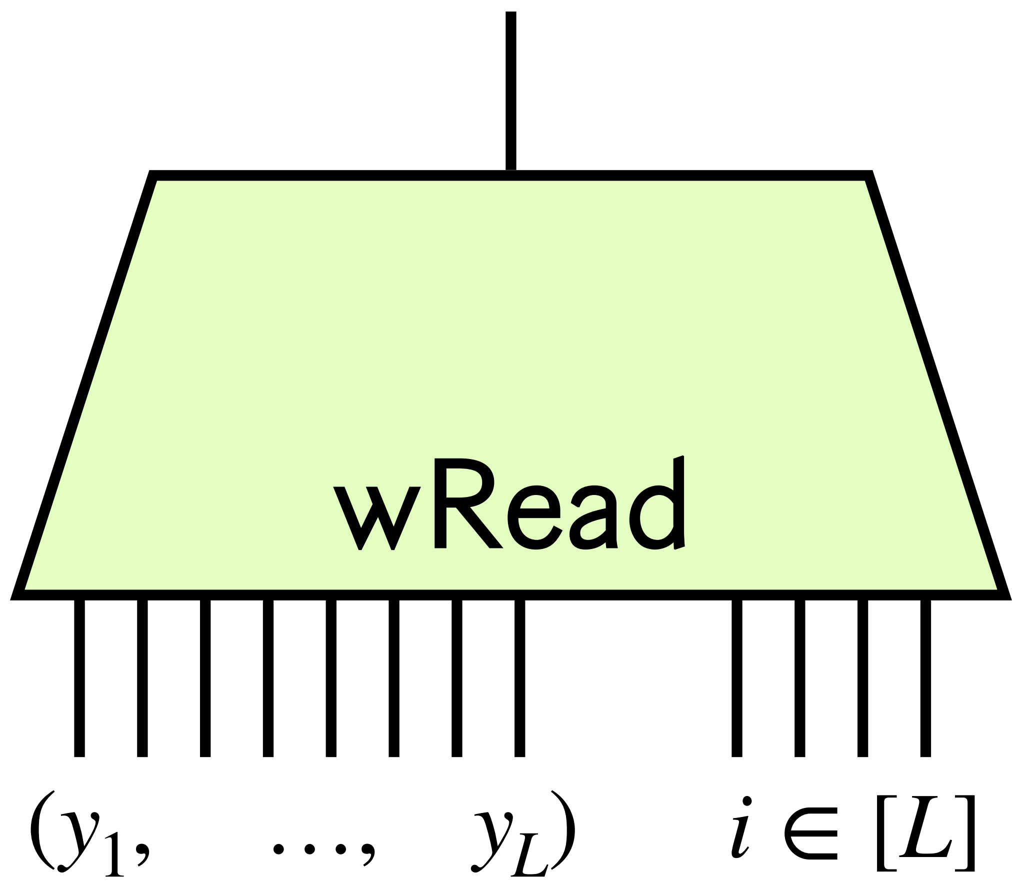
RAM Circuits

Boolean circuits + two new gates:

Data-Read Gates: For any DB $\in \{0,1\}^L$



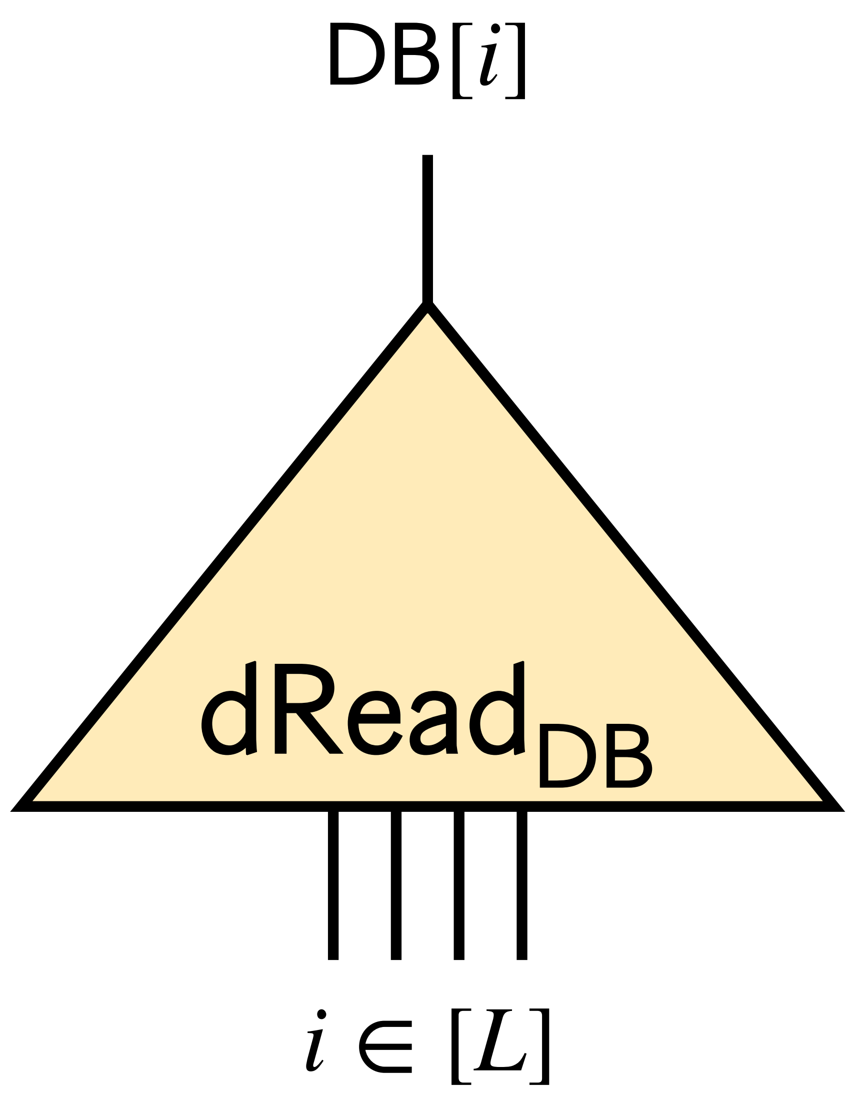
Wire-Read Gates:



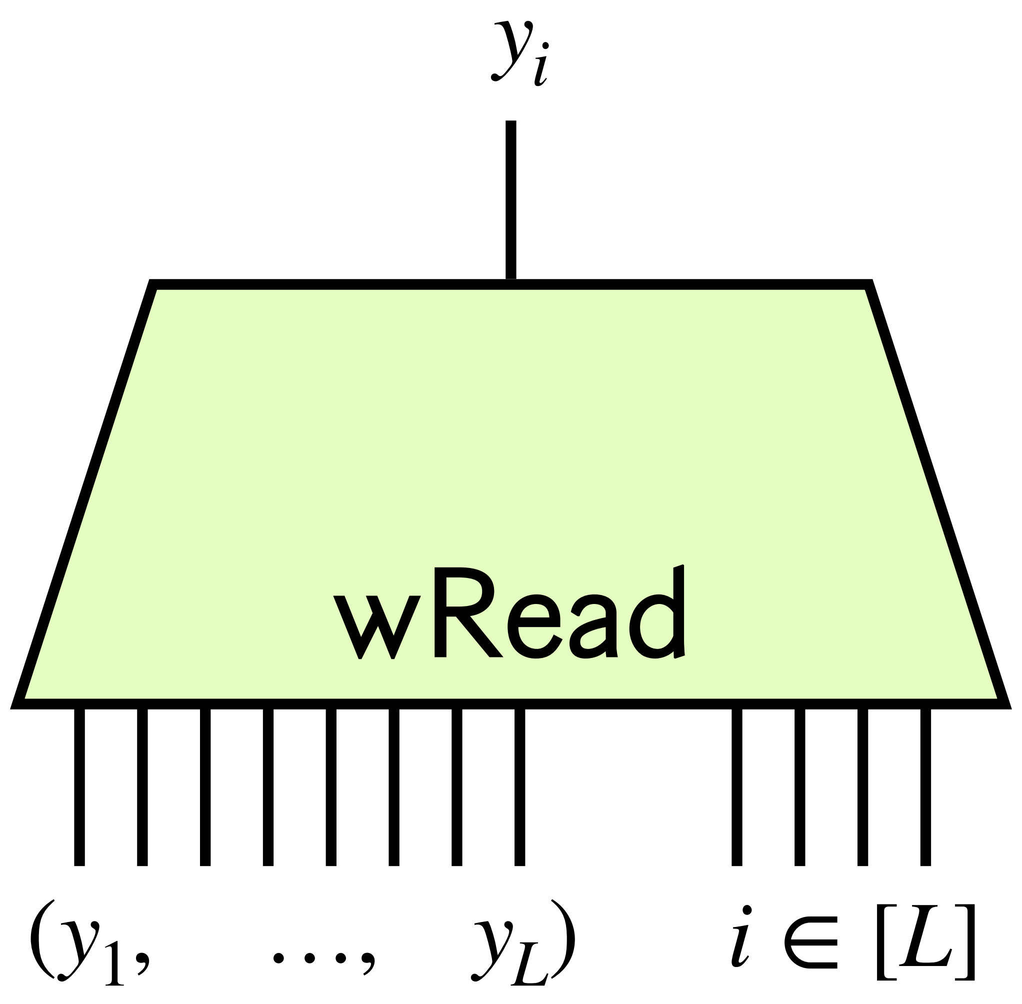
RAM Circuits

Boolean circuits + two new gates:

Data-Read Gates: For any DB $\in \{0,1\}^L$



Wire-Read Gates:

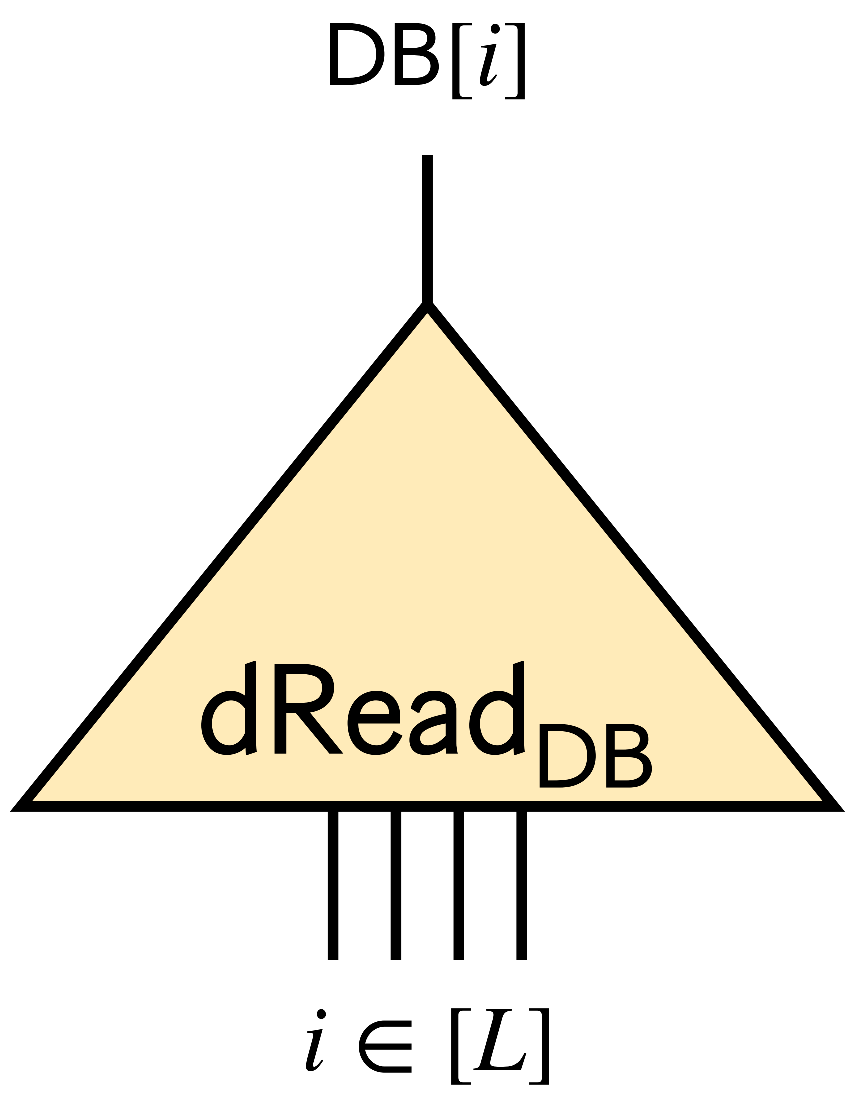


RAM Circuits

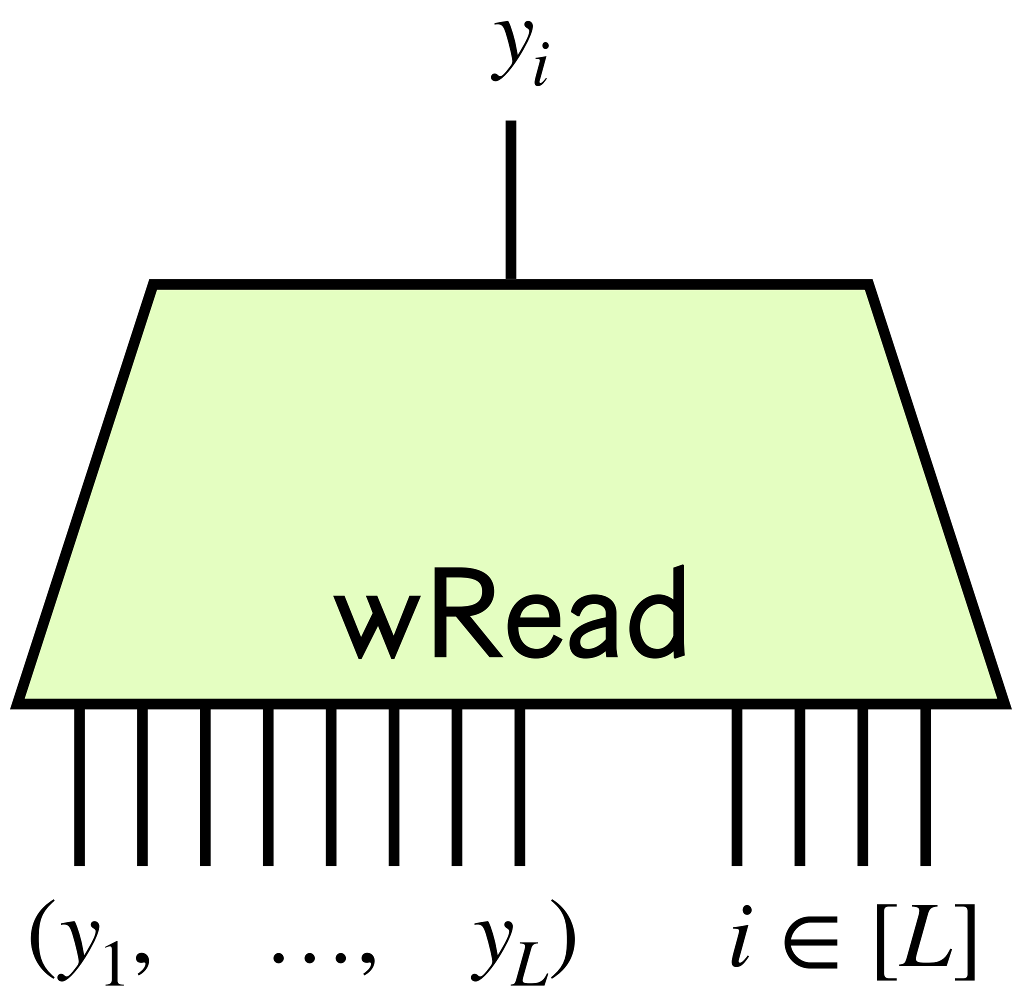
Boolean circuits + two new gates:

Data-Read Gates: For any DB $\in \{0,1\}^L$

Efficient access to a hardcoded database



Wire-Read Gates:

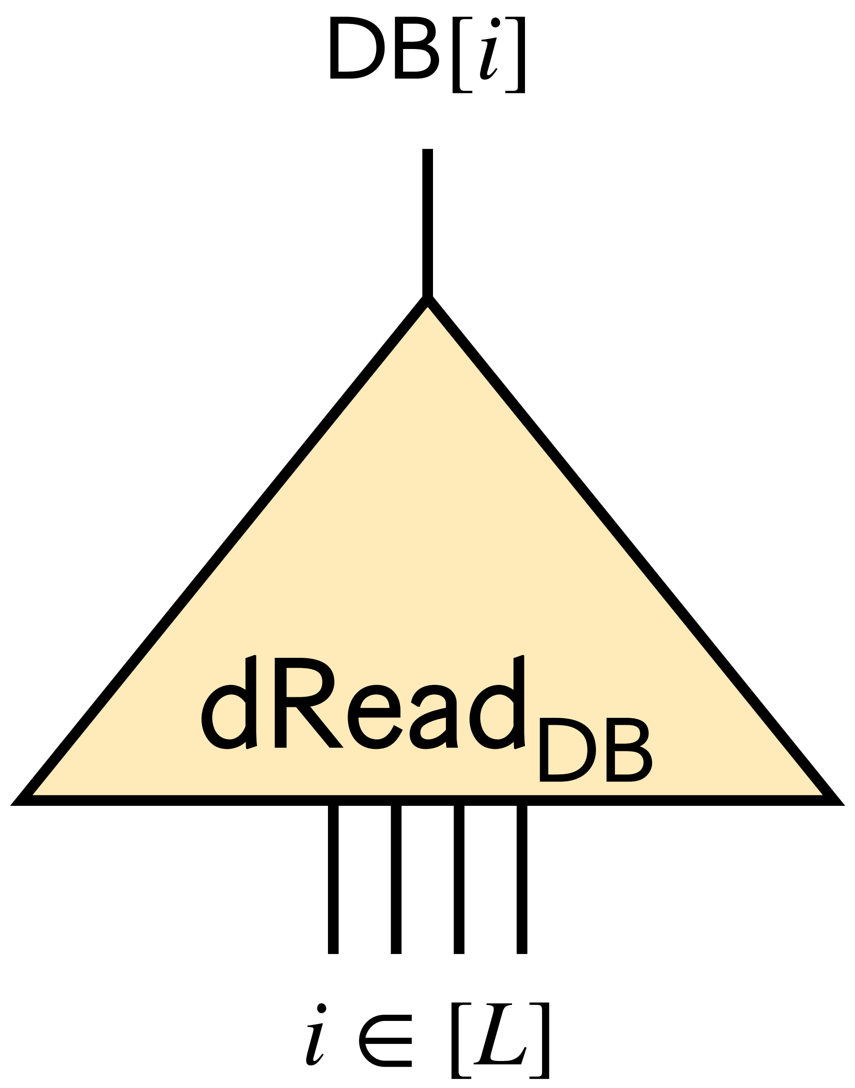


RAM Circuits

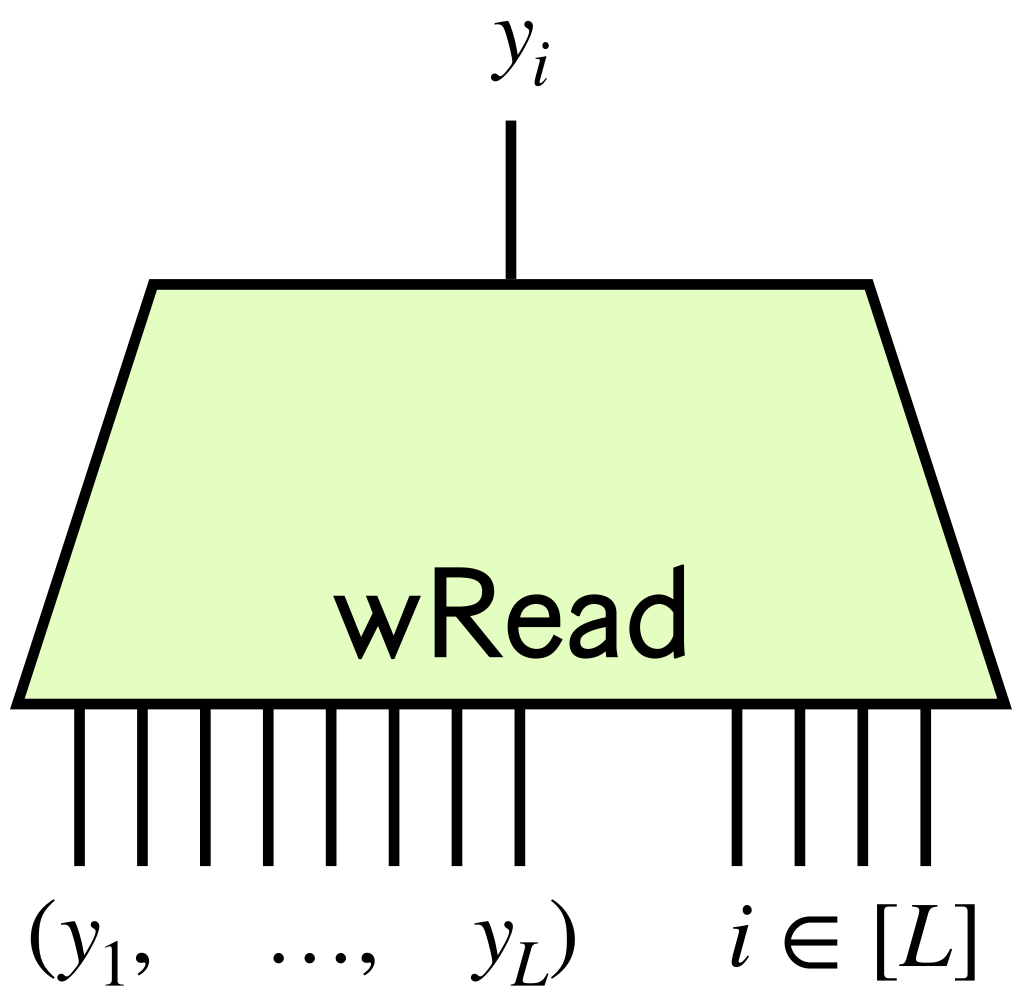
Boolean circuits + two new gates:

Data-Read Gates: For any DB $\in \{0,1\}^L$

Efficient access to a hardcoded database



Wire-Read Gates:



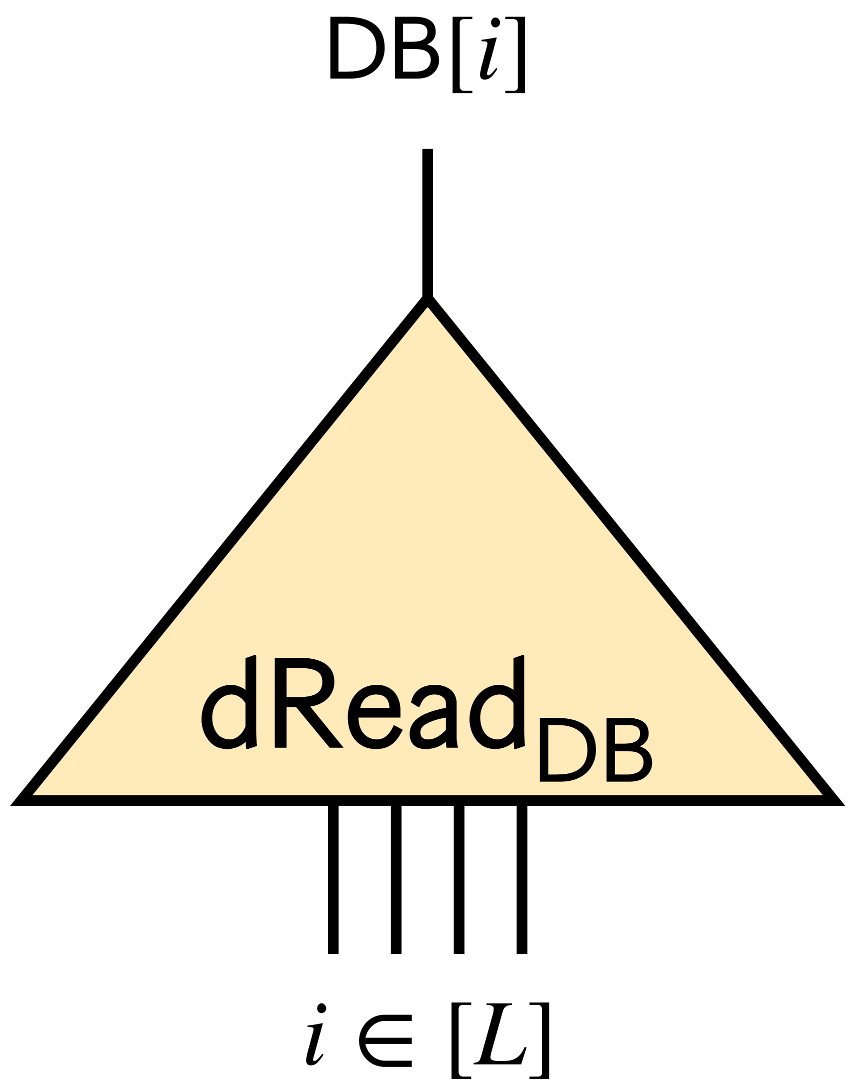
Efficient access to input/working memory

RAM Circuits

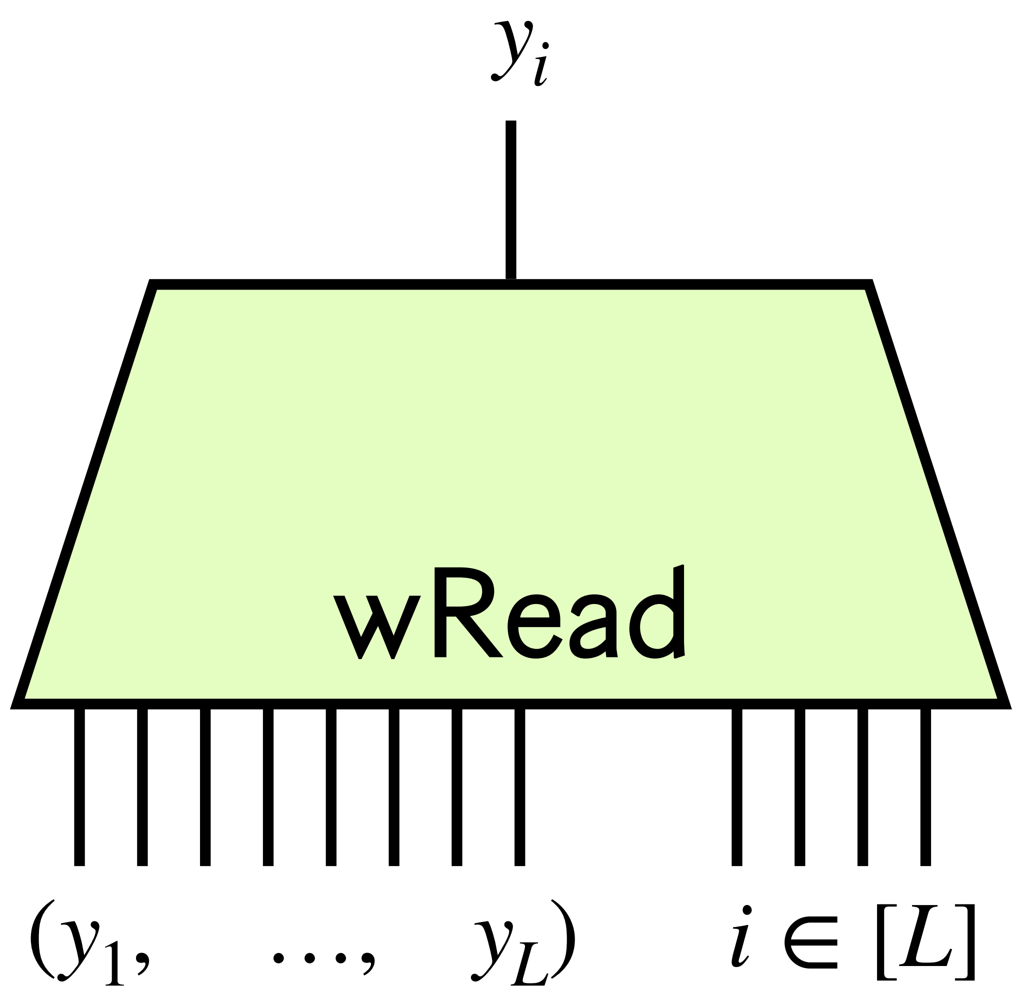
Boolean circuits + two new gates:

Data-Read Gates: For any DB $\in \{0,1\}^L$

Efficient access to a hardcoded database



Wire-Read Gates:

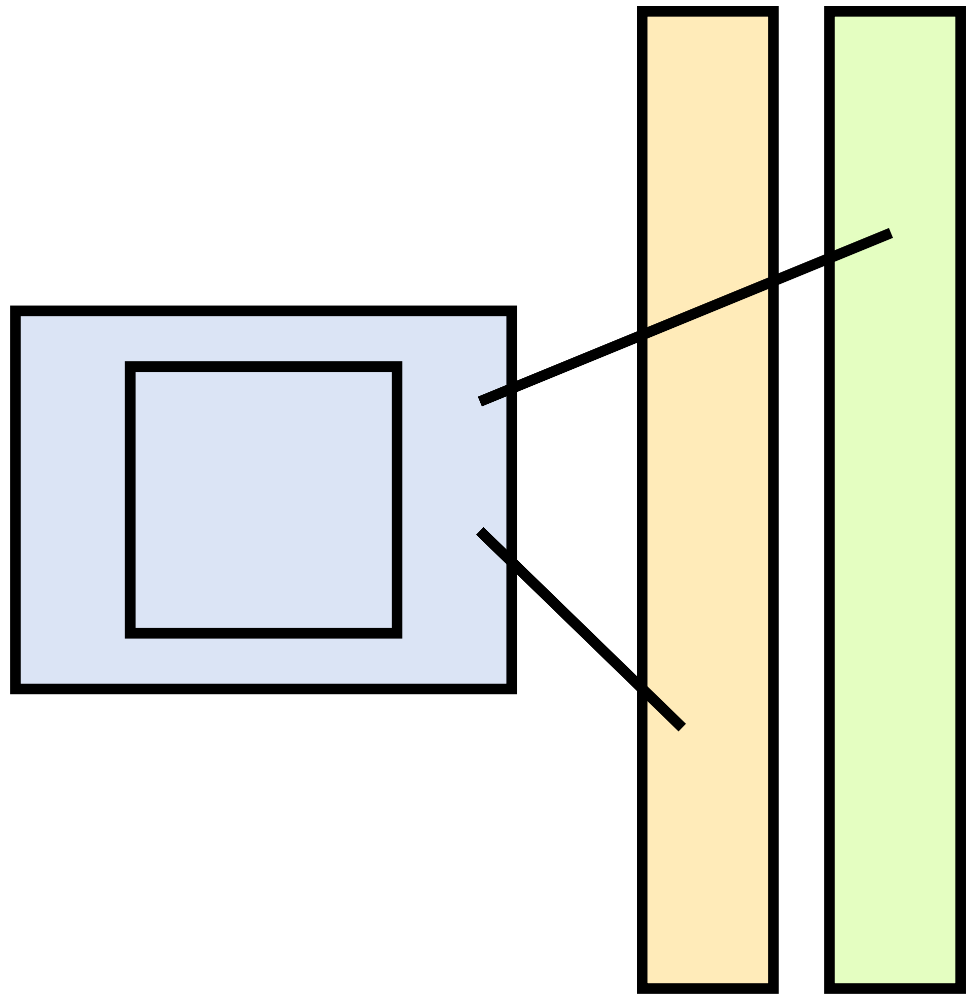


Efficient access to input/working memory

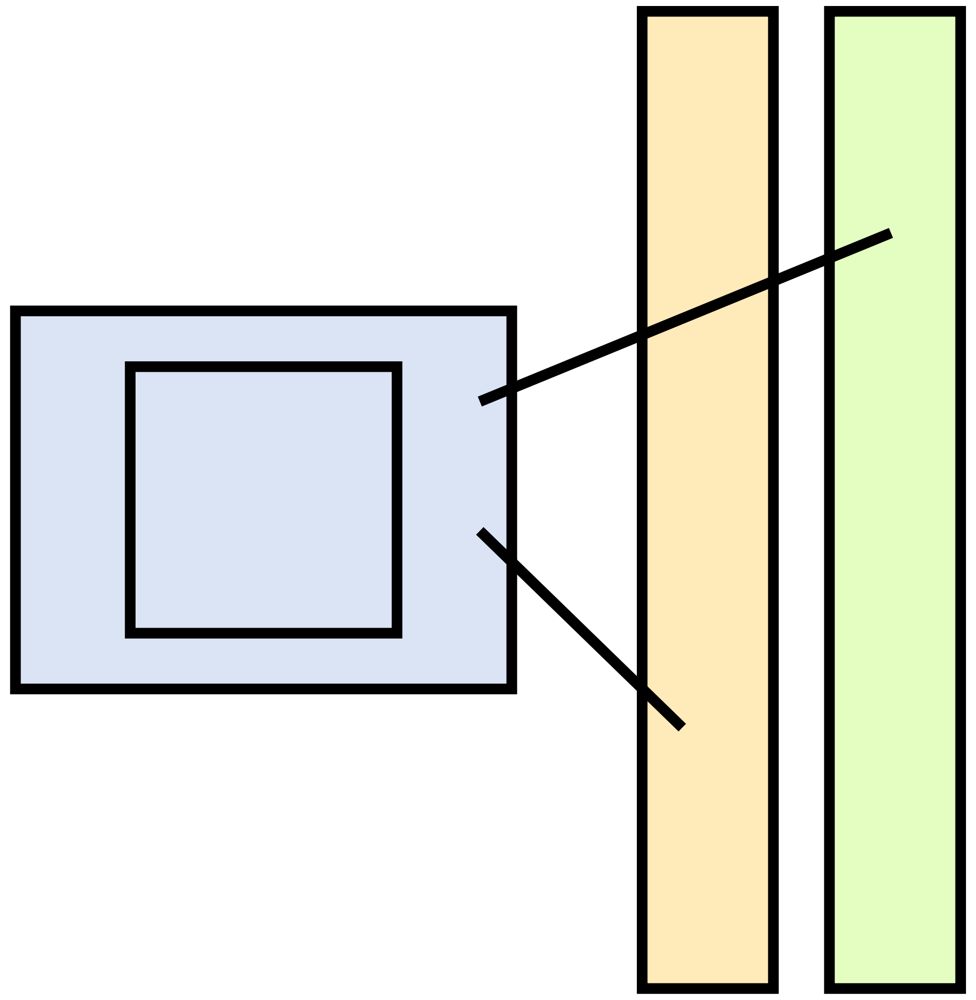
Fixed circuit topology $\Longrightarrow$ write locations must be fixed in advance

