LAPS

LATTICE-BASED PRIVATE-STREAM AGGREGATION

"REVISITING PRIVATE-STREAM AGGREGATION: LATTICE-BASED PSA"

NETWORK AND DISTRIBUTED SYSTEMS SECURITY (NDSS) SYMPOSIUM 2018

Daniela Becker
Robert Bosch LLC – RTC North America
Pittsburgh, USA

Jorge Guajardo
Robert Bosch LLC – RTC North America
Pittsburgh, USA

Karl-Heinz Zimmermann
Hamburg University of Technology
Hamburg, Germany

BOSCH

Outline

1. Introduction: Private Stream Aggregation (PSA)

   Problem Statement, Previous Work - Shi et al.'s PSA Scheme (NDSS 2011).

2. (Augmented) Learning With Errors

   Theory Background.

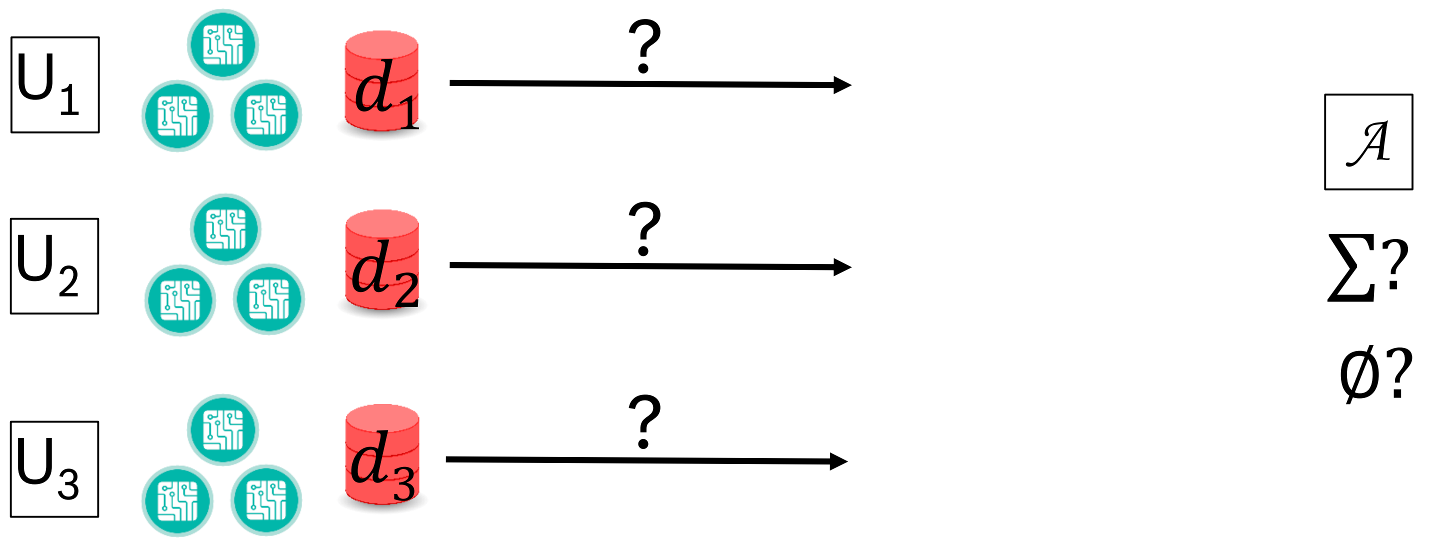
3. Lattice-Based PSA: LaPS

   - *General Construction.*
   - *LaPS instantiation & Experimental Results.*

4. Summary & Outlook

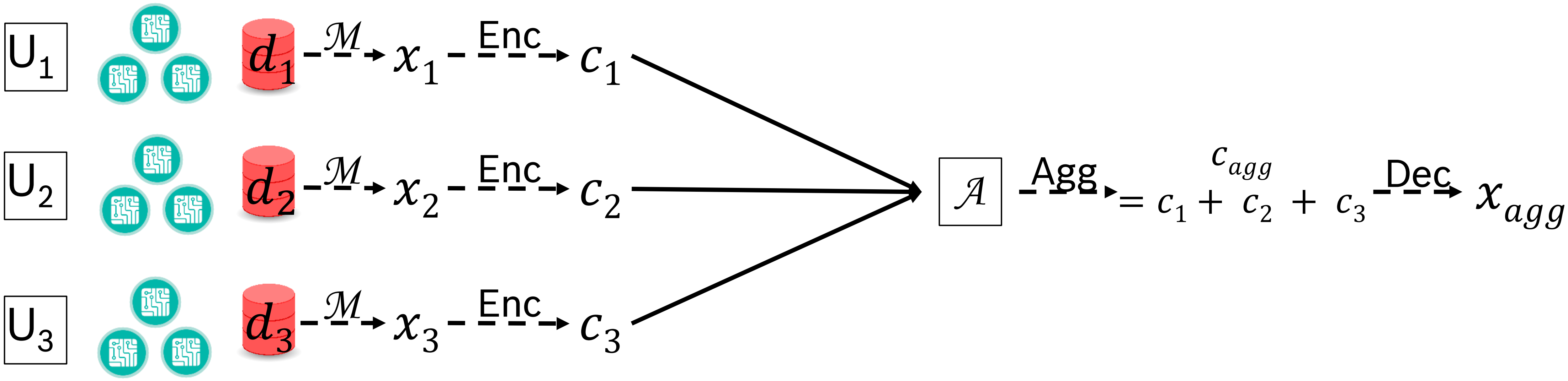
Private Stream Aggregation (PSA)
Problem

- **Distributed** set of users ({U_i}) want to compute **sum** of their sensitive data ({d_i})
- **No** information must be leaked about individual user U_i
- **Untrusted** aggregator ($\mathcal{A}$), i.e. honest-but-curious



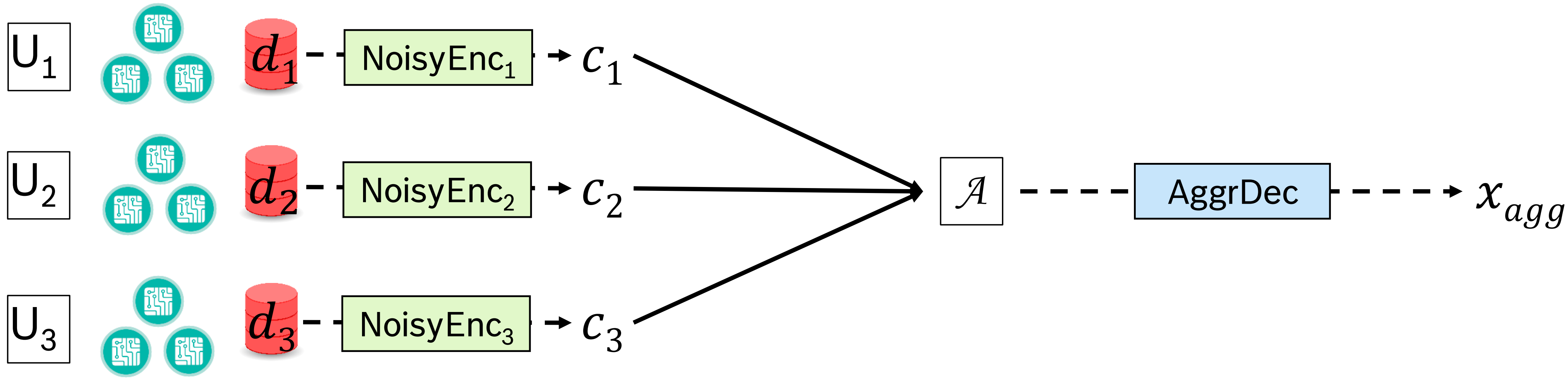
Private Stream Aggregation (PSA)
Solution

