Efficient Laconic Cryptography from Learning With Errors

Nico Döttling[1], Dimitris Kolonelos[2,3], **Russell W. F. Lai**[4],
Chuanwei Lin[1], Giulio Malavolta[5], Ahmadreza Rahimi[5]

[1]CISPA Helmholtz Center for Information Security, Germany
[2]IMDEA Software Institute, Spain
[3]Universidad Politecnica de Madrid, Spain
[4]Aalto University, Finland
[5]Max Planck Institute for Security and Privacy, Germany

@EUROCRYPT 2023

(Round-Optimal) Two-Party Computation (2PC)

Alice ("Receiver")		**Bob ("Sender")**
$\mathcal{A}(f, \mathbf{x} \in \{0,1\}^*)$		$\mathcal{B}(f, \mathbf{y} \in \{0,1\}^*)$
	$\xrightarrow{\text{Encode}(\mathbf{x})}$	
$f(\mathbf{x}, \mathbf{y}) \leftarrow \text{Eval}(f, \text{Encode}(\mathbf{x}), \text{Encode}(\mathbf{y}))$	$\xleftarrow{\text{Encode}(\mathbf{y})}$	

- Alice and Bob have some function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$.
- Alice has input $\mathbf{x} \in \{0,1\}^*$, Bob has input $\mathbf{y} \in \{0,1\}^*$.
- After the 2PC,
  - Alice should learn (i.e. "receive") $f(\mathbf{x}, \mathbf{y})$, but nothing else about $\mathbf{y}$, and
  - Bob should nothing about $\mathbf{x}$.
- In general, Alice and Bob could each represents a member of a large group.

Secure Delegation: "Alice-Optimised" 2PC

Let FHE $=$ FHE.(KGen, Enc, Dec, Eval) be a fully homomorphic encryption scheme.

Alice ("Receiver")		**Bob ("Sender")**
$\underline{\mathcal{A}(f, \mathbf{x} \in \{0,1\}^*)}$		$\underline{\mathcal{B}(f, \mathbf{y} \in \{0,1\}^*)}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KGen}(1^\lambda)$		
$\mathsf{ctxt}_{\mathbf{x}} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, \mathbf{x})$	$\xrightarrow{\mathsf{pk}, \mathsf{ctxt}_{\mathbf{x}}}$	$f_{\mathbf{y}}(\cdot) := f(\cdot, \mathbf{y})$
$f(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ctxt}_{f(\mathbf{x},\mathbf{y})})$	$\xleftarrow{\mathsf{ctxt}_{f(\mathbf{x},\mathbf{y})}}$	$\mathsf{ctxt}_{f(\mathbf{x},\mathbf{y})} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, f_{\mathbf{y}}, \mathsf{ctxt}_{\mathbf{x}})$

† Let $|\cdot|$ denote description size.

† Alice's work: $|\mathbf{x}| \ll |f| + |\mathbf{y}|$, i.e. "Alice-optimised".

† Bob's work: $|f| + |\mathbf{y}| \gg |\mathbf{x}|$.

† Can be done "efficiently" (say, atomic ops in ms) from standard lattice-based assumptions.

Secure Delegation: "Alice-Optimised" 2PC

Let FHE $=$ FHE.(KGen, Enc, Dec, Eval) be a fully homomorphic encryption scheme.

Alice ("Receiver")		**Bob ("Sender")**
$\underline{\mathcal{A}(f, \mathbf{x} \in \{0,1\}^*)}$		$\underline{\mathcal{B}(f, \mathbf{y} \in \{0,1\}^*)}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KGen}(1^\lambda)$		
$\mathsf{ctxt}_\mathbf{x} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, \mathbf{x})$	$\xrightarrow{\mathsf{pk}, \mathsf{ctxt}_\mathbf{x}}$	$f_\mathbf{y}(\cdot) := f(\cdot, \mathbf{y})$
$f(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ctxt}_{f(\mathbf{x},\mathbf{y})})$	$\xleftarrow{\mathsf{ctxt}_{f(\mathbf{x},\mathbf{y})}}$	$\mathsf{ctxt}_{f(\mathbf{x},\mathbf{y})} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, f_\mathbf{y}, \mathsf{ctxt}_\mathbf{x})$

- † Let $|\cdot|$ denote description size.
- † Alice's work: $|\mathbf{x}| \ll |f| + |\mathbf{y}|$, i.e. "Alice-optimised".
- † Bob's work: $|f| + |\mathbf{y}| \gg |\mathbf{x}|$.
- † Can be done "efficiently" (say, atomic ops in ms) from standard lattice-based assumptions.

Laconic Cryptography or "Reverse Delegation": "Bob-Optimised" 2PC

An emerging paradigm with numerous theoretical results [CDG+17; QWW18; DGI+19; DGGM19]

Alice ("Receiver")		**Bob ("Sender")**
$\underline{\mathcal{A}(f, \mathbf{x} \in \{0,1\}^*)}$		$\underline{\mathcal{B}(f, \mathbf{y} \in \{0,1\}^*)}$
	$\xrightarrow{\mathsf{Encode}(\mathbf{x})}$	
$f(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{Eval}(f, \mathsf{Encode}(\mathbf{x}), \mathsf{Encode}(\mathbf{y}))$	$\xleftarrow{\mathsf{Encode}(\mathbf{y})}$	

We want:

- † Bob's work: $|\mathbf{y}| \ll |f| + |\mathbf{x}|$, i.e. "Bob-optimised".
- † Alice's work: $|f| + |\mathbf{x}| \gg |\mathbf{y}|$.

Why is it challenging?

- † Bob does not even have time to read f or $\mathbf{x}$ in full.
- † $\mathsf{Encode}(\mathbf{y})$ needs to be just enough for Alice to unpack $f(\mathbf{x}, \mathbf{y})$.

Laconic Cryptography or "Reverse Delegation": "Bob-Optimised" 2PC

An emerging paradigm with numerous theoretical results [CDG+17; QWW18; DGI+19; DGGM19]

Alice ("Receiver")		**Bob ("Sender")**
$\mathcal{A}(f, \mathbf{x} \in \{0,1\}^*)$		$\mathcal{B}(f, \mathbf{y} \in \{0,1\}^*)$
	$\xrightarrow{\mathsf{Encode}(\mathbf{x})}$	
$f(\mathbf{x}, \mathbf{y}) \leftarrow \mathsf{Eval}(f, \mathsf{Encode}(\mathbf{x}), \mathsf{Encode}(\mathbf{y}))$	$\xleftarrow{\mathsf{Encode}(\mathbf{y})}$	

We want:

- Bob's work: $|\mathbf{y}| \ll |f| + |\mathbf{x}|$, i.e. "Bob-optimised".
- Alice's work: $|f| + |\mathbf{x}| \gg |\mathbf{y}|$.

Why is it challenging?

- Bob does not even have time to read f or $\mathbf{x}$ in full.
- $\mathsf{Encode}(\mathbf{y})$ needs to be just enough for Alice to unpack $f(\mathbf{x}, \mathbf{y})$.