RAM Circuits

RAM Circuits

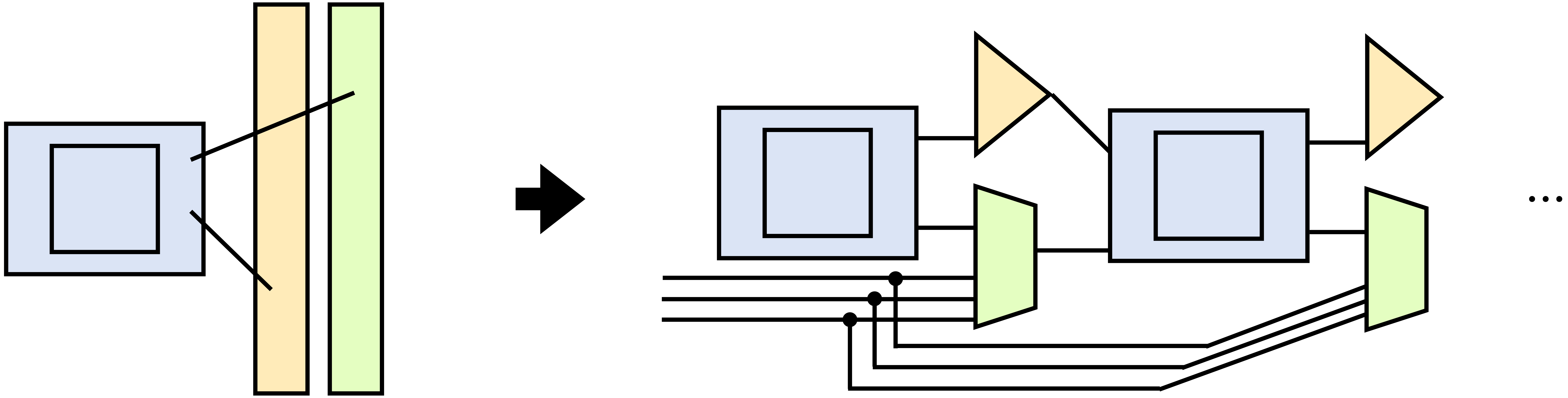


RAM Circuits



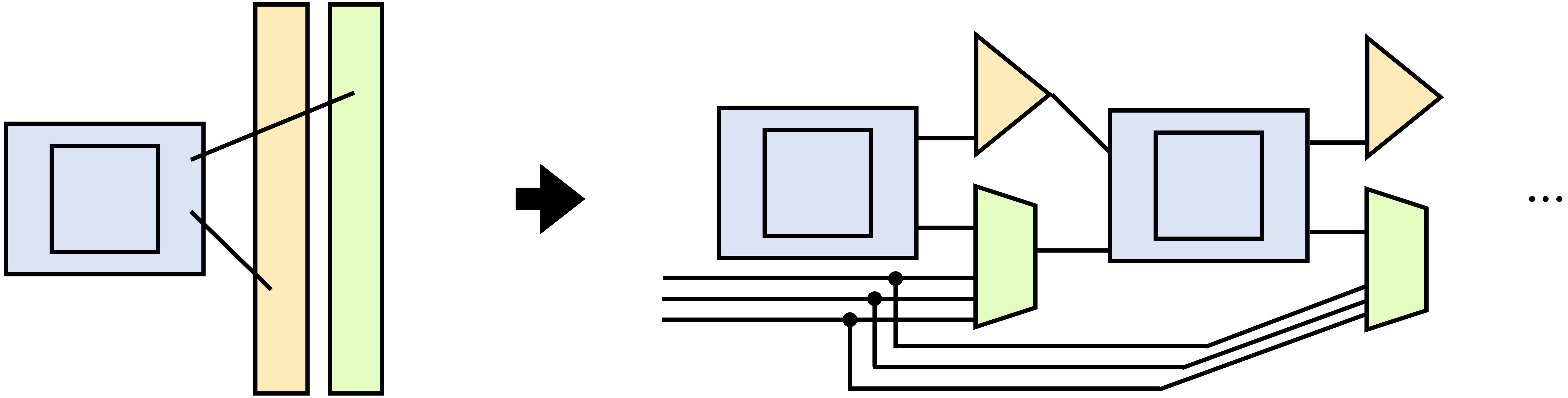
Lemma: A RAM program f_{DB} of runtime T can be represented by a RAM circuit of size $\tilde{O}(\max(T, N))$ for inputs of size N

RAM Circuits



Lemma: A RAM program f_{DB} of runtime T can be represented by a RAM circuit of size $\tilde{O}(\max(T, N))$ for inputs of size N

RAM Circuits



Lemma: A RAM program f_{DB} of runtime T can be represented by a RAM circuit of size $\tilde{O}(\max(T, N))$ for inputs of size N

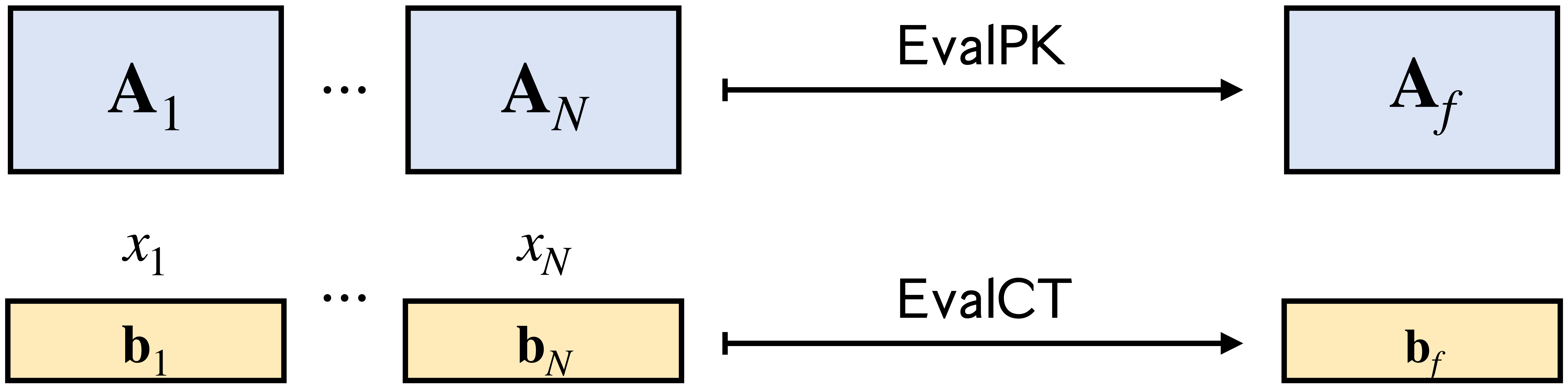
Note: The transformation yields RAM circuit of depth $\tilde{O}(T)$, but could do better if f_{DB} is parallelizable

Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \rightarrow \{0,1\}$

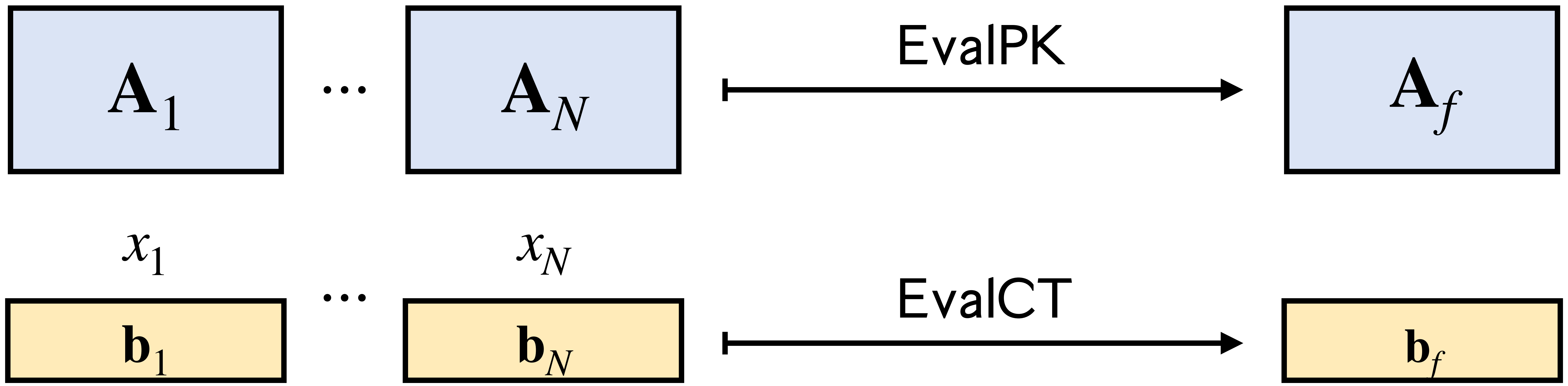


Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$



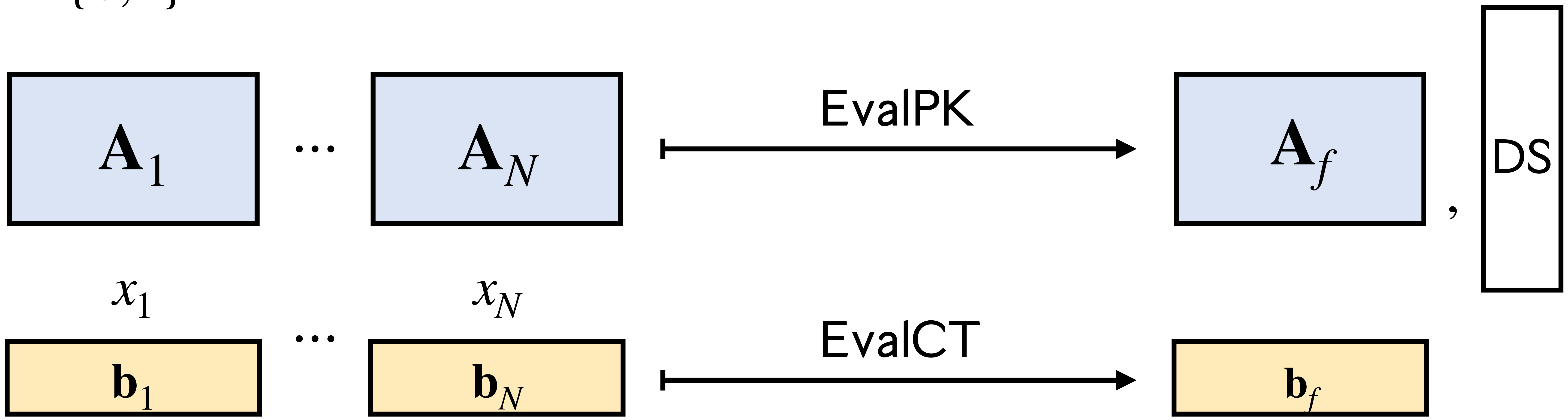
Result: We build (preprocessing) system of homomorphic operations s.t.:

Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$



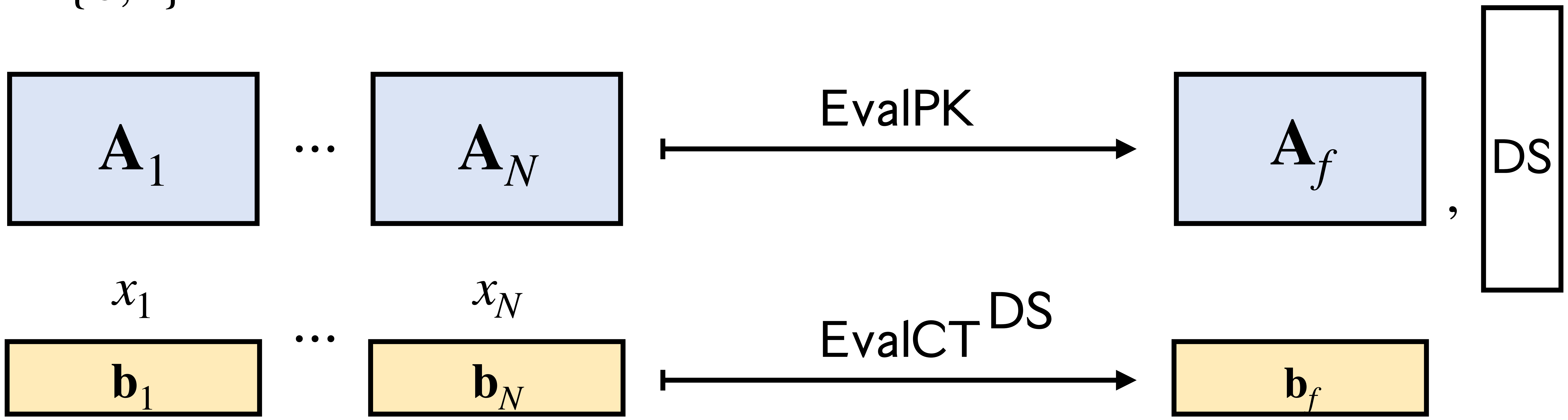
Result: We build (preprocessing) system of homomorphic operations s.t.:

Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f: \{0,1\}^N \to \{0,1\}$



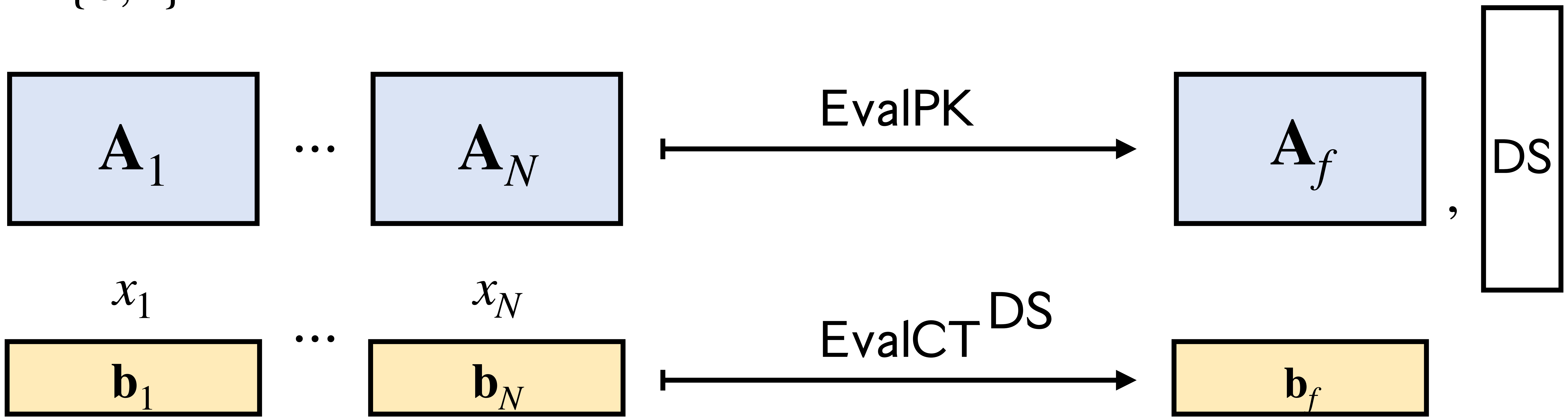
Result: We build (preprocessing) system of homomorphic operations s.t.:

Homomorphic Operations

[BGG+'14]

Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \rightarrow \{0,1\}$



Result: We build (preprocessing) system of homomorphic operations s.t.:

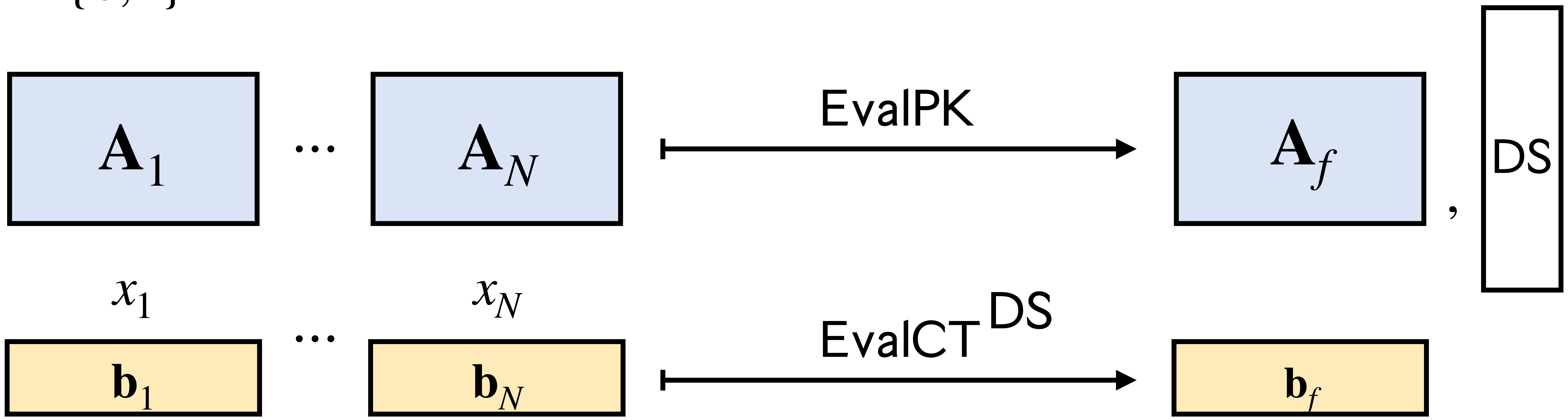
- EvalCT runs in time proportional to RAM circuit size of f