- Apply **differential privacy** mechanism $\mathcal{M}$ to each $d_i \Rightarrow$ create **noisy** version x_i
- Send **encrypted** x_i to aggregator $\mathcal{A}$
- $\mathcal{A}$ aggregates ciphertexts and decrypts – learns nothing but noisy sum x_{agg}



Private Stream Aggregation (PSA)
Security & Privacy Notions

- Aggregator **obliviousness** ↔ $\mathcal{A}$ learns **nothing but** noisy sum ⇒ x_{agg} differentially private



BOSCH

Private Stream Aggregation (PSA)

Shi et al. [1]

U_i - $(param, t, sk_i, d_i)$ — NoisyEnc$_i$

$$r_i \leftarrow Geom$$

$$x_i \leftarrow d_i + r_i$$

$$c_i \leftarrow g^{x_i} \cdot H(t)^{sk_i}$$

$\{c_i\}_{i=1}^{N}$

$\mathcal{A}$ - $(param, t, \{c_i\}_{i=1}^{N}, sk_0)$ — AggrDec

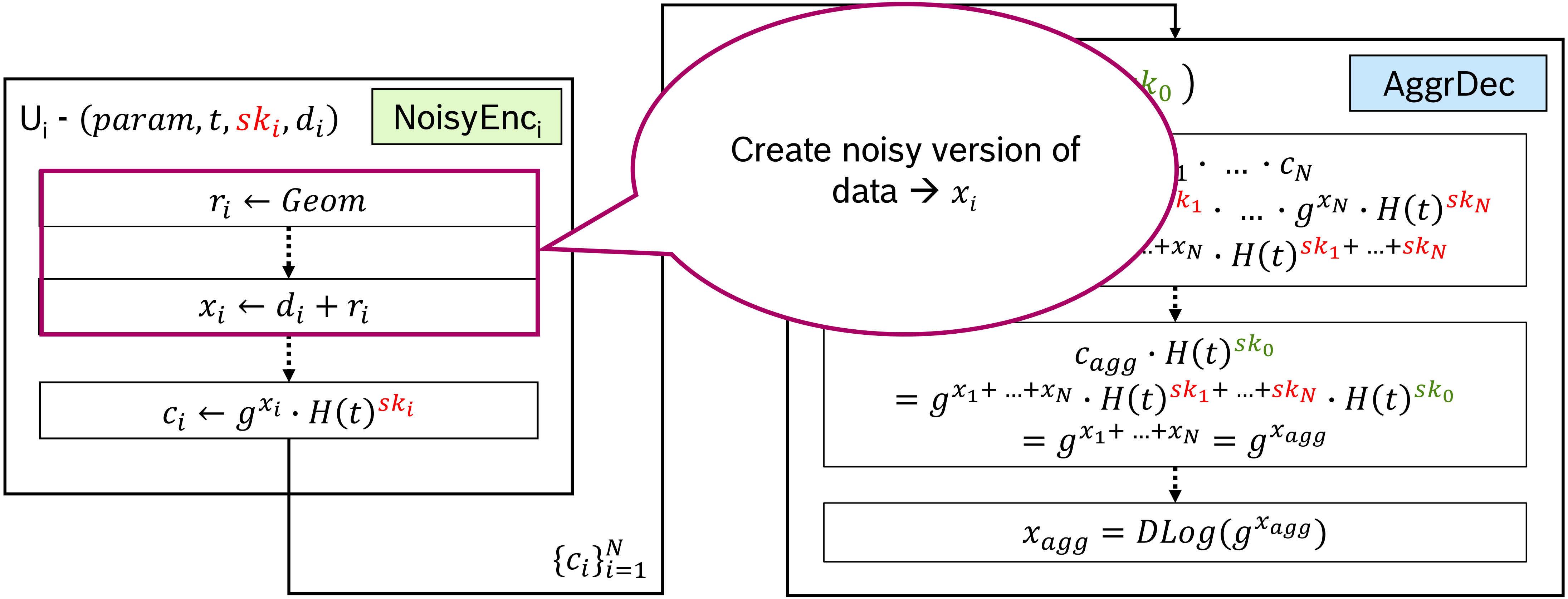
$$c_{agg} := c_1 \cdot \ldots \cdot c_N$$
$$\leftrightarrow c_{agg} = g^{x_1} \cdot H(t)^{sk_1} \cdot \ldots \cdot g^{x_N} \cdot H(t)^{sk_N}$$
$$\leftrightarrow c_{agg} = g^{x_1 + \ldots + x_N} \cdot H(t)^{sk_1 + \ldots + sk_N}$$