Applications to Non-Laconic Cryptography

(The techniques used for) laconic cryptography had led to:

- † **Identity-based encryption (IBE)** from weaker assumptions [DG17b; DG17a; DGHM18; BLSV18]
- † Two-round **multiparty computation (MPC)** from minimal assumptions [GS17; GS18c; BL18]
- † Adaptively secure **garbled circuits** from weaker assumptions [GS18b]
- † **Trapdoor functions** from weaker assumptions [GH18]
- † Simpler **indistinguishability obfuscation (iO)** for Turing machines [GS18a]
- † Single-server **private-information retrieval (PIR)** from weaker assumptions [DGI+19]

Laconic Cryptography Examples and "Non-Blackbox Barrier"

Primitive	$\mathbf{x}$	$\mathbf{y}$	$f(\mathbf{x}, \mathbf{y})$
L. Oblivious Transfer [CDG+17]	Big database $D \in \{0,1\}^n$	Index $i \in [n]$, messages μ_0, μ_1	μ_0 if $D[i] = 0$; μ_1 if $D[i] = 1$
L. Priv. Set Intersection [ABD+21; ALOS22]	Big set A	Small set B	$A \cap B$
L. Function Evaluation [QWW18]	Function g	Input $\mathbf{y}$	$g(\mathbf{y})$
Reg.-based Encryption [GHMR18; GHM+19; GV20] **L. Encryption (this work)**	$(\mathsf{sk}_{\mathsf{id}^\dagger}, (\mathsf{id}, \mathsf{pk}_{\mathsf{id}})_{\mathsf{id} \in R})$	Identity id^*, message μ	μ if $\mathsf{id}^\dagger = \mathsf{id}^* \in R$; $\perp$ otherwise

Before this work, all of the above requires "non-blackbox" techniques,

e.g. homomorphically evaluate a circuit implementing a public-key encryption

$\implies$ Completely impractical

Laconic Cryptography Examples and "Non-Blackbox Barrier"

Primitive	$\mathbf{x}$	$\mathbf{y}$	$f(\mathbf{x}, \mathbf{y})$
L. Oblivious Transfer [CDG+17]	Big database $D \in \{0,1\}^n$	Index $i \in [n]$, messages μ_0, μ_1	μ_0 if $D[i] = 0$; μ_1 if $D[i] = 1$
L. Priv. Set Intersection [ABD+21; ALOS22]	Big set A	Small set B	$A \cap B$
L. Function Evaluation [QWW18]	Function g	Input $\mathbf{y}$	$g(\mathbf{y})$
Reg.-based Encryption [GHMR18; GHM+19; GV20] **L. Encryption (this work)**	$(\mathsf{sk}_{\mathsf{id}^\dagger}, (\mathsf{id}, \mathsf{pk}_{\mathsf{id}})_{\mathsf{id} \in R})$	Identity id^*, message μ	μ if $\mathsf{id}^\dagger = \mathsf{id}^* \in R$; $\perp$ otherwise

Before this work, all of the above requires "non-blackbox" techniques,
e.g. homomorphically evaluate a circuit implementing a public-key encryption
$\implies$ Completely impractical

Laconic Cryptography Examples and “Non-Blackbox Barrier”

Primitive	$\mathbf{x}$	$\mathbf{y}$	$f(\mathbf{x}, \mathbf{y})$
L. Oblivious Transfer [CDG+17]	Big database $D \in \{0,1\}^n$	Index $i \in [n]$, messages μ_0, μ_1	μ_0 if $D[i] = 0$; μ_1 if $D[i] = 1$
L. Priv. Set Intersection [ABD+21; ALOS22]	Big set A	Small set B	$A \cap B$
L. Function Evaluation [QWW18]	Function g	Input $\mathbf{y}$	$g(\mathbf{y})$
Reg.-based Encryption [GHMR18; GHM+19; GV20] **L. Encryption (this work)**	$(\mathsf{sk}_{\mathsf{id}^\dagger}, (\mathsf{id}, \mathsf{pk}_{\mathsf{id}})_{\mathsf{id} \in R})$	Identity id^*, message μ	μ if $\mathsf{id}^\dagger = \mathsf{id}^* \in R$; $\perp$ otherwise

Before this work, all of the above requires “non-blackbox” techniques,
e.g. homomorphically evaluate a circuit implementing a public-key encryption
$\implies$ Completely impractical

Laconic Cryptography Examples and "Non-Blackbox Barrier"

Primitive	$\mathbf{x}$	$\mathbf{y}$	$f(\mathbf{x}, \mathbf{y})$
L. Oblivious Transfer [CDG+17]	Big database $D \in \{0,1\}^n$	Index $i \in [n]$, messages μ_0, μ_1	μ_0 if $D[i] = 0$; μ_1 if $D[i] = 1$
L. Priv. Set Intersection [ABD+21; ALOS22]	Big set A	Small set B	$A \cap B$
L. Function Evaluation [QWW18]	Function g	Input $\mathbf{y}$	$g(\mathbf{y})$
Reg.-based Encryption [GHMR18; GHM+19; GV20] **L. Encryption (this work)**	$(\mathsf{sk}_{\mathsf{id}^\dagger}, (\mathsf{id}, \mathsf{pk}_{\mathsf{id}})_{\mathsf{id} \in R})$	Identity id^*, message μ	μ if $\mathsf{id}^\dagger = \mathsf{id}^* \in R$; $\perp$ otherwise

Before this work, all of the above requires "non-blackbox" techniques,
e.g. homomorphically evaluate a circuit implementing a public-key encryption
$\implies$ Completely impractical

Laconic Cryptography Examples and "Non-Blackbox Barrier"

Primitive	$\mathbf{x}$	$\mathbf{y}$	$f(\mathbf{x}, \mathbf{y})$
L. Oblivious Transfer [CDG+17]	Big database $D \in \{0,1\}^n$	Index $i \in [n]$, messages μ_0, μ_1	μ_0 if $D[i] = 0$; μ_1 if $D[i] = 1$
L. Priv. Set Intersection [ABD+21; ALOS22]	Big set A	Small set B	$A \cap B$
L. Function Evaluation [QWW18]	Function g	Input $\mathbf{y}$	$g(\mathbf{y})$
Reg.-based Encryption [GHMR18; GHM+19; GV20] **L. Encryption (this work)**	$(\mathsf{sk}_{\mathsf{id}^\dagger}, (\mathsf{id}, \mathsf{pk}_{\mathsf{id}})_{\mathsf{id} \in R})$	Identity id^*, message μ	μ if $\mathsf{id}^\dagger = \mathsf{id}^* \in R$; $\perp$ otherwise

Before this work, all of the above requires "non-blackbox" techniques,

e.g. homomorphically evaluate a circuit implementing a public-key encryption

$\Longrightarrow$ Completely impractical

Our Results

- Introduce “laconic encryption” (LE) as central building block of laconic cryptography
  LE $=$ RBE without stringent update efficiency requirements
- Algebraic construction of LE from learning with errors (LWE) w/ poly. modulus-to-noise ratio
- Blackbox transformations LE $\rightarrow$ LOT, LPSI, and RBE
- Open-source implementation of LE (first of anything laconic):
  For database with at most 2^{50} identities, 10ms encrypt/decrypt time
- By-product: Identity-based encryption (IBE) w/ unbounded ID space and tight reduction from LWE

Our Results

- **Introduce "laconic encryption" (LE) as central building block of laconic cryptography**
  LE $=$ RBE without stringent update efficiency requirements
- **Algebraic construction of LE from learning with errors (LWE) w/ poly. modulus-to-noise ratio**
- Blackbox transformations LE $\rightarrow$ LOT, LPSI, and RBE
- Open-source implementation of LE (first of anything laconic):
  For database with at most 2^{50} identities, 10ms encrypt/decrypt time
- By-product: Identity-based encryption (IBE) w/ unbounded ID space and tight reduction from LWE

Laconic Encryption in a Nutshell

- Implicit in existing LOT constructions
- For simplicity, consider static array of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ (actual construction allows dynamic and efficient updates)

Alice ("Receiver")

$\mathcal{A}(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$

Bob ("Sender")

$\mathcal{B}(\mathsf{id}, \mu)$

$\mathsf{digest} \leftarrow \mathsf{Hash}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ — digest →

Pseudorandom (hiding both id and μ) without $\mathsf{sk}_{\mathsf{id}}$

$\mathsf{wit}_{\mathsf{id}} \leftarrow$ Witness of "$\mathsf{pk}_{\mathsf{id}} \in \mathsf{digest}$" — ← $\mathsf{ctxt}_{\mathsf{id},\mu}$ — $\underbrace{\mathsf{ctxt}_{\mathsf{id},\mu} \leftarrow \mathsf{LE.Enc}(\mathsf{digest}, \mathsf{id}, \mu)}_{\text{Time} \ll n}$