Homomorphic Operations

[BGG+'14]

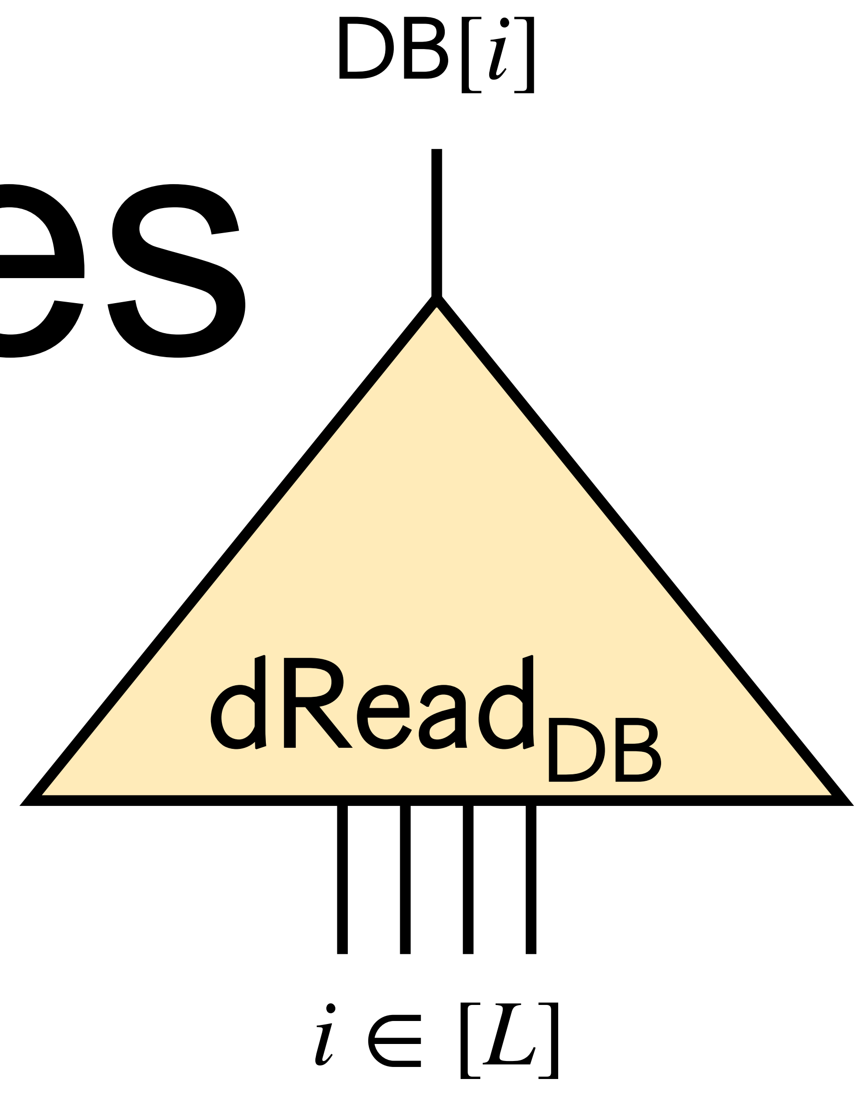
Goal: Given encodings of an input x want to get an encoding of the output $f(x)$

$f : \{0,1\}^N \to \{0,1\}$



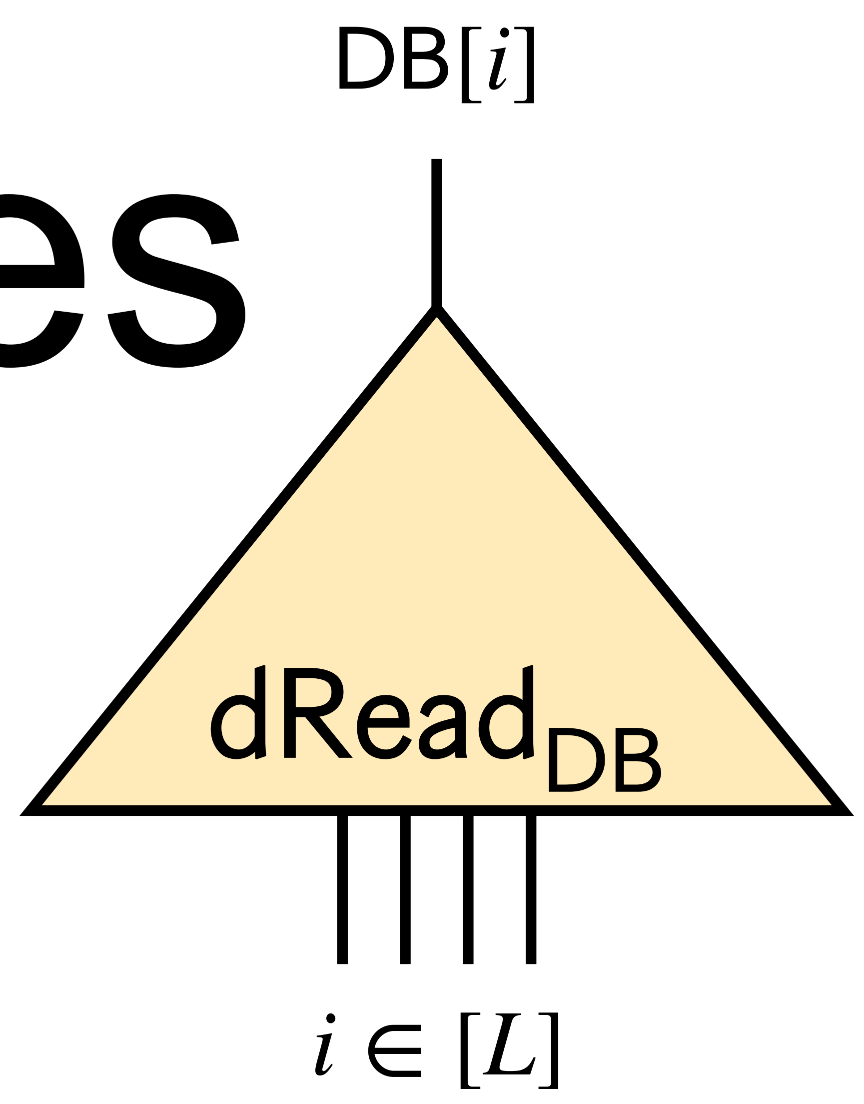
Result: We build (preprocessing) system of homomorphic operations s.t.:
- EvalCT runs in time proportional to RAM circuit size of f
- EvalPK runs in time proportional to boolean circuit size of f

Handling Data-Read Gates



Handling Data-Read Gates

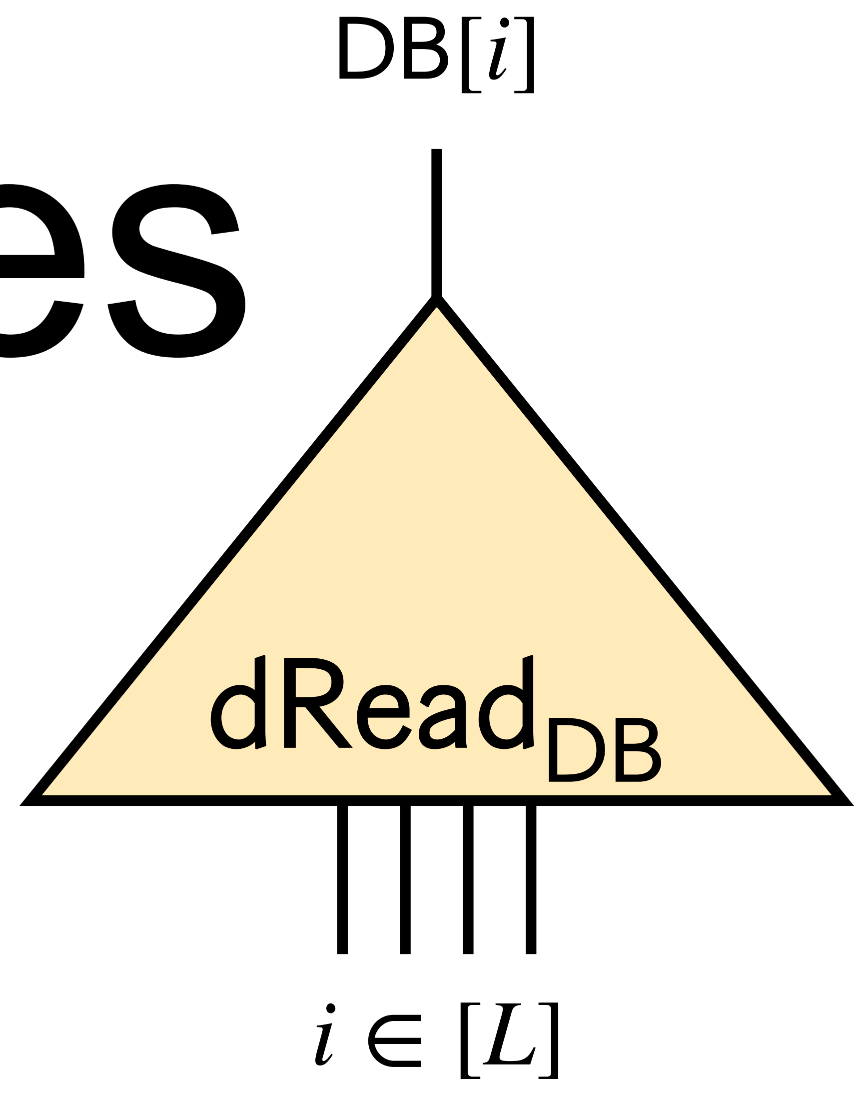
Goal: Apply [BGG+'14] operations to the function $\mathsf{dRead}_{\mathsf{DB}}$



Handling Data-Read Gates

Goal: Apply [BGG+'14] operations to the function $\mathsf{dRead}_{\mathsf{DB}}$

- But somehow speed up EvalCT

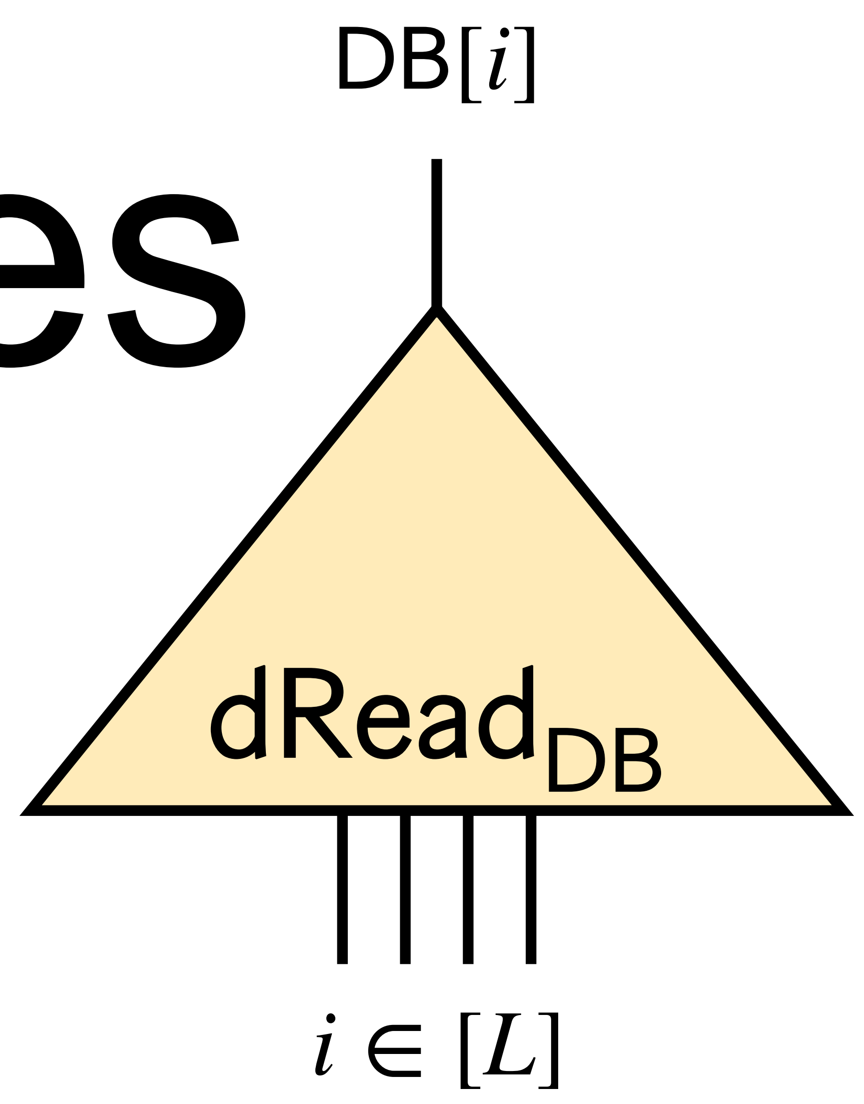


Handling Data-Read Gates

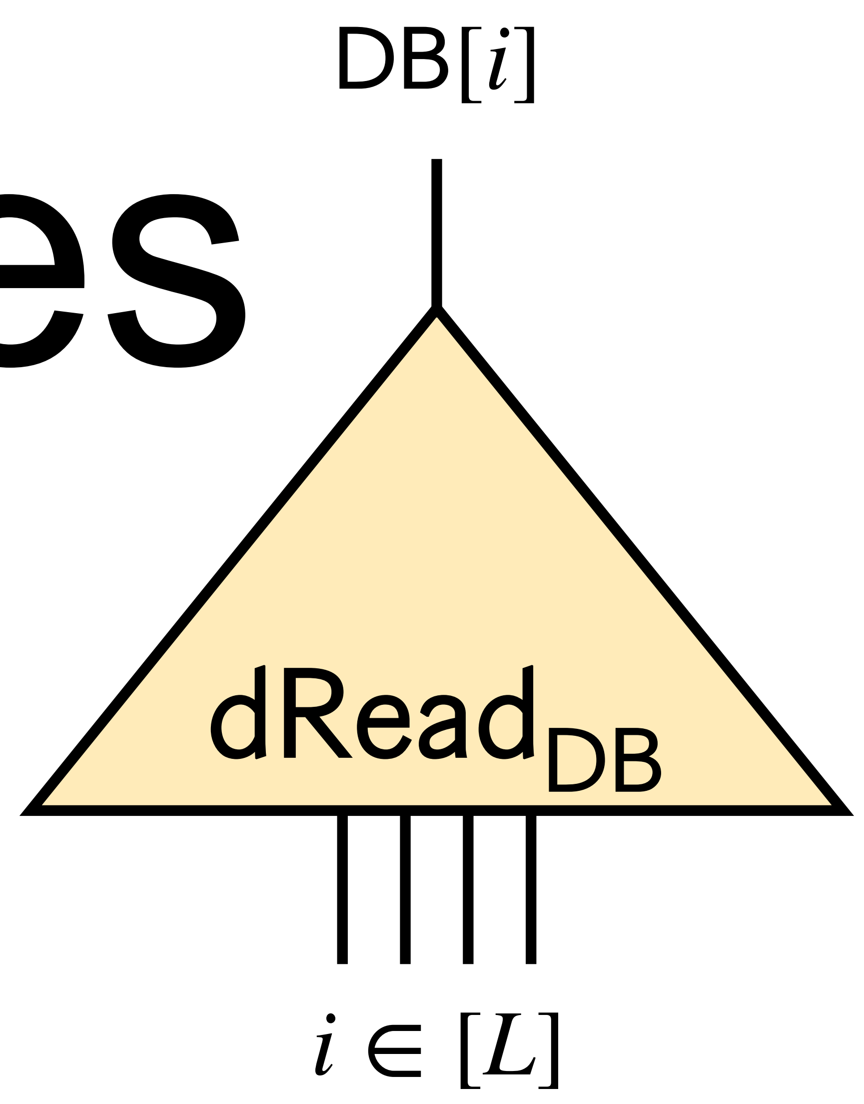
Goal: Apply [BGG+'14] operations to the function $\mathsf{dRead}_{\mathsf{DB}}$