$$c_{agg} \cdot H(t)^{sk_0}$$
$$= g^{x_1 + \ldots + x_N} \cdot H(t)^{sk_1 + \ldots + sk_N} \cdot H(t)^{sk_0}$$
$$= g^{x_1 + \ldots + x_N} = g^{x_{agg}}$$

$$x_{agg} = DLog(g^{x_{agg}})$$

[1] E. Shi, T.-H. H. Chan, E. Rieffel, R. Chow, and D. Song. *Privacy-Preserving Aggregation of Time-Series Data*. NDSS 2011.

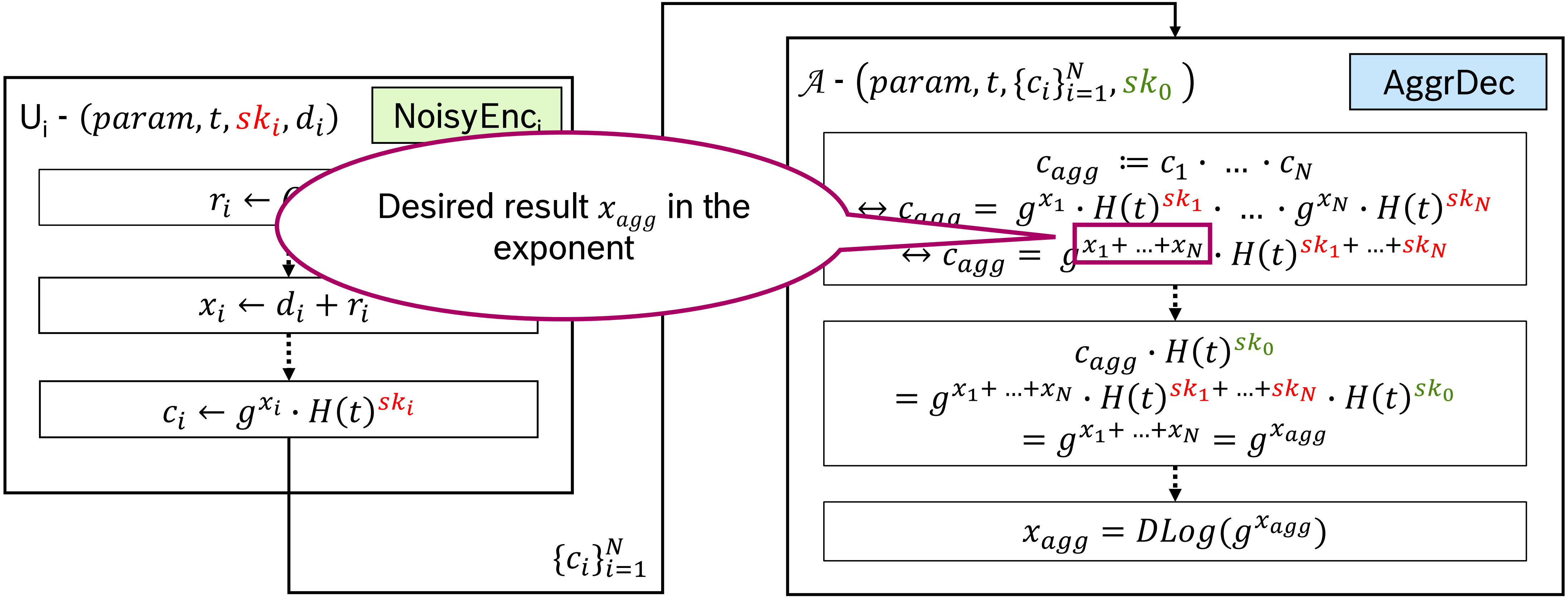
Private Stream Aggregation (PSA)
Shi et al. [1]



[1] E. Shi, T.-H. H. Chan, E. Rieffel, R. Chow, and D. Song. *Privacy-Preserving Aggregation of Time-Series Data*. NDSS 2011.