$\underbrace{\mu \leftarrow \mathsf{LE.Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{wit}_{\mathsf{id}}, \mathsf{ctxt}_{\mathsf{id},\mu})}_{\text{Time} \ll n}$

Laconic Encryption in a Nutshell

- Implicit in existing LOT constructions
- For simplicity, consider static array of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$
  (actual construction allows dynamic and efficient updates)

Alice ("Receiver")

$\mathcal{A}(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$

Bob ("Sender")

$\mathcal{B}(\mathsf{id}, \mu)$

$\mathsf{digest} \leftarrow \mathsf{Hash}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ —digest→

$\mathsf{wit}_{\mathsf{id}} \leftarrow$ Witness of "$\mathsf{pk}_{\mathsf{id}} \in \mathsf{digest}$"

Pseudorandom (hiding both id and μ) without $\mathsf{sk}_{\mathsf{id}}$

←$\mathsf{ctxt}_{\mathsf{id},\mu}$— $\mathsf{ctxt}_{\mathsf{id},\mu} \leftarrow \mathsf{LE.Enc}(\mathsf{digest}, \mathsf{id}, \mu)$ (Time $\ll n$)

$\mu \leftarrow \mathsf{LE.Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{wit}_{\mathsf{id}}, \mathsf{ctxt}_{\mathsf{id},\mu})$ (Time $\ll n$)

Laconic Encryption in a Nutshell

- Implicit in existing LOT constructions
- For simplicity, consider static array of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$
  (actual construction allows dynamic and efficient updates)

Alice ("Receiver") — $\underline{\mathcal{A}(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)}$

Bob ("Sender") — $\underline{\mathcal{B}(\mathsf{id}, \mu)}$

$\mathsf{digest} \leftarrow \mathsf{Hash}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ $\xrightarrow{\mathsf{digest}}$

Pseudorandom (hiding both id and μ) without $\mathsf{sk}_{\mathsf{id}}$

$\mathsf{wit}_{\mathsf{id}} \leftarrow$ Witness of "$\mathsf{pk}_{\mathsf{id}} \in \mathsf{digest}$" $\xleftarrow{\mathsf{ctxt}_{\mathsf{id},\mu}}$ $\underbrace{\mathsf{ctxt}_{\mathsf{id},\mu} \leftarrow \mathsf{LE.Enc}(\mathsf{digest}, \mathsf{id}, \mu)}_{\text{Time} \ll n}$

$\underbrace{\mu \leftarrow \mathsf{LE.Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{wit}_{\mathsf{id}}, \mathsf{ctxt}_{\mathsf{id},\mu})}_{\text{Time} \ll n}$

Laconic Encryption in a Nutshell

- Implicit in existing LOT constructions
- For simplicity, consider static array of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$
  (actual construction allows dynamic and efficient updates)

Alice ("Receiver")		**Bob ("Sender")**
$\mathcal{A}(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$		$\mathcal{B}(\mathsf{id}, \mu)$
$\mathsf{digest} \leftarrow \mathsf{Hash}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$	$\xrightarrow{\mathsf{digest}}$	
$\mathsf{wit}_{\mathsf{id}} \leftarrow$ Witness of "$\mathsf{pk}_{\mathsf{id}} \in \mathsf{digest}$"	Pseudorandom (hiding both id and μ) without $\mathsf{sk}_{\mathsf{id}}$ $\xleftarrow{\mathsf{ctxt}_{\mathsf{id},\mu}}$	$\underbrace{\mathsf{ctxt}_{\mathsf{id},\mu} \leftarrow \mathsf{LE.Enc}(\mathsf{digest}, \mathsf{id}, \mu)}_{\text{Time } \ll n}$
$\underbrace{\mu \leftarrow \mathsf{LE.Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{wit}_{\mathsf{id}}, \mathsf{ctxt}_{\mathsf{id},\mu})}_{\text{Time } \ll n}$		

Laconic Encryption in a Nutshell

- † Implicit in existing LOT constructions
- † For simplicity, consider static array of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ (actual construction allows dynamic and efficient updates)

Alice ("Receiver") — $\mathcal{A}(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$

Bob ("Sender") — $\mathcal{B}(\mathsf{id}, \mu)$

$\mathsf{digest} \leftarrow \mathsf{Hash}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ $\xrightarrow{\mathsf{digest}}$

$\mathsf{wit}_{\mathsf{id}} \leftarrow$ Witness of "$\mathsf{pk}_{\mathsf{id}} \in \mathsf{digest}$"

$\xleftarrow{\overbrace{\mathsf{ctxt}_{\mathsf{id},\mu}}^{\text{Pseudorandom (hiding both id and } \mu\text{) without } \mathsf{sk}_{\mathsf{id}}}}$

$\underbrace{\mathsf{ctxt}_{\mathsf{id},\mu} \leftarrow \mathsf{LE.Enc}(\mathsf{digest}, \mathsf{id}, \mu)}_{\text{Time } \ll n}$

$\underbrace{\mu \leftarrow \mathsf{LE.Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{wit}_{\mathsf{id}}, \mathsf{ctxt}_{\mathsf{id},\mu})}_{\text{Time } \ll n}$

Our Approach

Natural Construction Idea

- Pick favourite public-key encryption scheme
- Design "encryption-friendly" hash function
- Encrypt w.r.t. the statement "$\mathsf{pk}_{\mathsf{id}} \in$ digest", so that can decrypt with $(\mathsf{sk}_{\mathsf{id}}, \mathsf{wit}_{\mathsf{id}})$

Concrete Instantiation

- In Dual-Regev encryption [GPV08], $\mathsf{pp} = \mathbf{B}$, $\mathsf{pk} = \mathbf{y}$, $\mathsf{sk} = \mathbf{x}$ short vector satisfying

$$\mathbf{B} \cdot \mathbf{x} = \mathbf{y} \bmod q \quad \text{and} \quad \|\mathbf{x}\| \ll q \qquad \text{i.e. short integer solution (SIS) relation}$$

- Design encryption-friendly hash function Hash, so that $\mathsf{wit}_{\mathsf{id}} = \mathbf{w}_{\mathsf{id}}$ such that

$$\mathbf{B}_{\mathsf{Hash},\mathsf{id}} \cdot \begin{pmatrix} \mathbf{w}_{\mathsf{id}} \\ \mathbf{x}_{\mathsf{id}} \end{pmatrix} = \mathbf{y}_{\mathsf{digest}} \bmod q \qquad \text{and} \qquad \left\| \begin{pmatrix} \mathbf{w}_{\mathsf{id}} \\ \mathbf{x}_{\mathsf{id}} \end{pmatrix} \right\| \ll q$$

- Use Dual-Regev encryption to encrypt w.r.t. $(\mathbf{B}_{\mathsf{Hash},\mathsf{id}}, \mathbf{y}_{\mathsf{digest}})$

Our Approach

Natural Construction Idea

- Pick favourite public-key encryption scheme
- Design "encryption-friendly" hash function
- Encrypt w.r.t. the statement "$\mathsf{pk}_{\mathsf{id}} \in$ digest", so that can decrypt with $(\mathsf{sk}_{\mathsf{id}}, \mathsf{wit}_{\mathsf{id}})$

Concrete Instantiation

- In Dual-Regev encryption [GPV08], pp $= \mathbf{B}$, pk $= \mathbf{y}$, sk $= \mathbf{x}$ short vector satisfying

$$\mathbf{B} \cdot \mathbf{x} = \mathbf{y} \bmod q \quad \text{and} \quad \|\mathbf{x}\| \ll q \quad \text{i.e. short integer solution (SIS) relation}$$