- But somehow speed up EvalCT

Crucial Fact: [BGG+'14] operations are **linear**



Handling Data-Read Gates



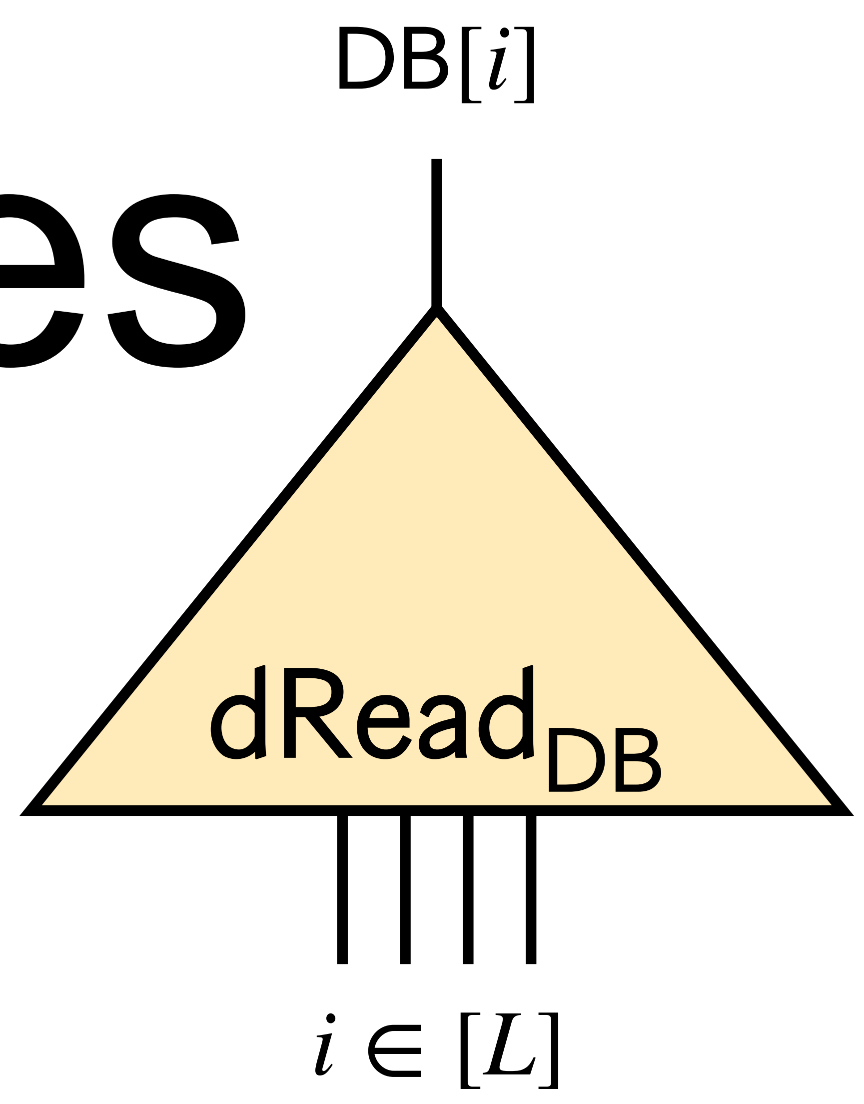
Goal: Apply [BGG+'14] operations to the function $\mathsf{dRead}_{\mathsf{DB}}$

- But somehow speed up EvalCT

Crucial Fact: [BGG+'14] operations are **linear**

There is a matrix $\mathbf{H}_{\mathsf{dRead}_{\mathsf{DB}},\mathbf{A},i}$ s.t.

Handling Data-Read Gates



Goal: Apply [BGG+'14] operations to the function $\mathsf{dRead}_{\mathsf{DB}}$

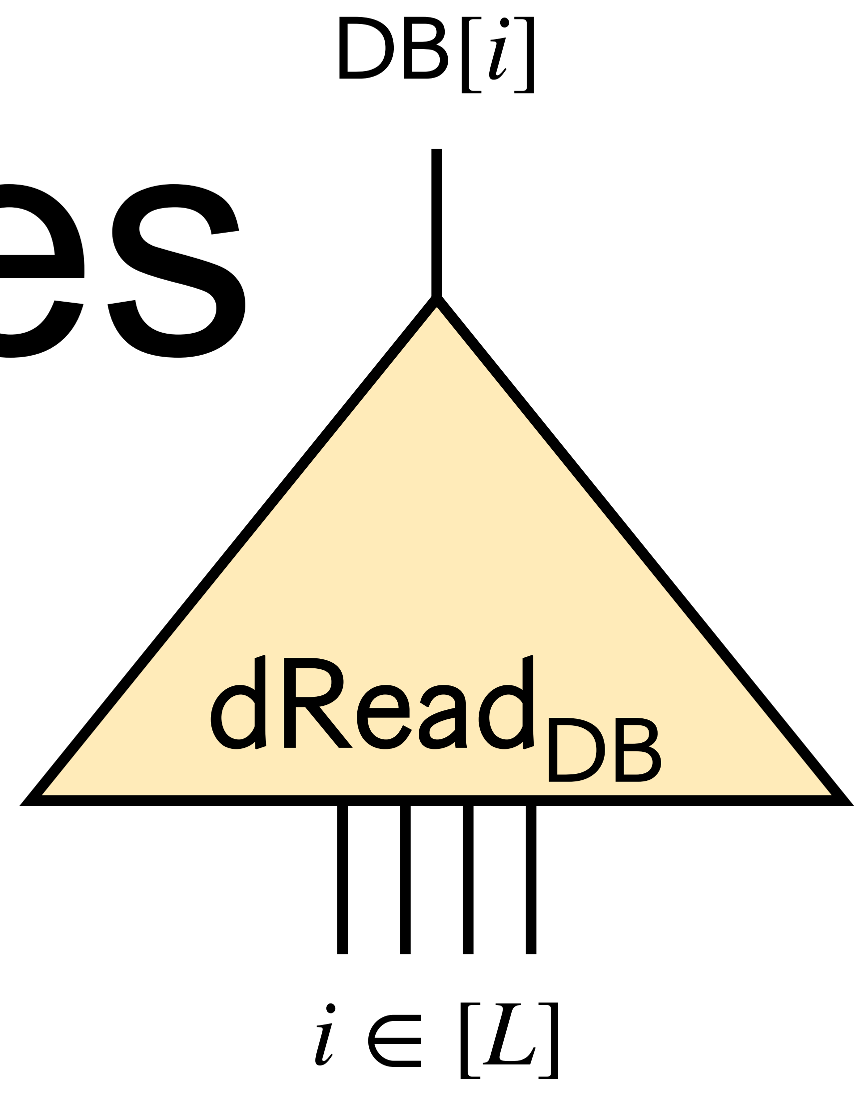
- But somehow speed up EvalCT

Crucial Fact: [BGG+'14] operations are **linear**

There is a matrix $\mathbf{H}_{\mathsf{dRead}_{\mathsf{DB}},\mathbf{A},i}$ s.t.

$$(\mathbf{b}_1, \ldots, \mathbf{b}_{\log L}) \cdot \mathbf{H}_{\mathsf{dRead}_{\mathsf{DB}},\mathbf{A},i} = \mathbf{b}_{\mathsf{dRead}_{\mathsf{DB}}}$$

Handling Data-Read Gates



Goal: Apply [BGG+'14] operations to the function dRead$_{\text{DB}}$

- But somehow speed up EvalCT

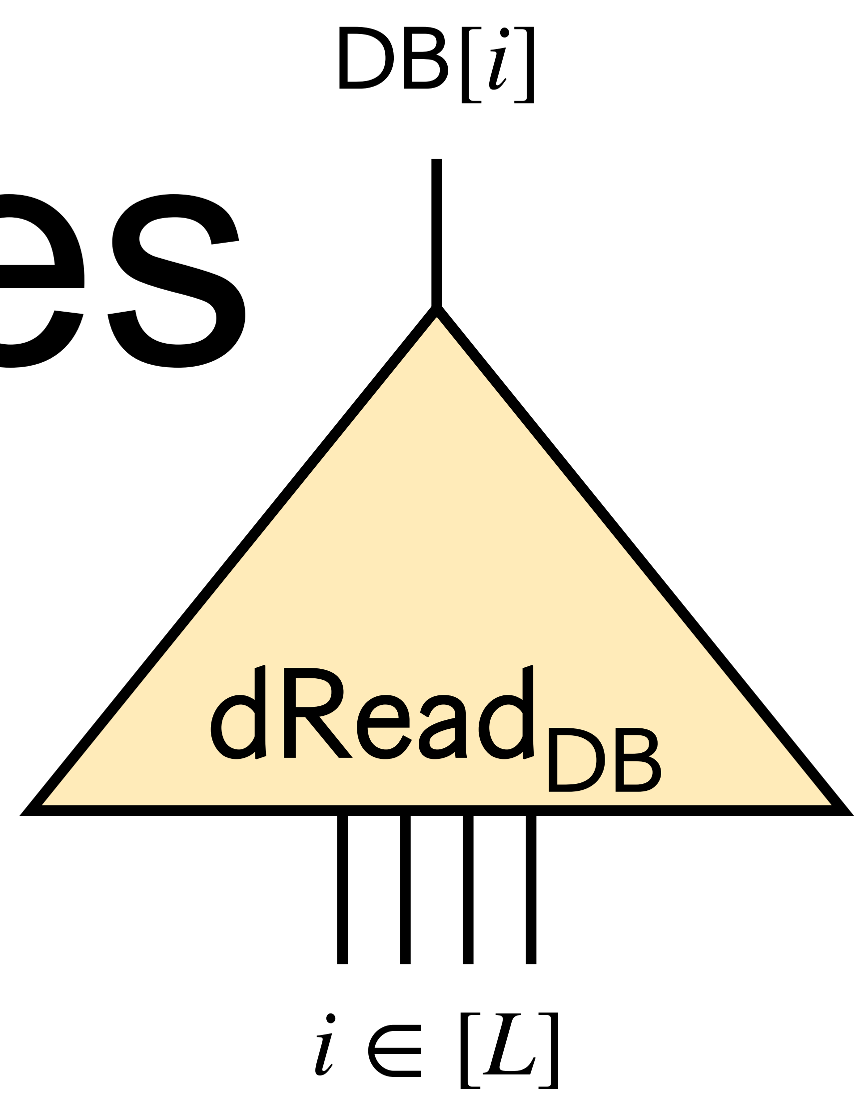
Crucial Fact: [BGG+'14] operations are **linear**

There is a matrix $\mathbf{H}_{\text{dRead}_{\text{DB}},\mathbf{A},i}$ s.t.

$$(\mathbf{b}_1, \ldots, \mathbf{b}_{\log L}) \cdot \mathbf{H}_{\text{dRead}_{\text{DB}},\mathbf{A},i} = \mathbf{b}_{\text{dRead}_{\text{DB}}}$$

Depends on $\mathbf{A}$ and i

Handling Data-Read Gates



Goal: Apply [BGG+'14] operations to the function dRead_{DB}

- But somehow speed up EvalCT

Crucial Fact: [BGG+'14] operations are **linear**

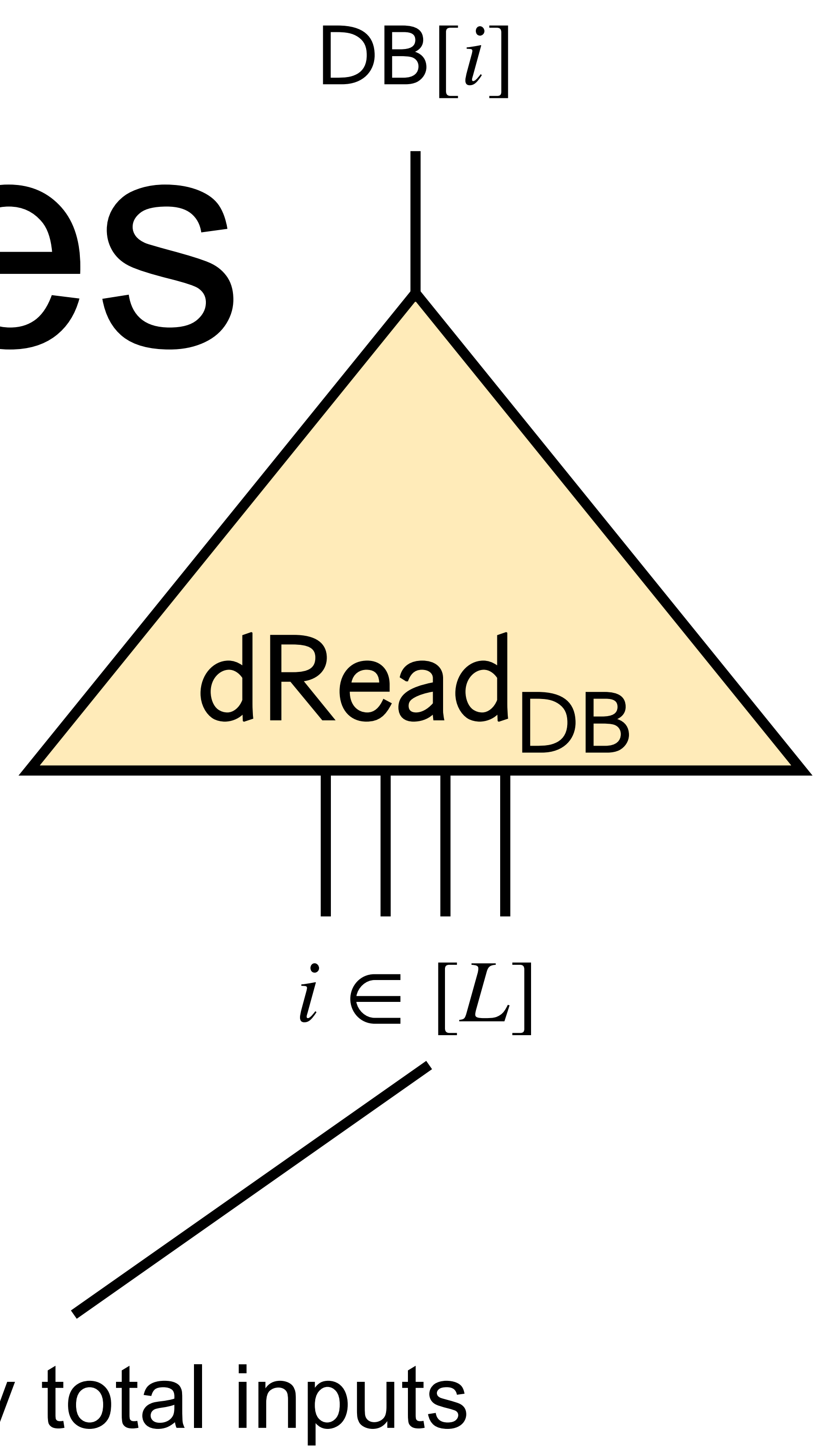
There is a matrix $\mathbf{H}_{\text{dRead}_{\text{DB}},\mathbf{A},i}$ s.t.

$$(\mathbf{b}_1, \ldots, \mathbf{b}_{\log L}) \cdot \mathbf{H}_{\text{dRead}_{\text{DB}},\mathbf{A},i} = \mathbf{b}_{\text{dRead}_{\text{DB}}}$$

Depends on $\mathbf{A}$ and i

Idea: During preprocessing, just compute all $\mathbf{H}_{\text{dRead}_{\text{DB}},\mathbf{A},i}$ in advance for each $i \in [L]$

Handling Data-Read Gates



Goal: Apply [BGG+'14] operations to the function $\mathsf{dRead}_{\mathsf{DB}}$

- But somehow speed up EvalCT

Crucial Fact: [BGG+'14] operations are **linear**

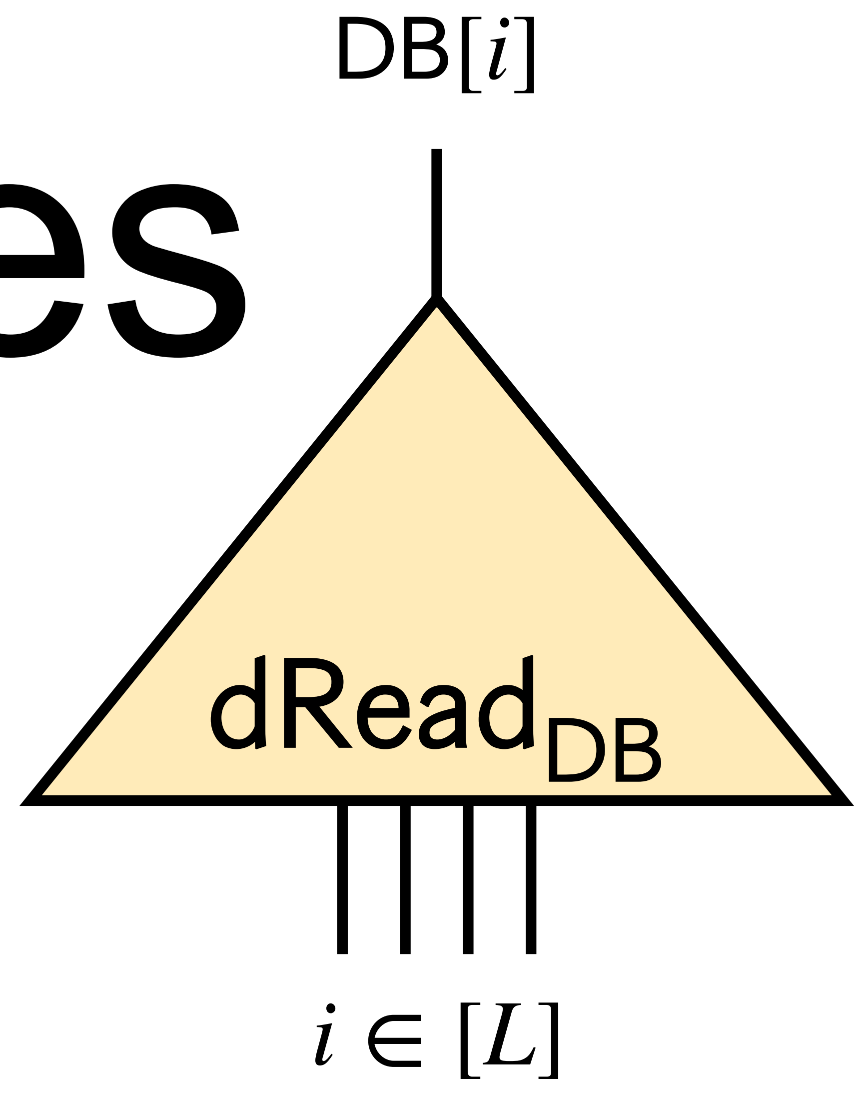
There is a matrix $\mathbf{H}_{\mathsf{dRead}_{\mathsf{DB}},\mathbf{A},i}$ s.t.