Private Stream Aggregation (PSA)

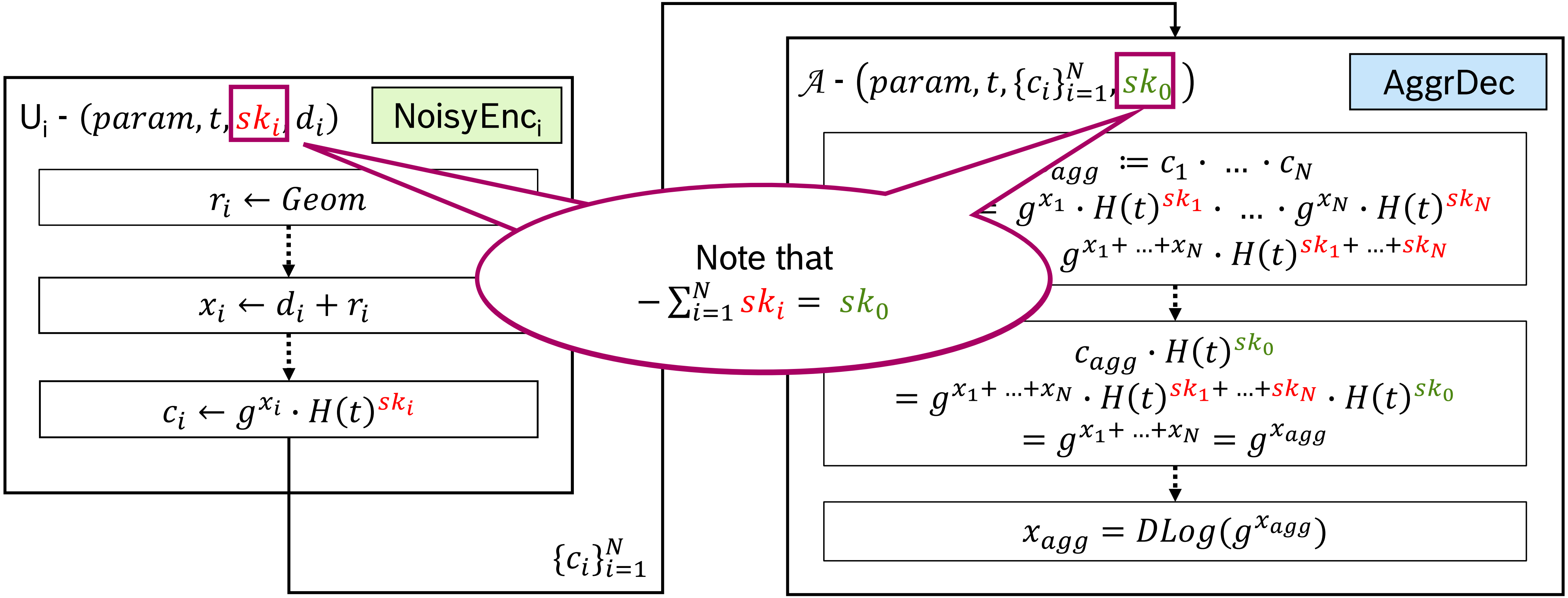
Shi et al. [1]



[1] E. Shi, T.-H. H. Chan, E. Rieffel, R. Chow, and D. Song. *Privacy-Preserving Aggregation of Time-Series Data*. NDSS 2011.

Private Stream Aggregation (PSA)

Shi et al. [1]



[1] E. Shi, T.-H. H. Chan, E. Rieffel, R. Chow, and D. Song. *Privacy-Preserving Aggregation of Time-Series Data*. NDSS 2011.