- Design encryption-friendly hash function Hash, so that $\mathsf{wit}_{\mathsf{id}} = \mathbf{w}_{\mathsf{id}}$ such that

$$\mathbf{B}_{\mathsf{Hash},\mathsf{id}} \cdot \begin{pmatrix} \mathbf{w}_{\mathsf{id}} \\ \mathbf{x}_{\mathsf{id}} \end{pmatrix} = \mathbf{y}_{\mathsf{digest}} \bmod q \quad \text{and} \quad \left\| \begin{pmatrix} \mathbf{w}_{\mathsf{id}} \\ \mathbf{x}_{\mathsf{id}} \end{pmatrix} \right\| \ll q$$

- Use Dual-Regev encryption to encrypt w.r.t. $(\mathbf{B}_{\mathsf{Hash},\mathsf{id}}, \mathbf{y}_{\mathsf{digest}})$

Dual-Regev Encryption [GPV08]

- Public parameters: $\mathsf{pp} = \mathbf{B}$ wide uniformly random matrix
- Public key: $\mathsf{pk} = \mathbf{y}$ vector
- Secret key: $\mathsf{sk} = \mathbf{x}$ *short* vector satisfying $\mathbf{B} \cdot \mathbf{x} = \mathbf{y} \bmod q$ and $\|\mathbf{x}\| \ll q$
- Encryption of μ: $\mathsf{ctxt} = (\mathbf{c}_0, c_1)$ where

$$\mathbf{c}_0^\mathsf{T} = \mathbf{s}^\mathsf{T} \cdot \mathbf{B} + \text{noise} \bmod q \qquad \text{and} \qquad c_1 = \mathbf{s}^\mathsf{T} \cdot \mathbf{y} + \mathsf{Encode}(\mu) + \text{noise} \bmod q$$

  for random LWE secret $\mathbf{s}$
- Decryption: Recover μ by decoding

$$\begin{aligned} c_1 - \mathbf{c}_0^\mathsf{T} \cdot \mathbf{x} &= \mathbf{s}^\mathsf{T} \cdot (\mathbf{y} - \mathbf{B} \cdot \mathbf{x}) + \mathsf{Encode}(\mu) + \text{noise} \bmod q \\ &= \mathsf{Encode}(\mu) + \text{noise} \bmod q \end{aligned}$$

- Pseudorandom ciphertext from LWE assumption, i.e. $((\mathbf{B} \mid \mathbf{y}), \mathbf{s}^\mathsf{T} \cdot (\mathbf{B} \mid \mathbf{y}) + \text{noise}) \approx \text{uniform}$
- Take home message: Allows encrypting w.r.t. a short integer solution (SIS) relation $(\mathbf{B}, \mathbf{y})$

Dual-Regev Encryption [GPV08]

- Public parameters: $\mathsf{pp} = \mathbf{B}$ wide uniformly random matrix
- Public key: $\mathsf{pk} = \mathbf{y}$ vector
- Secret key: $\mathsf{sk} = \mathbf{x}$ *short* vector satisfying $\mathbf{B} \cdot \mathbf{x} = \mathbf{y} \bmod q$ and $\|\mathbf{x}\| \ll q$
- Encryption of μ: $\mathsf{ctxt} = (\mathbf{c}_0, c_1)$ where

$$\mathbf{c}_0^\intercal = \mathbf{s}^\intercal \cdot \mathbf{B} + \text{noise} \bmod q \qquad \text{and} \qquad c_1 = \mathbf{s}^\intercal \cdot \mathbf{y} + \mathsf{Encode}(\mu) + \text{noise} \bmod q$$

  for random LWE secret $\mathbf{s}$

- Decryption: Recover μ by decoding

$$\begin{aligned} c_1 - \mathbf{c}_0^\intercal \cdot \mathbf{x} &= \mathbf{s}^\intercal \cdot (\mathbf{y} - \mathbf{B} \cdot \mathbf{x}) + \mathsf{Encode}(\mu) + \text{noise} \bmod q \\ &= \mathsf{Encode}(\mu) + \text{noise} \bmod q \end{aligned}$$

- Pseudorandom ciphertext from LWE assumption, i.e. $((\mathbf{B} \mid \mathbf{y}), \mathbf{s}^\intercal \cdot (\mathbf{B} \mid \mathbf{y}) + \text{noise}) \approx \text{uniform}$
- Take home message: Allows encrypting w.r.t. a short integer solution (SIS) relation $(\mathbf{B}, \mathbf{y})$

Dual-Regev Encryption [GPV08]

- Public parameters: $\mathsf{pp} = \mathbf{B}$ wide uniformly random matrix
- Public key: $\mathsf{pk} = \mathbf{y}$ vector
- Secret key: $\mathsf{sk} = \mathbf{x}$ *short* vector satisfying $\mathbf{B} \cdot \mathbf{x} = \mathbf{y} \bmod q$ and $\|\mathbf{x}\| \ll q$
- Encryption of μ: $\mathsf{ctxt} = (\mathbf{c}_0, c_1)$ where

$$\mathbf{c}_0^\top = \mathbf{s}^\top \cdot \mathbf{B} + \text{noise} \bmod q \qquad \text{and} \qquad c_1 = \mathbf{s}^\top \cdot \mathbf{y} + \mathsf{Encode}(\mu) + \text{noise} \bmod q$$

  for random LWE secret $\mathbf{s}$
- Decryption: Recover μ by decoding

$$\begin{aligned} c_1 - \mathbf{c}_0^\top \cdot \mathbf{x} &= \mathbf{s}^\top \cdot (\mathbf{y} - \mathbf{B} \cdot \mathbf{x}) + \mathsf{Encode}(\mu) + \text{noise} \bmod q \\ &= \mathsf{Encode}(\mu) + \text{noise} \bmod q \end{aligned}$$

- Pseudorandom ciphertext from LWE assumption, i.e. $((\mathbf{B} \mid \mathbf{y}), \mathbf{s}^\top \cdot (\mathbf{B} \mid \mathbf{y}) + \text{noise}) \approx \text{uniform}$
- Take home message: Allows encrypting w.r.t. a short integer solution (SIS) relation $(\mathbf{B}, \mathbf{y})$

Dual-Regev Encryption [GPV08]

- Public parameters: $\mathsf{pp} = \mathbf{B}$ wide uniformly random matrix
- Public key: $\mathsf{pk} = \mathbf{y}$ vector
- Secret key: $\mathsf{sk} = \mathbf{x}$ *short* vector satisfying $\mathbf{B} \cdot \mathbf{x} = \mathbf{y} \bmod q$ and $\|\mathbf{x}\| \ll q$
- Encryption of μ: $\mathsf{ctxt} = (\mathbf{c}_0, c_1)$ where

$$\mathbf{c}_0^\top = \mathbf{s}^\top \cdot \mathbf{B} + \text{noise} \bmod q \qquad \text{and} \qquad c_1 = \mathbf{s}^\top \cdot \mathbf{y} + \mathsf{Encode}(\mu) + \text{noise} \bmod q$$

  for random LWE secret $\mathbf{s}$
- Decryption: Recover μ by decoding

$$\begin{aligned} c_1 - \mathbf{c}_0^\top \cdot \mathbf{x} &= \mathbf{s}^\top \cdot (\mathbf{y} - \mathbf{B} \cdot \mathbf{x}) + \mathsf{Encode}(\mu) + \text{noise} \bmod q \\ &= \mathsf{Encode}(\mu) + \text{noise} \bmod q \end{aligned}$$

- Pseudorandom ciphertext from LWE assumption, i.e. $((\mathbf{B} \mid \mathbf{y}), \mathbf{s}^\top \cdot (\mathbf{B} \mid \mathbf{y}) + \text{noise}) \approx \text{uniform}$
- Take home message: Allows encrypting w.r.t. a short integer solution (SIS) relation $(\mathbf{B}, \mathbf{y})$

Dual-Regev Encryption [GPV08]