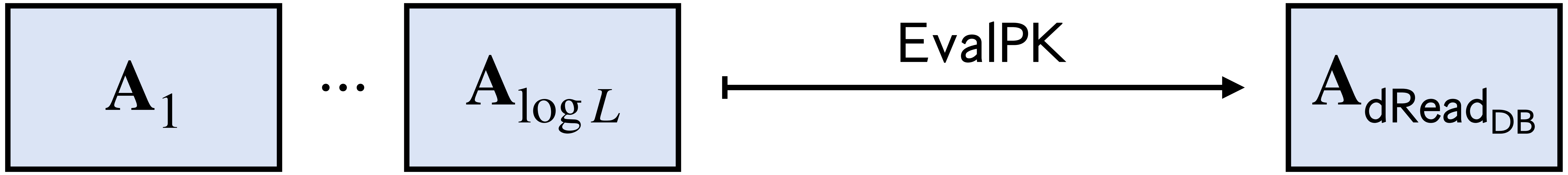
$$(\mathbf{b}_1, \ldots, \mathbf{b}_{\log L}) \cdot \mathbf{H}_{\mathsf{dRead}_{\mathsf{DB}},\mathbf{A},i} = \mathbf{b}_{\mathsf{dRead}_{\mathsf{DB}}}$$

Depends on $\mathbf{A}$ and i

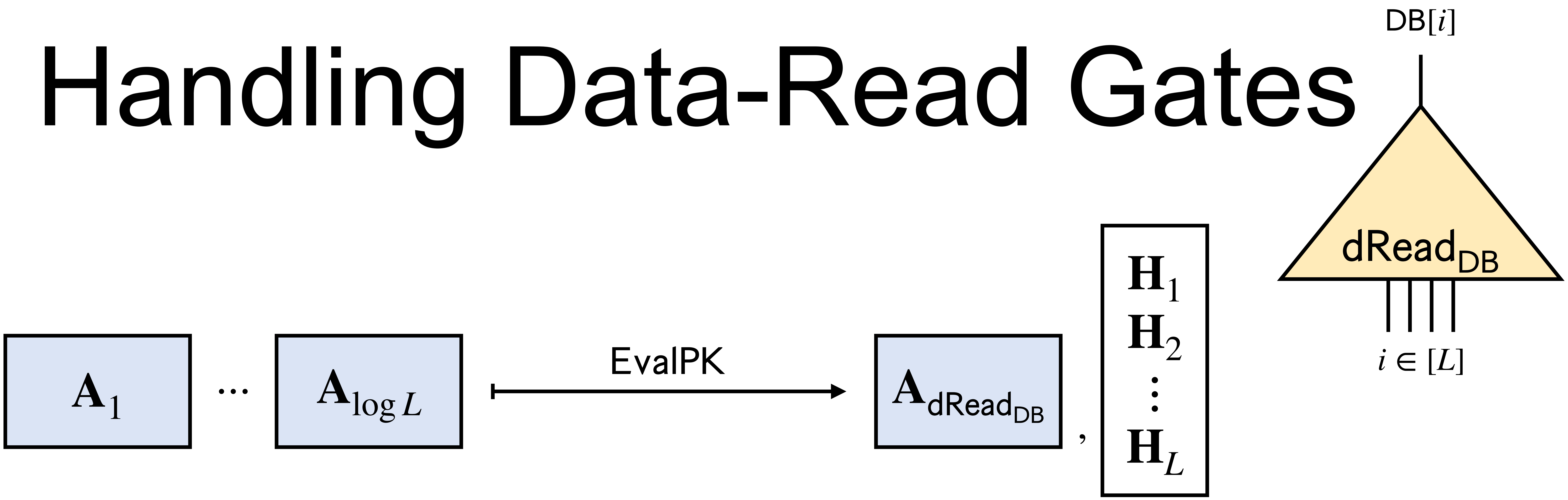
Idea: During preprocessing, just compute all $\mathbf{H}_{\mathsf{dRead}_{\mathsf{DB}},\mathbf{A},i}$ in advance for each $i \in [L]$