Private Stream Aggregation (PSA)
Shi et al. [1]

U_i - $(param, t, sk_i, d_i)$ — NoisyEnc$_i$

$$r_i \leftarrow Geom$$

$$x_i \leftarrow d_i + r_i$$

$$c_i \leftarrow g^{x_i} \cdot H(t)^{sk_i}$$

$\{c_i\}_{i=1}^{N}$

$\mathcal{A}$ - $(param, t, \{c_i\}_{i=1}^{N}, sk_0)$ — AggrDec

$$c_{agg} \coloneqq c_1 \cdot \ldots \cdot c_N$$
$$\leftrightarrow c_{agg} = g^{x_1} \cdot H(t)^{sk_1} \cdot \ldots \cdot g^{x_N} \cdot H(t)^{sk_N}$$
$$\leftrightarrow c_{agg} = g^{x_1 + \ldots + x_N} \cdot H(t)^{sk_1 + \ldots + sk_N}$$

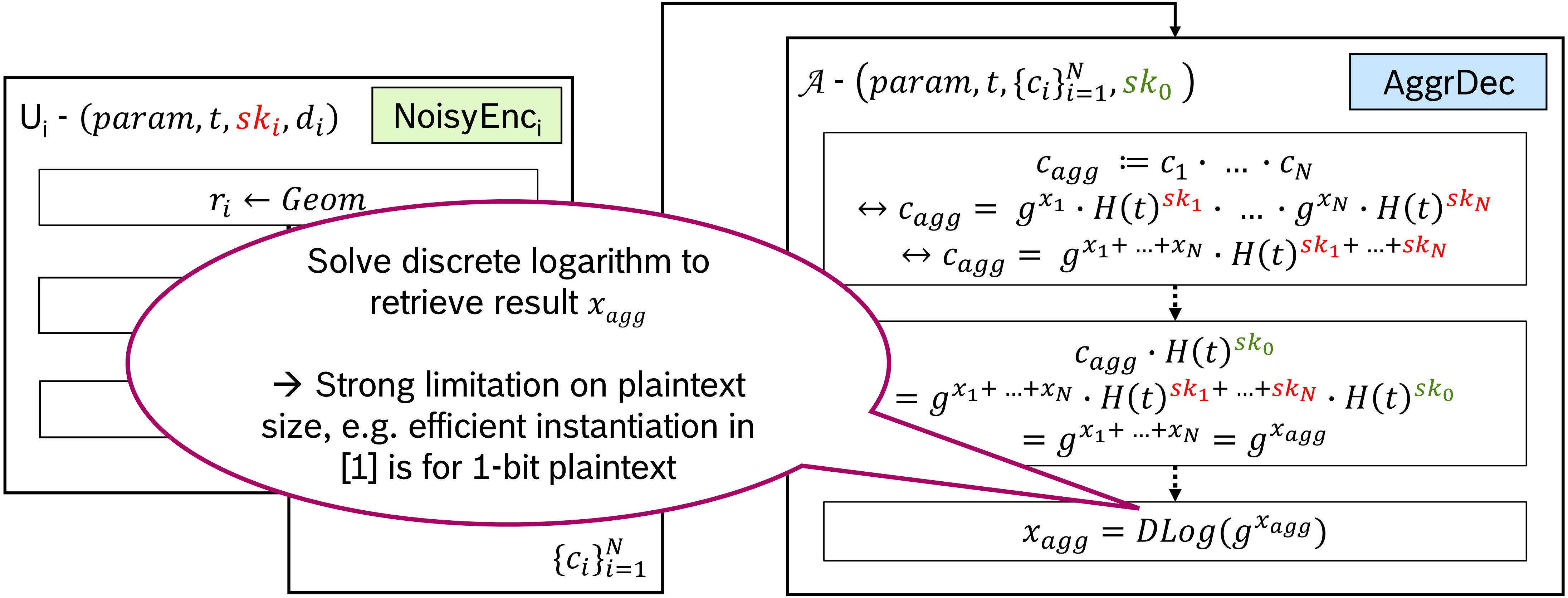
$$c_{agg} \cdot H(t)^{sk_0}$$
$$= g^{x_1 + \ldots + x_N} \cdot H(t)^{sk_1 + \ldots + sk_N} \cdot H(t)^{sk_0}$$
$$= g^{x_1 + \ldots + x_N} = g^{x_{agg}}$$

Cancels out

$$x_{agg} = DLog(g^{x_{agg}})$$

[1] E. Shi, T.-H. H. Chan, E. Rieffel, R. Chow, and D. Song. *Privacy-Preserving Aggregation of Time-Series Data*. NDSS 2011.