- Public parameters: $\mathsf{pp} = \mathbf{B}$ wide uniformly random matrix
- Public key: $\mathsf{pk} = \mathbf{y}$ vector
- Secret key: $\mathsf{sk} = \mathbf{x}$ *short* vector satisfying $\mathbf{B} \cdot \mathbf{x} = \mathbf{y} \bmod q$ and $\|\mathbf{x}\| \ll q$
- Encryption of μ: $\mathsf{ctxt} = (\mathbf{c}_0, c_1)$ where

$$\mathbf{c}_0^\top = \mathbf{s}^\top \cdot \mathbf{B} + \text{noise} \bmod q \qquad \text{and} \qquad c_1 = \mathbf{s}^\top \cdot \mathbf{y} + \mathsf{Encode}(\mu) + \text{noise} \bmod q$$

  for random LWE secret $\mathbf{s}$
- Decryption: Recover μ by decoding

$$\begin{aligned} c_1 - \mathbf{c}_0^\top \cdot \mathbf{x} &= \mathbf{s}^\top \cdot (\mathbf{y} - \mathbf{B} \cdot \mathbf{x}) + \mathsf{Encode}(\mu) + \text{noise} \bmod q \\ &= \mathsf{Encode}(\mu) + \text{noise} \bmod q \end{aligned}$$

- Pseudorandom ciphertext from LWE assumption, i.e. $((\mathbf{B} \mid \mathbf{y}), \mathbf{s}^\top \cdot (\mathbf{B} \mid \mathbf{y}) + \text{noise}) \approx \text{uniform}$
- Take home message: Allows encrypting w.r.t. a short integer solution (SIS) relation $(\mathbf{B}, \mathbf{y})$

Gadget Matrix and Binary Decomposition

Recall “gadget matrix” $\mathbf{G}$ [MP12]

$$\mathbf{G} = \begin{pmatrix} 1 & 2 & \dots & 2^{\lfloor \log q \rfloor - 1} & & & & \\ & & & & \ddots & & & \\ & & & & & 1 & 2 & \dots & 2^{\lfloor \log q \rfloor - 1} \end{pmatrix}$$

and let $\mathbf{G}^{-1}$ denote the “binary-decomposition operator”.

For any q-ary vector $\mathbf{v}$ as tall as $\mathbf{G}$, we have

$$\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{v}) = \mathbf{v}.$$

Encryption-Friendly Hash Function

$$\mathsf{Hash}(\mathbf{v}_0, \mathbf{v}_1) := \mathbf{A}_0 \cdot (-\mathbf{G}^{-1}(\mathbf{v}_0)) + \mathbf{A}_1 \cdot (-\mathbf{G}^{-1}(\mathbf{v}_1)) = \mathbf{A} \cdot \begin{pmatrix} -\mathbf{G}^{-1}(\mathbf{v}_0) \\ -\mathbf{G}^{-1}(\mathbf{v}_1) \end{pmatrix} \bmod q$$

$$\underbrace{\begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & & \\ \mathbf{G} & & \mathbf{A}_0 & \mathbf{A}_1 & \\ & & & \mathbf{G} & \mathbf{B} \end{pmatrix}}_{\mathbf{B}_{\mathsf{Hash},01}} \begin{pmatrix} \mathbf{w}_0 \\ \mathbf{w}_1 \\ \mathbf{w}_{00} \\ \mathbf{w}_{01} \\ \mathbf{x}_{01} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{y}_\epsilon \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}}_{\mathbf{y}_{\mathsf{digest}}} \bmod q$$

Encryption-Friendly Hash Function

$$\mathsf{Hash}(\mathbf{v}_0, \mathbf{v}_1) := \mathbf{A}_0 \cdot (-\mathbf{G}^{-1}(\mathbf{v}_0)) + \mathbf{A}_1 \cdot (-\mathbf{G}^{-1}(\mathbf{v}_1)) = \mathbf{A} \cdot \begin{pmatrix} -\mathbf{G}^{-1}(\mathbf{v}_0) \\ -\mathbf{G}^{-1}(\mathbf{v}_1) \end{pmatrix} \bmod q$$

$$\text{digest} = \mathbf{y}_\epsilon = \mathbf{A} \begin{pmatrix} \mathbf{w}_0 \\ \mathbf{w}_1 \end{pmatrix}$$

$\mathbf{w}_0 = -\mathbf{G}^{-1}(\mathbf{y}_0)$ $\qquad$ $\mathbf{w}_1 = -\mathbf{G}^{-1}(\mathbf{y}_1)$

$$\mathbf{y}_0 = \mathbf{A} \begin{pmatrix} \mathbf{w}_{00} \\ \mathbf{w}_{01} \end{pmatrix} \qquad \mathbf{y}_1 = \mathbf{A} \begin{pmatrix} \mathbf{w}_{10} \\ \mathbf{w}_{11} \end{pmatrix}$$

$\mathbf{w}_{00} = -\mathbf{G}^{-1}(\mathbf{y}_{00})$ $\qquad$ $\mathbf{w}_{01} = -\mathbf{G}^{-1}(\mathbf{y}_{01})$ $\qquad$ $\mathbf{w}_{10} = -\mathbf{G}^{-1}(\mathbf{y}_{10})$ $\qquad$ $\mathbf{w}_{11} = -\mathbf{G}^{-1}(\mathbf{y}_{11})$

$$\mathbf{y}_{00} = \mathbf{B}\mathbf{x}_{00} \qquad \underbrace{\mathbf{y}_{01}}_{\mathsf{pk}_{01}} = \mathbf{B}\underbrace{\mathbf{x}_{01}}_{\mathsf{sk}_{01}} \qquad \mathbf{y}_{10} = \mathbf{B}\mathbf{x}_{10} \qquad \mathbf{y}_{11} = \mathbf{B}\mathbf{x}_{11}$$

$$\underbrace{\begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & & \\ \mathbf{G} & & \mathbf{A}_0 & \mathbf{A}_1 & \\ & & & \mathbf{G} & \mathbf{B} \end{pmatrix}}_{\mathbf{B}_{\mathsf{Hash},01}} \begin{pmatrix} \mathbf{w}_0 \\ \mathbf{w}_1 \\ \mathbf{w}_{00} \\ \mathbf{w}_{01} \\ \mathbf{x}_{01} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{y}_\epsilon \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}}_{\mathbf{y}_{\mathsf{digest}}} \bmod q$$

Encryption-Friendly Hash Function

$$\mathsf{Hash}(\mathbf{v}_0, \mathbf{v}_1) := \mathbf{A}_0 \cdot (-\mathbf{G}^{-1}(\mathbf{v}_0)) + \mathbf{A}_1 \cdot (-\mathbf{G}^{-1}(\mathbf{v}_1)) = \mathbf{A} \cdot \begin{pmatrix} -\mathbf{G}^{-1}(\mathbf{v}_0) \\ -\mathbf{G}^{-1}(\mathbf{v}_1) \end{pmatrix} \bmod q$$

$$\mathsf{digest} = \mathbf{y}_\epsilon = \mathbf{A} \begin{pmatrix} \mathbf{w}_0 \\ \mathbf{w}_1 \end{pmatrix}$$

$$\mathbf{w}_0 = -\mathbf{G}^{-1}(\mathbf{y}_0) \qquad \mathbf{w}_1 = -\mathbf{G}^{-1}(\mathbf{y}_1)$$

$$\mathbf{y}_0 = \mathbf{A} \begin{pmatrix} \mathbf{w}_{00} \\ \mathbf{w}_{01} \end{pmatrix} \qquad \mathbf{y}_1 = \mathbf{A} \begin{pmatrix} \mathbf{w}_{10} \\ \mathbf{w}_{11} \end{pmatrix}$$