Handling Data-Read Gates

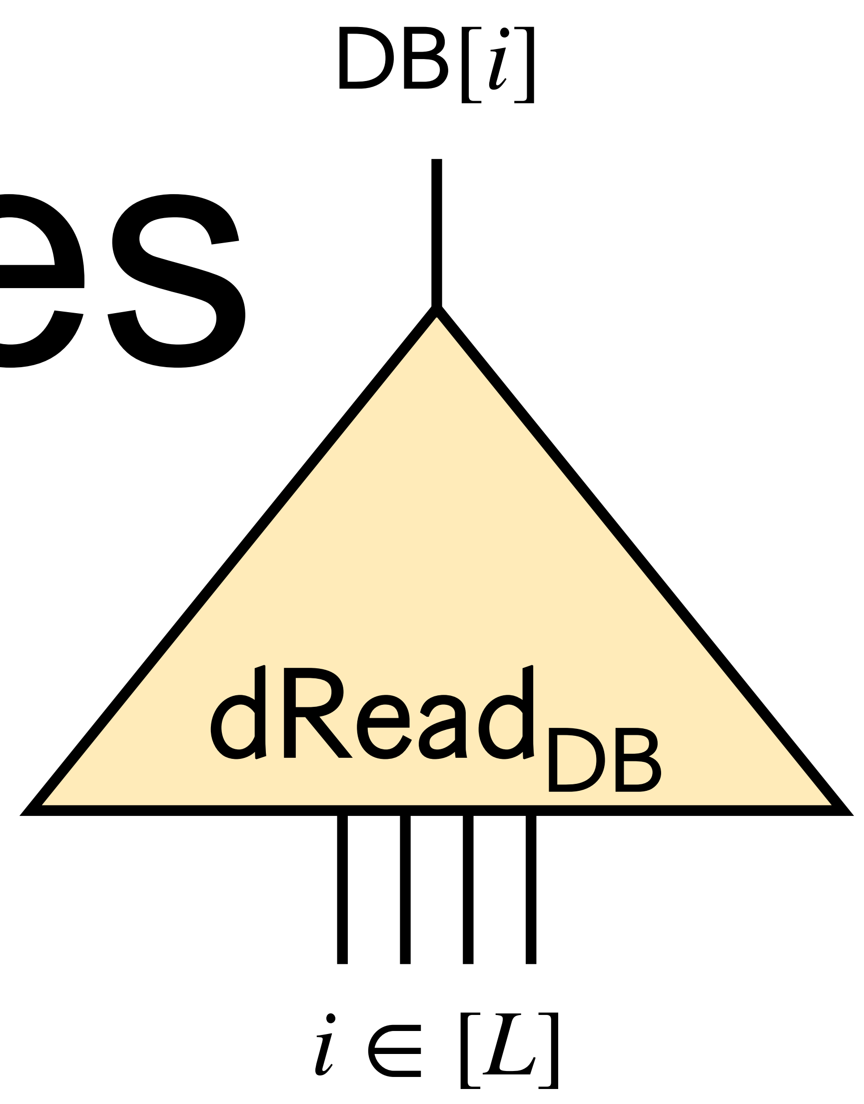


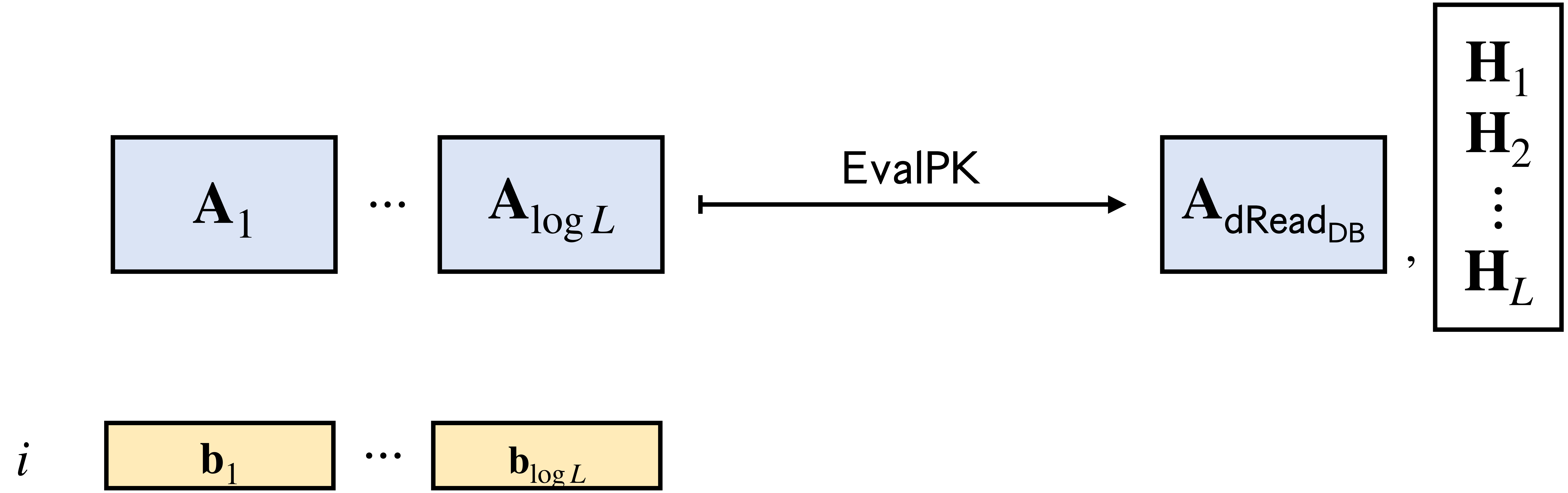


Handling Data-Read Gates

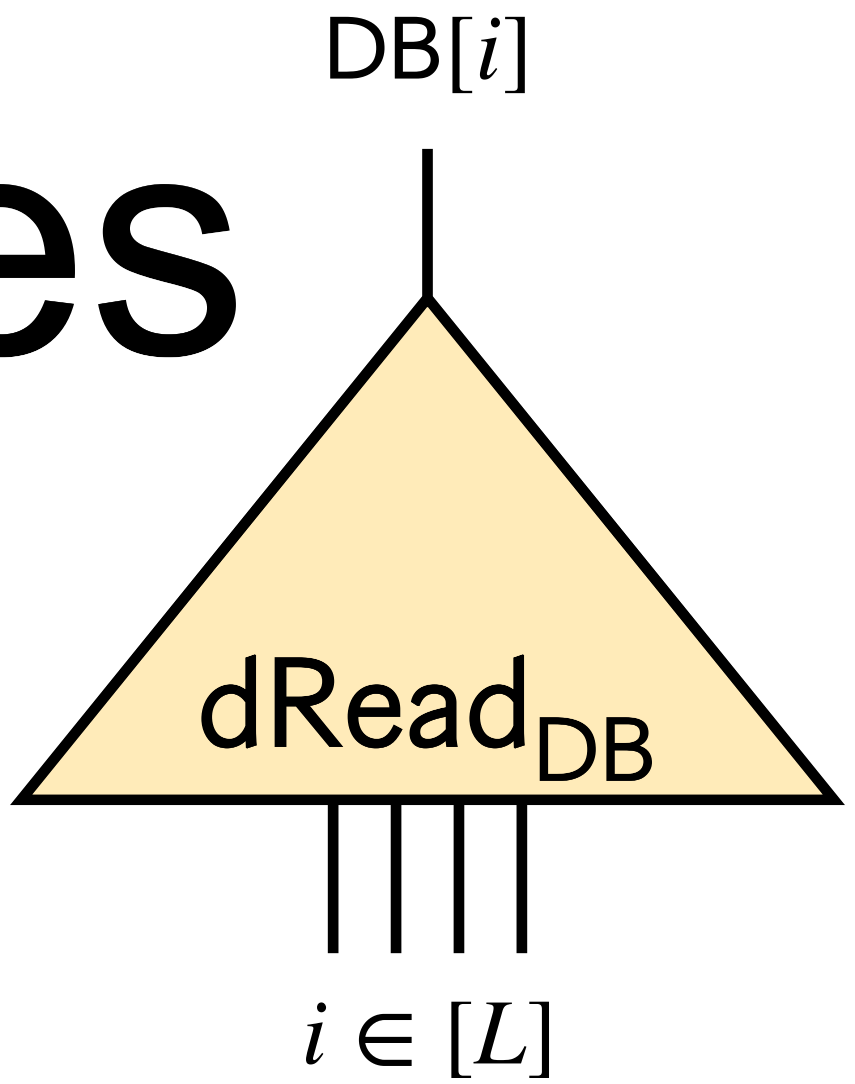


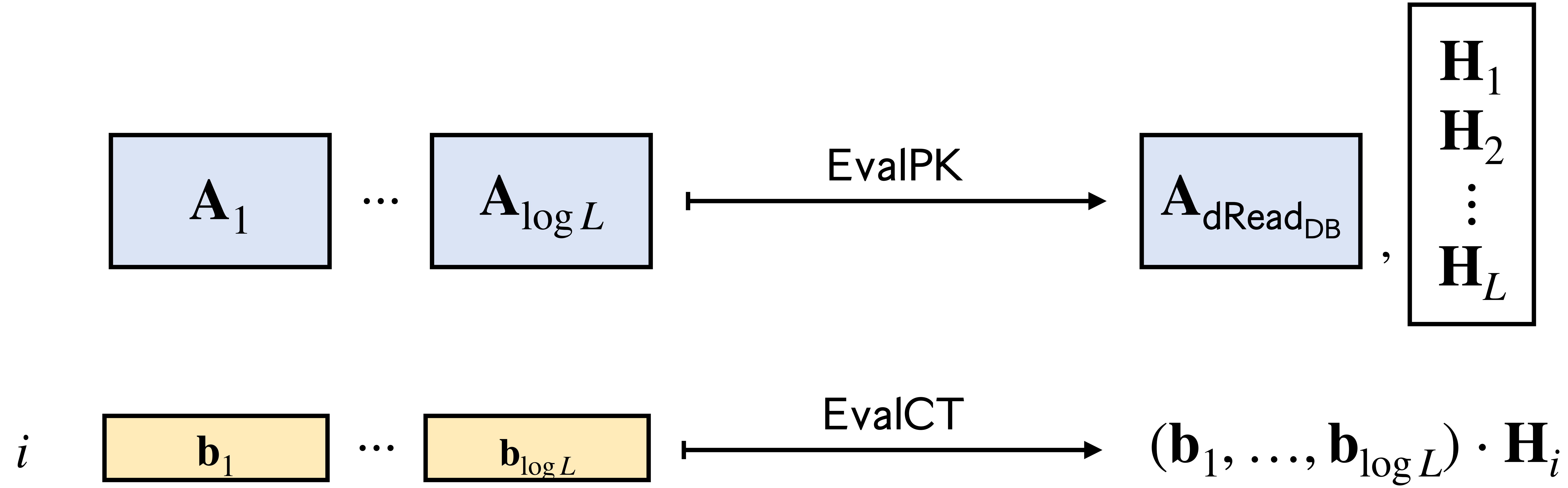
Handling Data-Read Gates



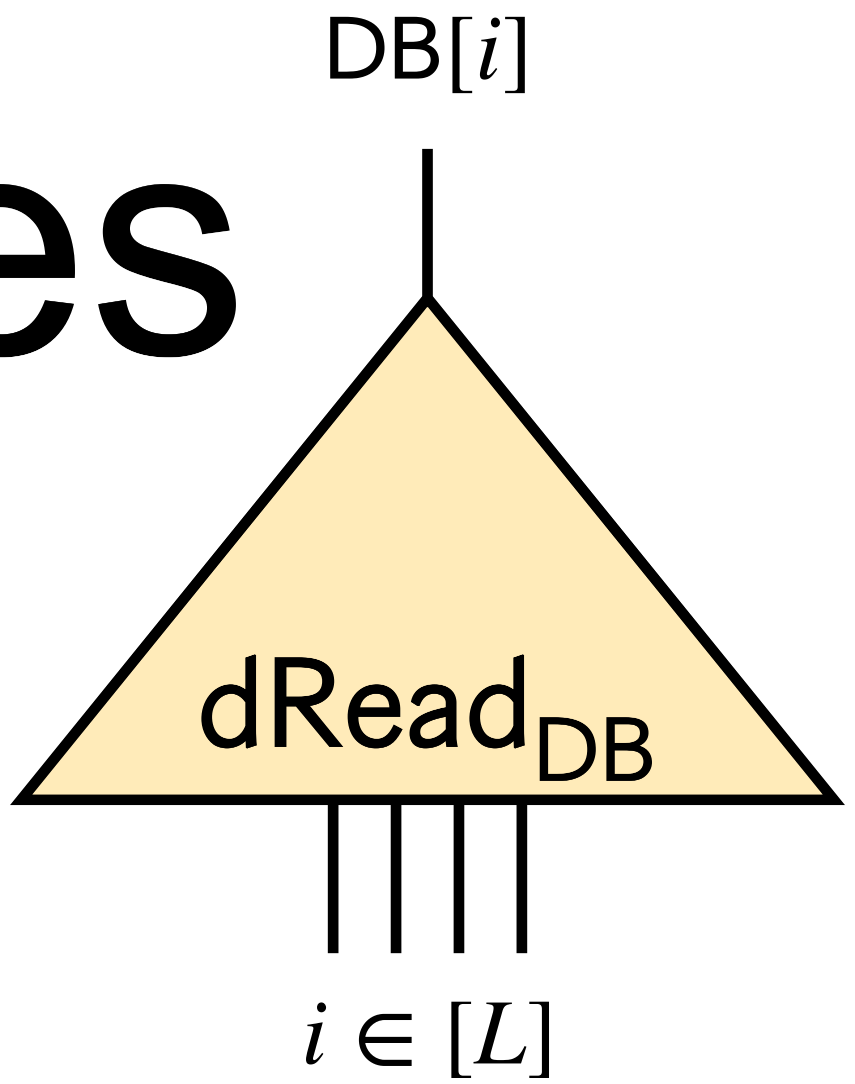


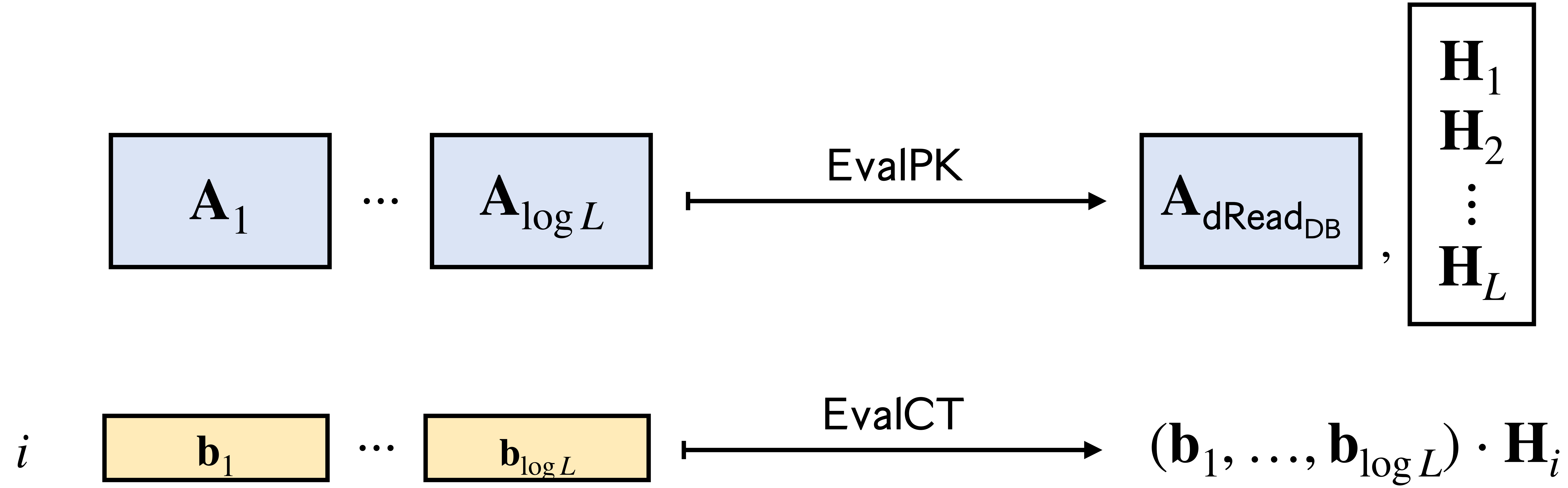
Handling Data-Read Gates





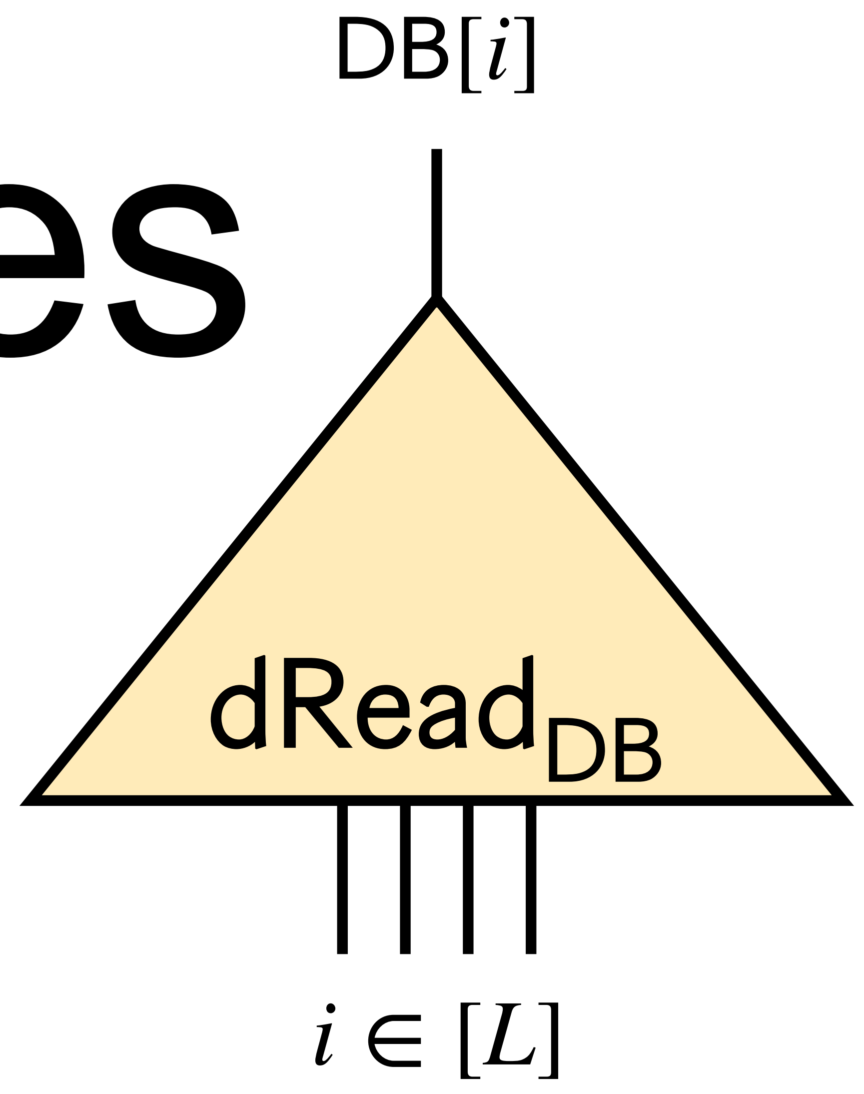
Handling Data-Read Gates

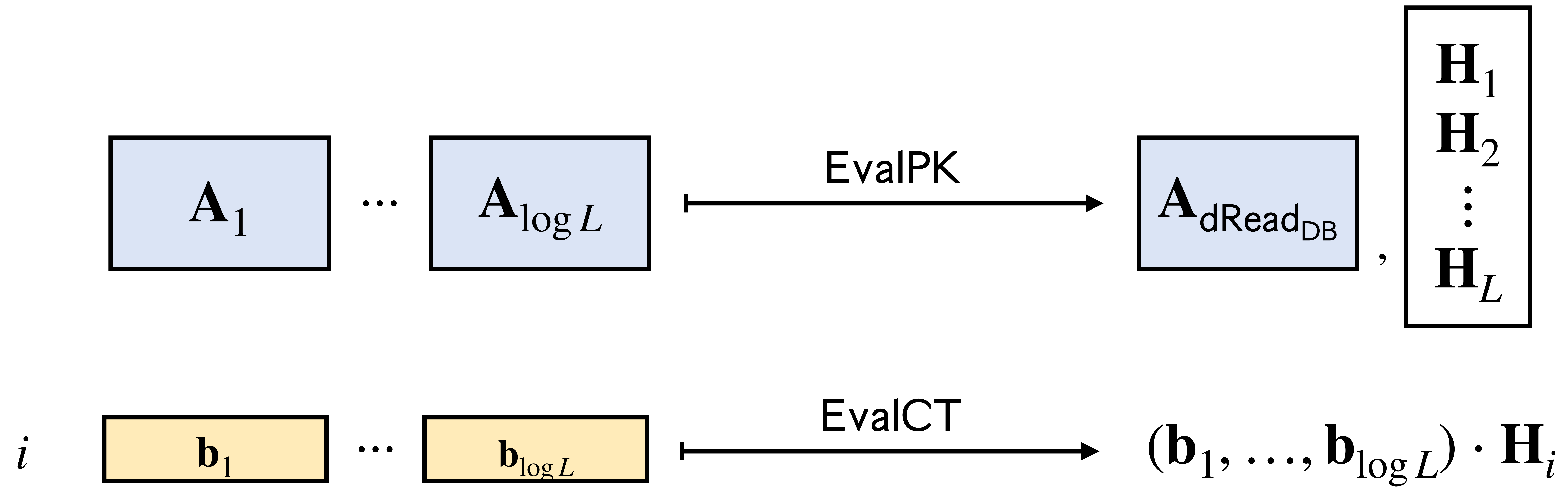




There are L many $\mathbf{H}$ matrices and each takes $\tilde{O}(L)$ time to compute
$\Longrightarrow$ EvalPK runs in time $\tilde{O}(L^2)$

Handling Data-Read Gates

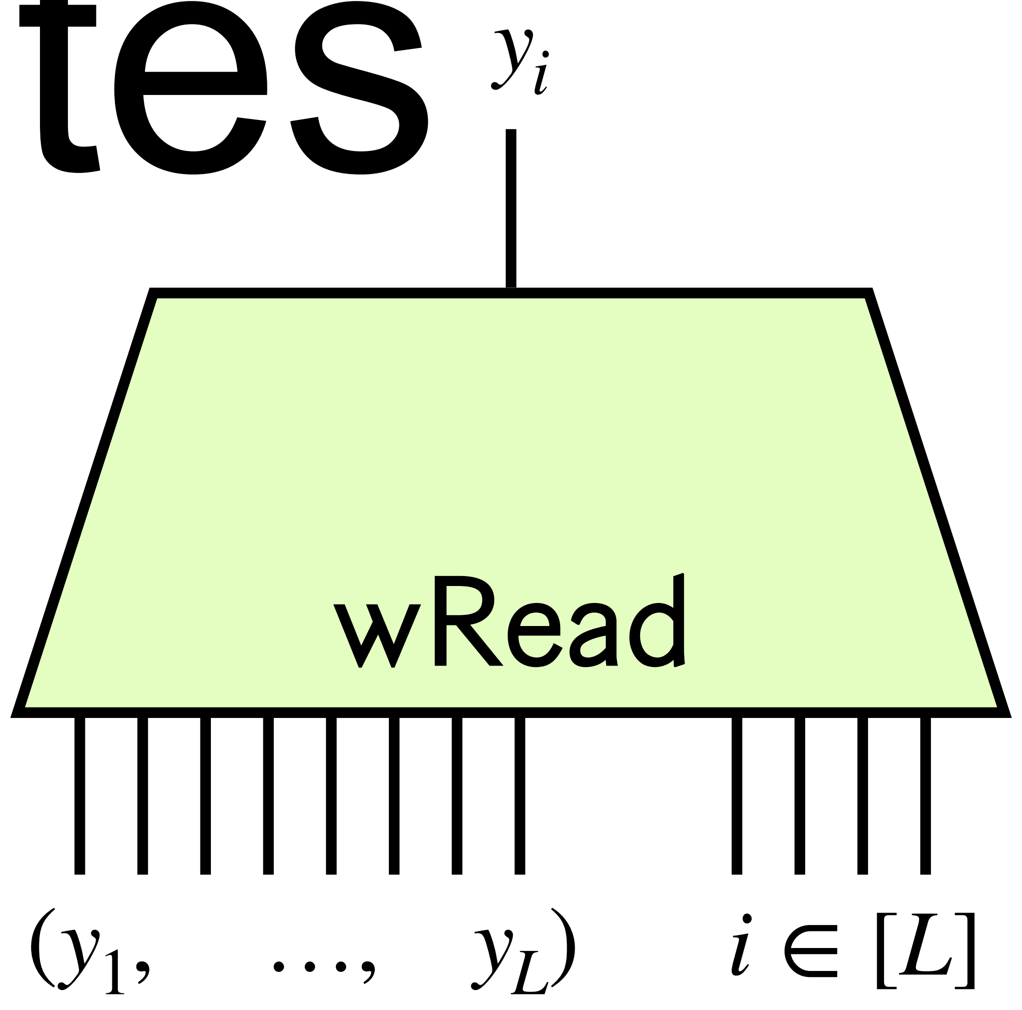




There are L many $\mathbf{H}$ matrices and each takes $\tilde{O}(L)$ time to compute
$\Longrightarrow$ EvalPK runs in time $\tilde{O}(L^2)$

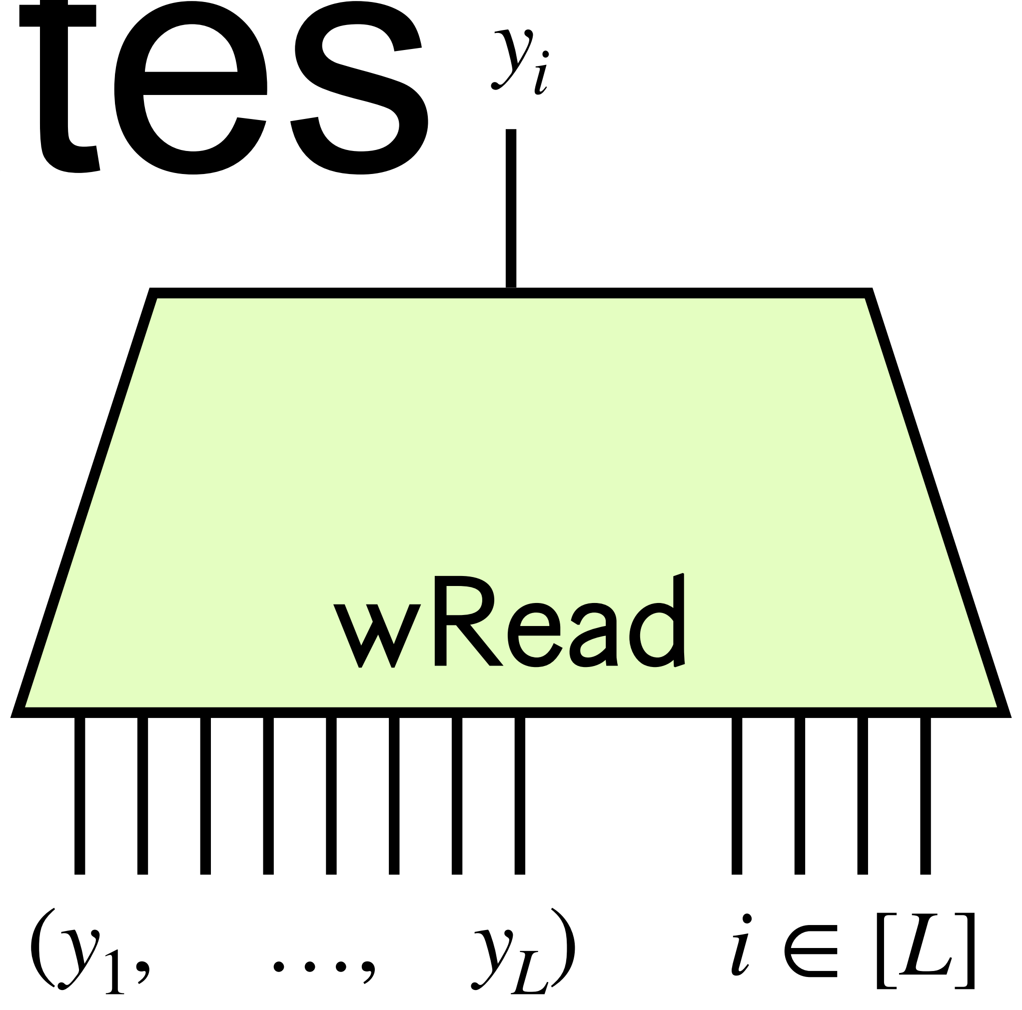
Can get just $\tilde{O}(L)$ using a recursive data structure that lets you compute $\mathbf{H}$ on the fly