Private Stream Aggregation (PSA)
Shi et al. [1]



[1] E. Shi, T.-H. H. Chan, E. Rieffel, R. Chow, and D. Song. *Privacy-Preserving Aggregation of Time-Series Data*. NDSS 2011.

Lattice-based PSA (LaPS)

Imagine, each user can only send value 0 or 1.

→ max. aggregation value ≤ 1000

Goals

- Overcome previous input limitation [1] of 1 bit for 1000 users
- Improve security guarantee

Note: current schemes are breakable by *quantum* computers

Use LWE-based construction

Encrypt values up to 65537 $\approx 2^{16}$

→ max. aggregation value $\lesssim$ 66 million
→ ≈66 thousand times bigger values

Contributions

- Resolve open problem from [1]
- Post-quantum security
- First implementation of PSA scheme: 150 times faster aggregation compared to [1]

[1] E. Shi, T.-H. H. Chan, E. Rieffel, R. Chow, and D. Song. *Privacy-Preserving Aggregation of Time-Series Data*. NDSS 2011.

Learning With Errors (LWE) [2]

$A \in \mathbb{Z}_q^{\lambda \times \kappa}$, $s \in \mathbb{Z}_q^{\kappa}$, $b \in \mathbb{Z}_q^{\lambda}$

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1\kappa} \\ A_{21} & A_{22} & A_{23} & \dots & A_{2\kappa} \\ A_{31} & A_{32} & A_{33} & \dots & A_{3\kappa} \\ A_{41} & A_{42} & A_{43} & \dots & A_{4\kappa} \\ \vdots & & & & \\ A_{\lambda 1} & A_{\lambda 2} & A_{\lambda 3} & \dots & A_{\lambda \kappa} \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ \dots \\ s_\kappa \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \dots \\ b_\lambda \end{pmatrix} \pmod{q}$$

Given (A,b): find s

[2] O. Regev. *On lattices, learning with errors, random linear codes, and cryptography*. STOC 2005.

BOSCH

Learning With Errors (LWE) [2]

discrete Gaussian distribution

coefficients $A \in \mathbb{Z}_q^{\lambda \times \kappa}$ | secret $s \in \mathbb{Z}_q^{\kappa}$ | error $e_i \leftarrow \Psi$ | result $b \in \mathbb{Z}_q^{\lambda}$

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1\kappa} \\ A_{21} & A_{22} & A_{23} & \dots & A_{2\kappa} \\ A_{31} & A_{32} & A_{33} & \dots & A_{3\kappa} \\ A_{41} & A_{42} & A_{43} & \dots & A_{4\kappa} \\ \vdots & & & & \\ A_{\lambda 1} & A_{\lambda 2} & A_{\lambda 3} & \dots & A_{\lambda \kappa} \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ \dots \\ s_\kappa \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ \dots \\ e_\lambda \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \dots \\ b_\lambda \end{pmatrix} \pmod{q}$$