$$\mathbf{w}_{00} = -\mathbf{G}^{-1}(\mathbf{y}_{00}) \qquad \mathbf{w}_{01} = -\mathbf{G}^{-1}(\mathbf{y}_{01}) \qquad \mathbf{w}_{10} = -\mathbf{G}^{-1}(\mathbf{y}_{10}) \qquad \mathbf{w}_{11} = -\mathbf{G}^{-1}(\mathbf{y}_{11})$$

$$\mathbf{y}_{00} = \mathbf{B}\mathbf{x}_{00} \qquad \underbrace{\mathbf{y}_{01}}_{\mathsf{pk}_{01}} = \mathbf{B}\,\underbrace{\mathbf{x}_{01}}_{\mathsf{sk}_{01}} \qquad \mathbf{y}_{10} = \mathbf{B}\mathbf{x}_{10} \qquad \mathbf{y}_{11} = \mathbf{B}\mathbf{x}_{11}$$

$$\underbrace{\begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & & \\ \mathbf{G} & & \mathbf{A}_0 & \mathbf{A}_1 & \\ & & & \mathbf{G} & \mathbf{B} \end{pmatrix}}_{\mathbf{B}_{\mathsf{Hash},01}} \begin{pmatrix} \mathbf{w}_0 \\ \mathbf{w}_1 \\ \mathbf{w}_{00} \\ \mathbf{w}_{01} \\ \mathbf{x}_{01} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{y}_\epsilon \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}}_{\mathbf{y}_{\mathsf{digest}}} \bmod q$$

Putting Everything Together

- † User id has public key $\mathsf{pk}_{\mathsf{id}} = \mathbf{y}_{\mathsf{id}}$ and secret key $\mathsf{sk}_{\mathsf{id}} = \mathbf{x}_{\mathsf{id}}$
- † Digest $\mathsf{digest} = \mathbf{y}_{\epsilon} =$ Merkle-tree hash of $(\mathsf{pk}_{\mathsf{id}}) = (\mathbf{y}_{\mathsf{id}})_{\mathsf{id}}$
- † Witness $\mathsf{wit}_{\mathsf{id}} =$ concatenation of all $\mathbf{w}$'s along the root-to-id path and their siblings, e.g.

$$\mathsf{wit}_{01}^{\mathsf{T}} = (\mathbf{w}_0^{\mathsf{T}}, \mathbf{w}_1^{\mathsf{T}}, \mathbf{w}_{00}^{\mathsf{T}}, \mathbf{w}_{01}^{\mathsf{T}})$$

- † To encrypt μ to $\mathsf{id} \in \{0,1\}^{\ell}$, use Dual-Regev to encrypt w.r.t. $(\mathbf{B}_{\mathsf{Hash},\mathsf{id}}, \mathbf{y}_{\mathsf{digest}})$ where

$$\mathbf{B}_{\mathsf{Hash},\mathsf{id}} = \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & & & & & & \\ \bar{\mathsf{id}}_1\mathbf{G} & \mathsf{id}_1\mathbf{G} & \mathbf{A}_0 & \mathbf{A}_1 & & & & & \\ & & \bar{\mathsf{id}}_2\mathbf{G} & \mathsf{id}_2\mathbf{G} & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \mathbf{A}_0 & \mathbf{A}_1 & \\ & & & & & & \bar{\mathsf{id}}_{\ell}\mathbf{G} & \mathsf{id}_{\ell}\mathbf{G} & \mathbf{B} \end{pmatrix} \quad \text{and} \quad \mathbf{y}_{\mathsf{digest}} = \begin{pmatrix} \mathbf{y}_{\epsilon} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}$$

- † Decrypt using $(\mathsf{wit}_{\mathsf{id}}^{\mathsf{T}}, \mathbf{x}_{\mathsf{id}})$ as Dual-Regev secret key, where $\mathbf{B}_{\mathsf{Hash},\mathsf{id}} \cdot \begin{pmatrix} \mathsf{wit}_{\mathsf{id}} \\ \mathbf{x}_{\mathsf{id}} \end{pmatrix} = \mathbf{y}_{\mathsf{digest}} \bmod q$

Putting Everything Together

- User id has public key $\mathsf{pk}_{\mathsf{id}} = \mathbf{y}_{\mathsf{id}}$ and secret key $\mathsf{sk}_{\mathsf{id}} = \mathbf{x}_{\mathsf{id}}$
- Digest $\mathsf{digest} = \mathbf{y}_\epsilon =$ Merkle-tree hash of $(\mathsf{pk}_{\mathsf{id}}) = (\mathbf{y}_{\mathsf{id}})_{\mathsf{id}}$
- Witness $\mathsf{wit}_{\mathsf{id}} =$ concatenation of all $\mathbf{w}$'s along the root-to-id path and their siblings, e.g.

$$\mathsf{wit}_{01}^{\mathsf{T}} = (\mathbf{w}_0^{\mathsf{T}}, \mathbf{w}_1^{\mathsf{T}}, \mathbf{w}_{00}^{\mathsf{T}}, \mathbf{w}_{01}^{\mathsf{T}})$$

- To encrypt μ to $\mathsf{id} \in \{0,1\}^\ell$, use Dual-Regev to encrypt w.r.t. $(\mathbf{B}_{\mathsf{Hash},\mathsf{id}}, \mathbf{y}_{\mathsf{digest}})$ where

$$\mathbf{B}_{\mathsf{Hash},\mathsf{id}} = \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & & & & & & \\ \overline{\mathsf{id}}_1\mathbf{G} & \mathsf{id}_1\mathbf{G} & \mathbf{A}_0 & \mathbf{A}_1 & & & & & \\ & & \overline{\mathsf{id}}_2\mathbf{G} & \mathsf{id}_2\mathbf{G} & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \mathbf{A}_0 & \mathbf{A}_1 & \\ & & & & & & \overline{\mathsf{id}}_\ell\mathbf{G} & \mathsf{id}_\ell\mathbf{G} & \mathbf{B} \end{pmatrix} \quad \text{and} \quad \mathbf{y}_{\mathsf{digest}} = \begin{pmatrix} \mathbf{y}_\epsilon \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}$$

- Decrypt using $(\mathsf{wit}_{\mathsf{id}}^{\mathsf{T}}, \mathbf{x}_{\mathsf{id}})$ as Dual-Regev secret key, where $\mathbf{B}_{\mathsf{Hash},\mathsf{id}} \cdot \begin{pmatrix} \mathsf{wit}_{\mathsf{id}} \\ \mathbf{x}_{\mathsf{id}} \end{pmatrix} = \mathbf{y}_{\mathsf{digest}} \bmod q$

Putting Everything Together

- User id has public key $\mathsf{pk}_{\mathsf{id}} = \mathbf{y}_{\mathsf{id}}$ and secret key $\mathsf{sk}_{\mathsf{id}} = \mathbf{x}_{\mathsf{id}}$
- Digest digest $= \mathbf{y}_\epsilon =$ Merkle-tree hash of $(\mathsf{pk}_{\mathsf{id}}) = (\mathbf{y}_{\mathsf{id}})_{\mathsf{id}}$
- Witness $\mathsf{wit}_{\mathsf{id}} =$ concatenation of all $\mathbf{w}$'s along the root-to-id path and their siblings, e.g.

$$\mathsf{wit}_{01}^{\mathsf{T}} = (\mathbf{w}_0^{\mathsf{T}}, \mathbf{w}_1^{\mathsf{T}}, \mathbf{w}_{00}^{\mathsf{T}}, \mathbf{w}_{01}^{\mathsf{T}})$$