Handling Wire-Read Gates

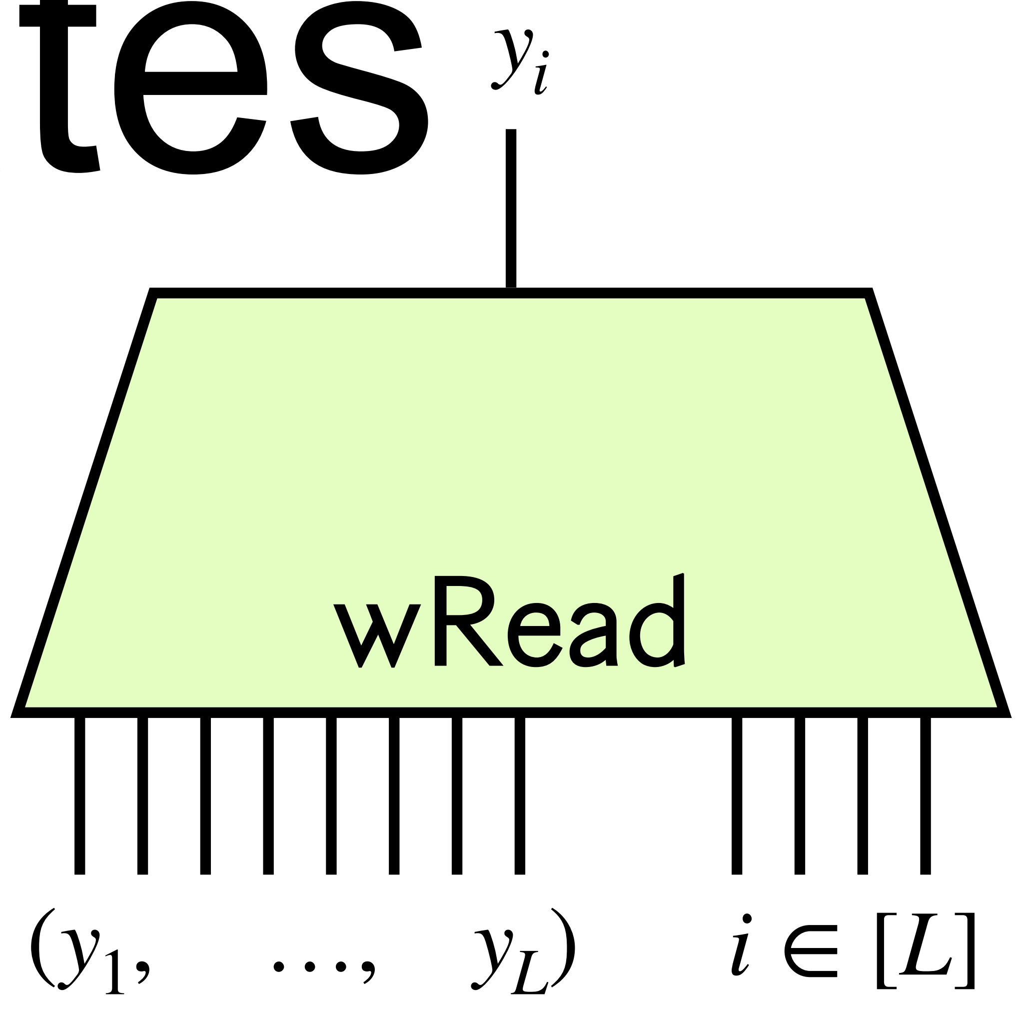


Handling Wire-Read Gates

Problem: wRead has exponentially many inputs!



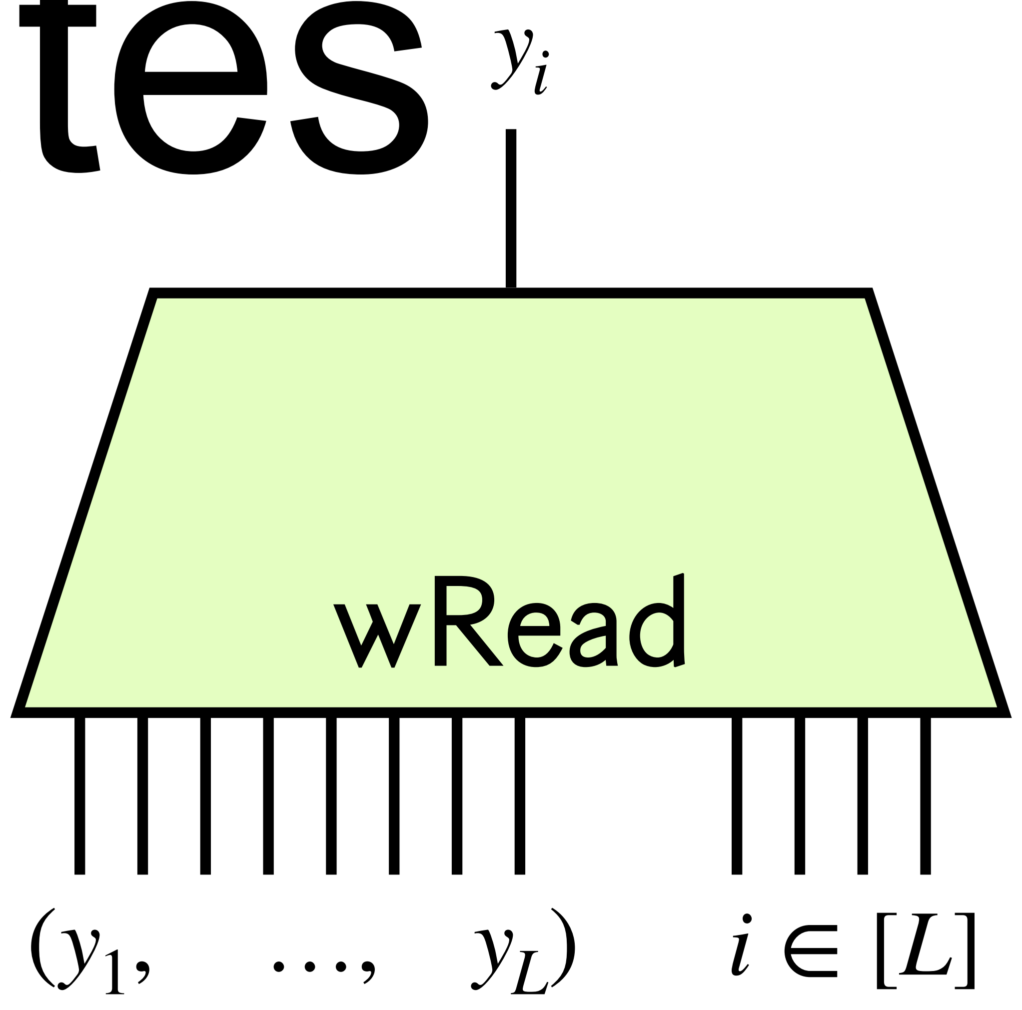
Handling Wire-Read Gates



Problem: wRead has exponentially many inputs!

$$\mathrm{wRead}(y, i) = \sum_{j=1}^{L} y_i \cdot \mathbb{I}(i = j)$$

Handling Wire-Read Gates

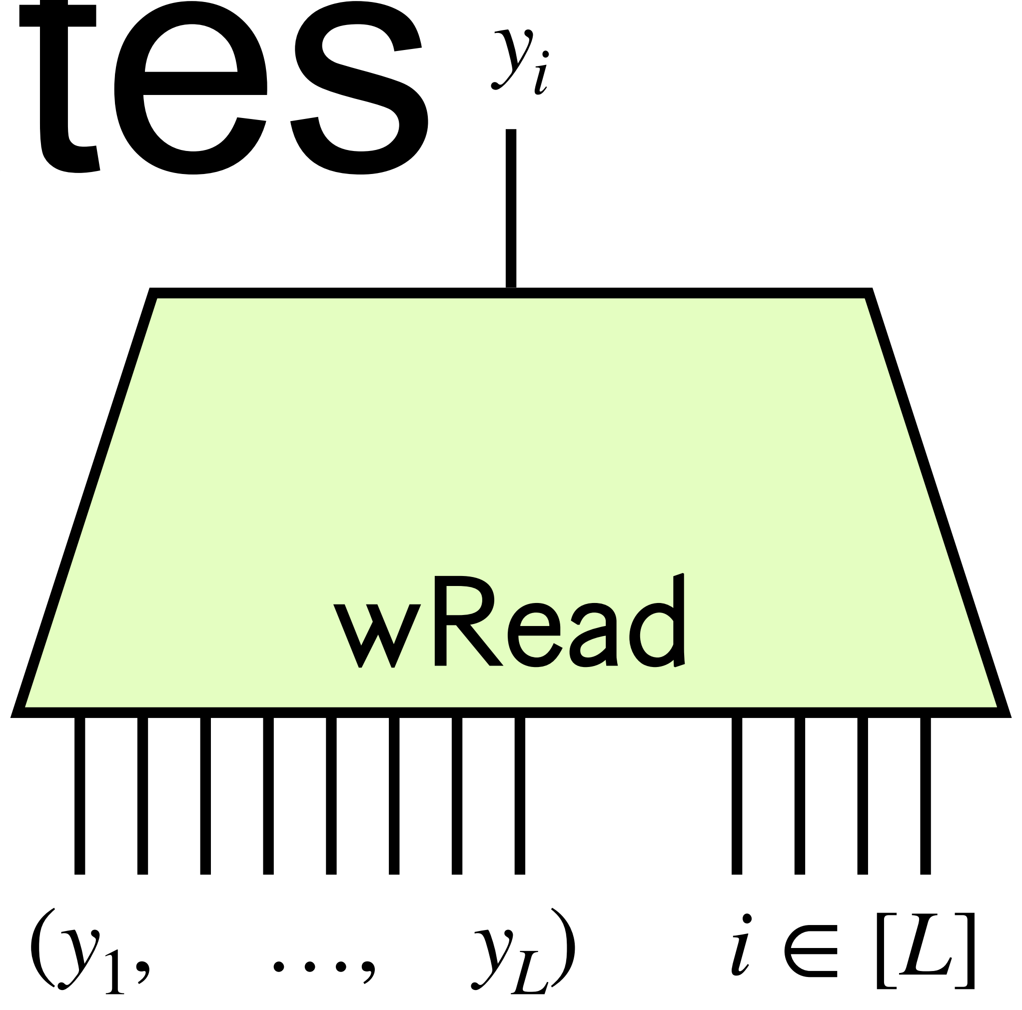


Problem: wRead has exponentially many inputs!

$$\text{wRead}(y, i) = \sum_{j=1}^{L} y_i \cdot \mathbb{I}(i = j)$$

Saving Grace: For a given i, $\mathbf{b}_{\text{wRead}}$ only depends on one location of y

Handling Wire-Read Gates



Problem: wRead has exponentially many inputs!

$$\text{wRead}(y, i) = \sum_{j=1}^{L} y_i \cdot \mathbb{1}(i = j)$$

Saving Grace: For a given i, $\mathbf{b}_{\text{wRead}}$ only depends on one location of y

Can precompute some data structure in time $\tilde{O}(L)$ over all i's that allows computing $\mathbf{H}_{y,i}$ on the fly

Removing depth dependence

Removing depth dependence

Prior work: [HLL'23] show how to bootstrap homomorphic operations to eliminate error growth assuming circular security

Removing depth dependence

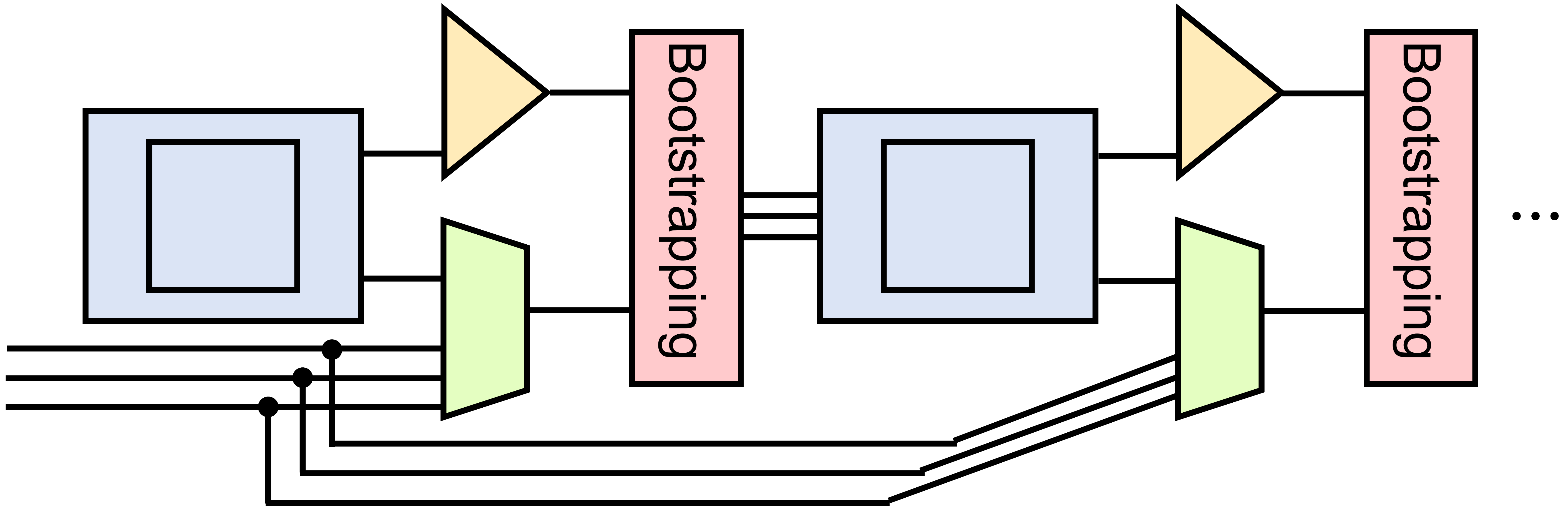
Prior work: [HLL'23] show how to bootstrap homomorphic operations to eliminate error growth assuming circular security

We can apply [HLL'23] techniques to our setting to get unbounded depth RAM-LFE

Removing depth dependence

Prior work: [HLL'23] show how to bootstrap homomorphic operations to eliminate error growth assuming circular security

We can apply [HLL'23] techniques to our setting to get unbounded depth RAM-LFE



Additional Result: ABE

Additional Result: ABE

Result: We build ABE for RAM circuits of bounded depth from LWE

Additional Result: ABE

Result: We build ABE for RAM circuits of bounded depth from LWE

- Our preprocessing preserves linearity and hence lattice trapdoors $\implies$ [BGG+'14] ABE construction goes through

Additional Result: ABE

Result: We build ABE for RAM circuits of bounded depth from LWE

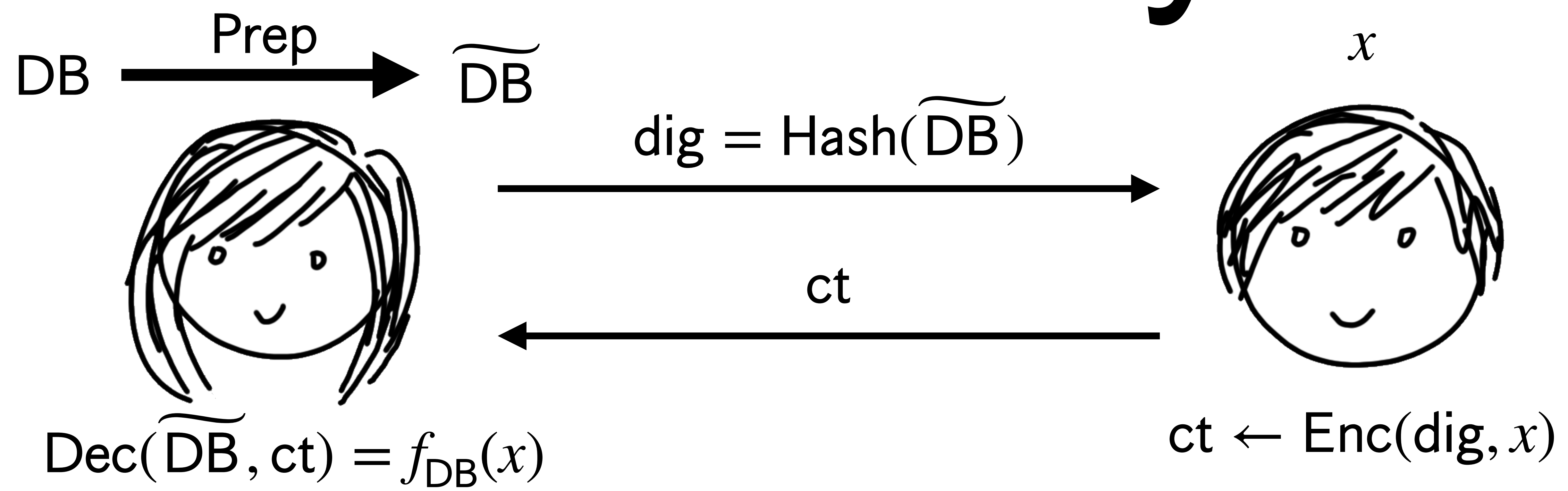
- Our preprocessing preserves linearity and hence lattice trapdoors $\Longrightarrow$ [BGG+'14] ABE construction goes through
- As in [HLL'23] cannot remove depth dependence without stronger assumptions

Additional Result: ABE

Result: We build ABE for RAM circuits of bounded depth from LWE

- Our preprocessing preserves linearity and hence lattice trapdoors $\Longrightarrow$ [BGG+'14] ABE construction goes through
- As in [HLL'23] cannot remove depth dependence without stronger assumptions
  - Still captures parallel RAM computation

Summary



Result: We build LFE for RAM programs from RingLWE + circular security

- Prep runtime scales with **circuit size** of RAM program
- Enc runtime slightly superlinear in input size
- Dec runtime slightly superlinear in RAM runtime

Result: We build ABE for RAM circuits of bounded depth from LWE