- To encrypt μ to $\mathsf{id} \in \{0,1\}^\ell$, use Dual-Regev to encrypt w.r.t. $(\mathbf{B}_{\mathsf{Hash},\mathsf{id}}, \mathbf{y}_{\mathsf{digest}})$ where

$$\mathbf{B}_{\mathsf{Hash},\mathsf{id}} = \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & & & & & & \\ \bar{\mathsf{id}}_1\mathbf{G} & \mathsf{id}_1\mathbf{G} & \mathbf{A}_0 & \mathbf{A}_1 & & & & & \\ & & \bar{\mathsf{id}}_2\mathbf{G} & \mathsf{id}_2\mathbf{G} & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \mathbf{A}_0 & \mathbf{A}_1 & \\ & & & & & & \bar{\mathsf{id}}_\ell\mathbf{G} & \mathsf{id}_\ell\mathbf{G} & \mathbf{B} \end{pmatrix} \quad \text{and} \quad \mathbf{y}_{\mathsf{digest}} = \begin{pmatrix} \mathbf{y}_\epsilon \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}$$

- Decrypt using $(\mathsf{wit}_{\mathsf{id}}^{\mathsf{T}}, \mathbf{x}_{\mathsf{id}})$ as Dual-Regev secret key, where $\mathbf{B}_{\mathsf{Hash},\mathsf{id}} \cdot \begin{pmatrix} \mathsf{wit}_{\mathsf{id}} \\ \mathbf{x}_{\mathsf{id}} \end{pmatrix} = \mathbf{y}_{\mathsf{digest}} \bmod q$

Putting Everything Together

- User id has public key $\mathsf{pk}_{\mathsf{id}} = \mathbf{y}_{\mathsf{id}}$ and secret key $\mathsf{sk}_{\mathsf{id}} = \mathbf{x}_{\mathsf{id}}$
- Digest $\mathsf{digest} = \mathbf{y}_\epsilon$ = Merkle-tree hash of $(\mathsf{pk}_{\mathsf{id}}) = (\mathbf{y}_{\mathsf{id}})_{\mathsf{id}}$
- Witness $\mathsf{wit}_{\mathsf{id}}$ = concatenation of all $\mathbf{w}$'s along the root-to-id path and their siblings, e.g.

$$\mathsf{wit}_{01}^{\mathsf{T}} = (\mathbf{w}_0^{\mathsf{T}}, \mathbf{w}_1^{\mathsf{T}}, \mathbf{w}_{00}^{\mathsf{T}}, \mathbf{w}_{01}^{\mathsf{T}})$$

- To encrypt μ to $\mathsf{id} \in \{0,1\}^\ell$, use Dual-Regev to encrypt w.r.t. $(\mathbf{B}_{\mathsf{Hash},\mathsf{id}}, \mathbf{y}_{\mathsf{digest}})$ where

$$\mathbf{B}_{\mathsf{Hash},\mathsf{id}} = \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & & & & & & \\ \overline{\mathsf{id}}_1\mathbf{G} & \mathsf{id}_1\mathbf{G} & \mathbf{A}_0 & \mathbf{A}_1 & & & & & \\ & & \overline{\mathsf{id}}_2\mathbf{G} & \mathsf{id}_2\mathbf{G} & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \mathbf{A}_0 & \mathbf{A}_1 & \\ & & & & & & \overline{\mathsf{id}}_\ell\mathbf{G} & \mathsf{id}_\ell\mathbf{G} & \mathbf{B} \end{pmatrix} \quad \text{and} \quad \mathbf{y}_{\mathsf{digest}} = \begin{pmatrix} \mathbf{y}_\epsilon \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}$$

- Decrypt using $(\mathsf{wit}_{\mathsf{id}}^{\mathsf{T}}, \mathbf{x}_{\mathsf{id}})$ as Dual-Regev secret key, where $\mathbf{B}_{\mathsf{Hash},\mathsf{id}} \cdot \begin{pmatrix} \mathsf{wit}_{\mathsf{id}} \\ \mathbf{x}_{\mathsf{id}} \end{pmatrix} = \mathbf{y}_{\mathsf{digest}} \bmod q$

Key Points in Security Proof

- † We cannot use security of Dual-Regev directly:
  $(\mathbf{B}_{\text{Hash,id}}, \mathbf{y}_{\text{digest}})$ is not a properly distributed Dual-Regev public key.
- † Instead, we need to argue layer-by-layer about pseudorandomness of LWE samples w.r.t.

$$(\mathbf{B}_{\text{Hash,id}}, \mathbf{y}_{\text{digest}}) = \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & & & & & & & \mathbf{y}_\epsilon \\ \bar{\text{id}}_1 \mathbf{G} & \text{id}_1 \mathbf{G} & \mathbf{A}_0 & \mathbf{A}_1 & & & & & & \mathbf{0} \\ & & \bar{\text{id}}_2 \mathbf{G} & \text{id}_2 \mathbf{G} & \ddots & \ddots & & & & \vdots \\ & & & & \ddots & \ddots & \mathbf{A}_0 & \mathbf{A}_1 & & \mathbf{0} \\ & & & & & & \bar{\text{id}}_\ell \mathbf{G} & \text{id}_\ell \mathbf{G} & \mathbf{B} & \mathbf{0} \end{pmatrix}.$$

- † Somewhere in the proof, we need to perform noise flooding/drowning/smudging/etc., i.e. show that

  {Linearly-correlated LWE samples} $\approx$ {Short linear combinations of LWE samples + fresh noise}

  Traditionally, this is either done by
  - ‡ using super-polynomial-size modulus $\Longrightarrow$ require LWE w/ super-polynomial modulus-to-noise ratio, or
  - ‡ arguing about Rényi divergence $\Longrightarrow$ incur polynomial reduction loss.

  We instead show a tight reduction from LWE w/ polynomial modulus-to-noise ratio.

Key Points in Security Proof

- † We cannot use security of Dual-Regev directly:
  $(\mathbf{B}_{\mathrm{Hash,id}}, \mathbf{y}_{\mathrm{digest}})$ is not a properly distributed Dual-Regev public key.
- † Instead, we need to argue layer-by-layer about pseudorandomness of LWE samples w.r.t.

$$(\mathbf{B}_{\mathrm{Hash,id}}, \mathbf{y}_{\mathrm{digest}}) = \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & & & & & & & \mathbf{y}_\epsilon \\ \bar{\mathrm{id}}_1\mathbf{G} & \mathrm{id}_1\mathbf{G} & \mathbf{A}_0 & \mathbf{A}_1 & & & & & & \mathbf{0} \\ & & \bar{\mathrm{id}}_2\mathbf{G} & \mathrm{id}_2\mathbf{G} & \ddots & \ddots & & & & \vdots \\ & & & & \ddots & \ddots & \mathbf{A}_0 & \mathbf{A}_1 & & \mathbf{0} \\ & & & & & & \bar{\mathrm{id}}_\ell\mathbf{G} & \mathrm{id}_\ell\mathbf{G} & \mathbf{B} & \mathbf{0} \end{pmatrix}.$$

- † Somewhere in the proof, we need to perform noise flooding/drowning/smudging/etc., i.e. show that

$$\{\text{Linearly-correlated LWE samples}\} \quad \approx \quad \{\text{Short linear combinations of LWE samples + fresh noise}\}$$

  Traditionally, this is either done by
  - ‡ using super-polynomial-size modulus $\implies$ require LWE w/ super-polynomial modulus-to-noise ratio, or
  - ‡ arguing about Rényi divergence $\implies$ incur polynomial reduction loss.

  We instead show a tight reduction from LWE w/ polynomial modulus-to-noise ratio.

Key Points in Security Proof

- † We cannot use security of Dual-Regev directly:
  $(\mathbf{B}_{\mathsf{Hash},\mathsf{id}}, \mathbf{y}_{\mathsf{digest}})$ is not a properly distributed Dual-Regev public key.
- † Instead, we need to argue layer-by-layer about pseudorandomness of LWE samples w.r.t.

$$(\mathbf{B}_{\mathsf{Hash},\mathsf{id}}, \mathbf{y}_{\mathsf{digest}}) = \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & & & & & & & \mathbf{y}_\epsilon \\ \overline{\mathsf{id}}_1\mathbf{G} & \mathsf{id}_1\mathbf{G} & \mathbf{A}_0 & \mathbf{A}_1 & & & & & & \mathbf{0} \\ & & \overline{\mathsf{id}}_2\mathbf{G} & \mathsf{id}_2\mathbf{G} & \ddots & \ddots & & & & \vdots \\ & & & & \ddots & \ddots & \mathbf{A}_0 & \mathbf{A}_1 & & \mathbf{0} \\ & & & & & & \overline{\mathsf{id}}_\ell\mathbf{G} & \mathsf{id}_\ell\mathbf{G} & \mathbf{B} & \mathbf{0} \end{pmatrix}.$$

- † Somewhere in the proof, we need to perform noise flooding/drowning/smudging/etc., i.e. show that

  $\{$Linearly-correlated LWE samples$\}$ $\approx$ $\{$Short linear combinations of LWE samples + fresh noise$\}$

  Traditionally, this is either done by
  - ‡ using super-polynomial-size modulus $\Longrightarrow$ require LWE w/ super-polynomial modulus-to-noise ratio, or
  - ‡ arguing about Rényi divergence $\Longrightarrow$ incur polynomial reduction loss.

  We instead show a tight reduction from LWE w/ polynomial modulus-to-noise ratio.

Conclusion

- Algebraic construction of laconic encryption (LE) from standard LWE
- Blackbox transformations LE $\rightarrow$ LOT, LPSI, and RBE
- Open-source implementation of LE:
  For database with at most 2^{50} identities, 10ms encrypt/decrypt time
- By-product: Identity-based encryption (IBE) w/ unbounded ID space and tight reduction from LWE

Code

https://github.com/ahmadrezarahimi/laconic-encryption

Paper

https://ia.cr/2023/404

Russell W. F. Lai
Aalto University, Finland
russell.lai@aalto.fi
russell-lai.hk

References I

[ABD+21] Navid Alamati et al. "Laconic Private Set Intersection and Applications". In: *TCC 2021, Part III*. Ed. by Kobbi Nissim and Brent Waters. Vol. 13044. LNCS. Springer, Heidelberg, Nov. 2021, pp. 94–125. DOI: 10.1007/978-3-030-90456-2_4.

[ALOS22] Diego Aranha et al. *Laconic Private Set-Intersection From Pairings*. Cryptology ePrint Archive, Report 2022/529. https://eprint.iacr.org/2022/529. 2022.

[BL18] Fabrice Benhamouda and Huijia Lin. "k-Round Multiparty Computation from k-Round Oblivious Transfer via Garbled Interactive Circuits". In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Springer, Heidelberg, 2018, pp. 500–532. DOI: 10.1007/978-3-319-78375-8_17.

[BLSV18] Zvika Brakerski et al. "Anonymous IBE, Leakage Resilience and Circular Security from New Assumptions". In: *EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. LNCS. Springer, Heidelberg, 2018, pp. 535–564. DOI: 10.1007/978-3-319-78381-9_20.

References II

[CDG+17] Chongwon Cho et al. “Laconic Oblivious Transfer and Its Applications”. In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Heidelberg, Aug. 2017, pp. 33–65. DOI: 10.1007/978-3-319-63715-0_2.

[DG17a] Nico Döttling and Sanjam Garg. “From Selective IBE to Full IBE and Selective HIBE”. In: *TCC 2017, Part I*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10677. LNCS. Springer, Heidelberg, Nov. 2017, pp. 372–408. DOI: 10.1007/978-3-319-70500-2_13.

[DG17b] Nico Döttling and Sanjam Garg. “Identity-Based Encryption from the Diffie-Hellman Assumption”. In: *CRYPTO 2017, Part I*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10401. LNCS. Springer, Heidelberg, Aug. 2017, pp. 537–569. DOI: 10.1007/978-3-319-63688-7_18.

[DGGM19] Nico Döttling et al. “Laconic Conditional Disclosure of Secrets and Applications”. In: *60th FOCS*. Ed. by David Zuckerman. IEEE Computer Society Press, Nov. 2019, pp. 661–685. DOI: 10.1109/FOCS.2019.00046.

References III

[DGHM18] Nico Döttling et al. “New Constructions of Identity-Based and Key-Dependent Message Secure Encryption Schemes”. In: *PKC 2018, Part I*. Ed. by Michel Abdalla and Ricardo Dahab. Vol. 10769. LNCS. Springer, Heidelberg, Mar. 2018, pp. 3–31. DOI: 10.1007/978-3-319-76578-5_1.

[DGI+19] Nico Döttling et al. “Trapdoor Hash Functions and Their Applications”. In: *CRYPTO 2019, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. LNCS. Springer, Heidelberg, Aug. 2019, pp. 3–32. DOI: 10.1007/978-3-030-26954-8_1.

[GH18] Sanjam Garg and Mohammad Hajiabadi. “Trapdoor Functions from the Computational Diffie-Hellman Assumption”. In: *CRYPTO 2018, Part II*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10992. LNCS. Springer, Heidelberg, Aug. 2018, pp. 362–391. DOI: 10.1007/978-3-319-96881-0_13.

[GHM+19] Sanjam Garg et al. “Registration-Based Encryption from Standard Assumptions”. In: *PKC 2019, Part II*. Ed. by Dongdai Lin and Kazue Sako. Vol. 11443. LNCS. Springer, Heidelberg, Apr. 2019, pp. 63–93. DOI: 10.1007/978-3-030-17259-6_3.

References IV

[GHMR18] Sanjam Garg et al. "Registration-Based Encryption: Removing Private-Key Generator from IBE". In: *TCC 2018, Part I*. Ed. by Amos Beimel and Stefan Dziembowski. Vol. 11239. LNCS. Springer, Heidelberg, Nov. 2018, pp. 689–718. DOI: 10.1007/978-3-030-03807-6_25.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions". In: *40th ACM STOC*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 197–206. DOI: 10.1145/1374376.1374407.

[GS17] Sanjam Garg and Akshayaram Srinivasan. "Garbled Protocols and Two-Round MPC from Bilinear Maps". In: *58th FOCS*. Ed. by Chris Umans. IEEE Computer Society Press, Oct. 2017, pp. 588–599. DOI: 10.1109/FOCS.2017.60.

[GS18a] Sanjam Garg and Akshayaram Srinivasan. "A Simple Construction of iO for Turing Machines". In: *TCC 2018, Part II*. Ed. by Amos Beimel and Stefan Dziembowski. Vol. 11240. LNCS. Springer, Heidelberg, Nov. 2018, pp. 425–454. DOI: 10.1007/978-3-030-03810-6_16.

References V

[GS18b] Sanjam Garg and Akshayaram Srinivasan. "Adaptively Secure Garbling with Near Optimal Online Complexity". In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Springer, Heidelberg, 2018, pp. 535–565. DOI: 10.1007/978-3-319-78375-8_18.

[GS18c] Sanjam Garg and Akshayaram Srinivasan. "Two-Round Multiparty Secure Computation from Minimal Assumptions". In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Springer, Heidelberg, 2018, pp. 468–499. DOI: 10.1007/978-3-319-78375-8_16.

[GV20] Rishab Goyal and Satyanarayana Vusirikala. "Verifiable Registration-Based Encryption". In: *CRYPTO 2020, Part I*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12170. LNCS. Springer, Heidelberg, Aug. 2020, pp. 621–651. DOI: 10.1007/978-3-030-56784-2_21.

[MP12] Daniele Micciancio and Chris Peikert. "Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller". In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 700–718. DOI: 10.1007/978-3-642-29011-4_41.

References VI

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. "Laconic Function Evaluation and Applications". In: *59th FOCS*. Ed. by Mikkel Thorup. IEEE Computer Society Press, Oct. 2018, pp. 859–870. DOI: 10.1109/FOCS.2018.00086.