Given (A,b): find s

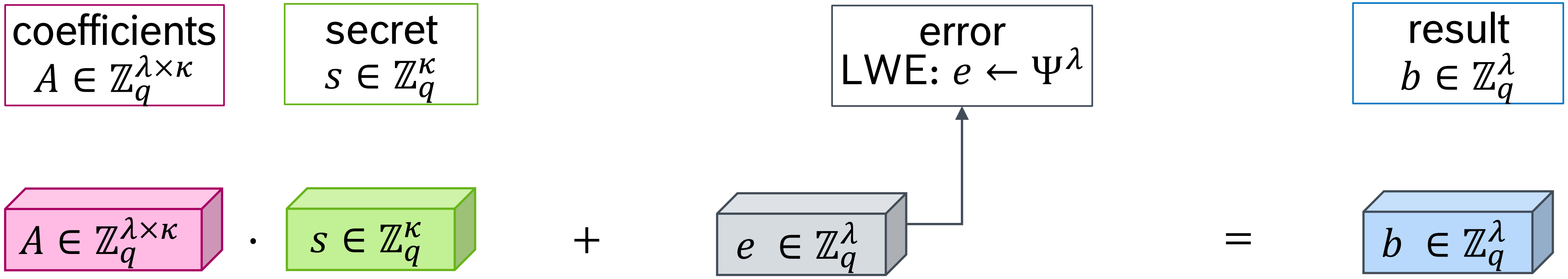
➡ as hard as **worst-case** lattice problems*

➡ **post-quantum** secure

*) relevant parameters: (q, κ, Ψ)

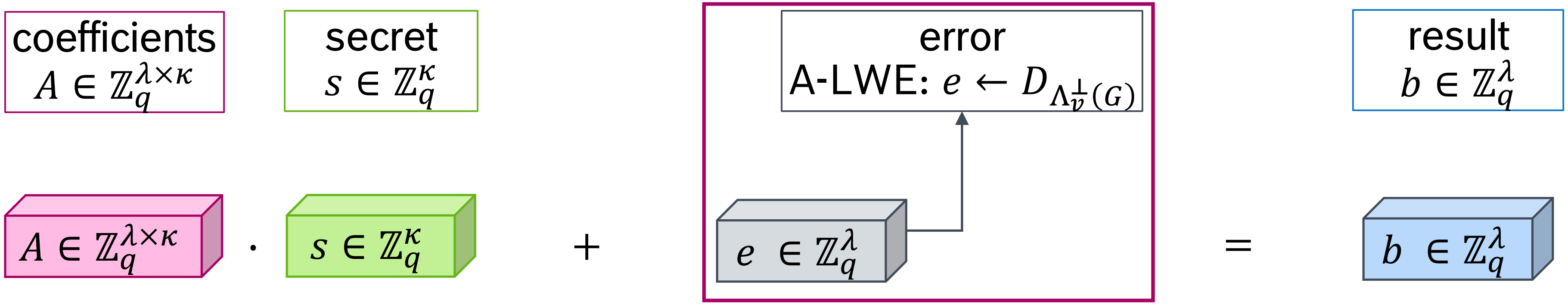
[2] O. Regev. *On lattices, learning with errors, random linear codes, and cryptography*. STOC 2005.

LWE [2]



[2] O. Regev. *On lattices, learning with errors, random linear codes, and cryptography*. STOC 2005.

Augmented LWE (A-LWE) [3]



[3] R. El Bansarkhani, Ö. Dagdelen, J. Buchmann. *Augmented Learning with Errors: The Untapped Potential of the Error Term*. FC 2015.

Augmented LWE (A-LWE) – Message Embedding [3]

Straightforward encryption

coefficients
$A \in \mathbb{Z}_q^{\lambda \times \kappa}$

secret
$s \in \mathbb{Z}_q^{\kappa}$

error
A-LWE: $e \leftarrow D_{\Lambda_v^\perp(G)}$

result
$b \in \mathbb{Z}_q^{\lambda}$

$A \in \mathbb{Z}_q^{\lambda \times \kappa} \cdot s \in \mathbb{Z}_q^{\kappa} + e \in \mathbb{Z}_q^{\lambda} = b \in \mathbb{Z}_q^{\lambda}$

Compute $v = f(m)$

Draw $e \leftarrow D_{\Lambda_v^\perp(G)}$

A-LWE: $e \leftarrow D_{\Lambda_v^\perp(G)}$

- encode message m using "some" function f
- $D_{\Lambda_v^\perp(G)}$ defined by v and G

→ m is embedded in e

A-LWE: b looks like LWE-term but contains message m

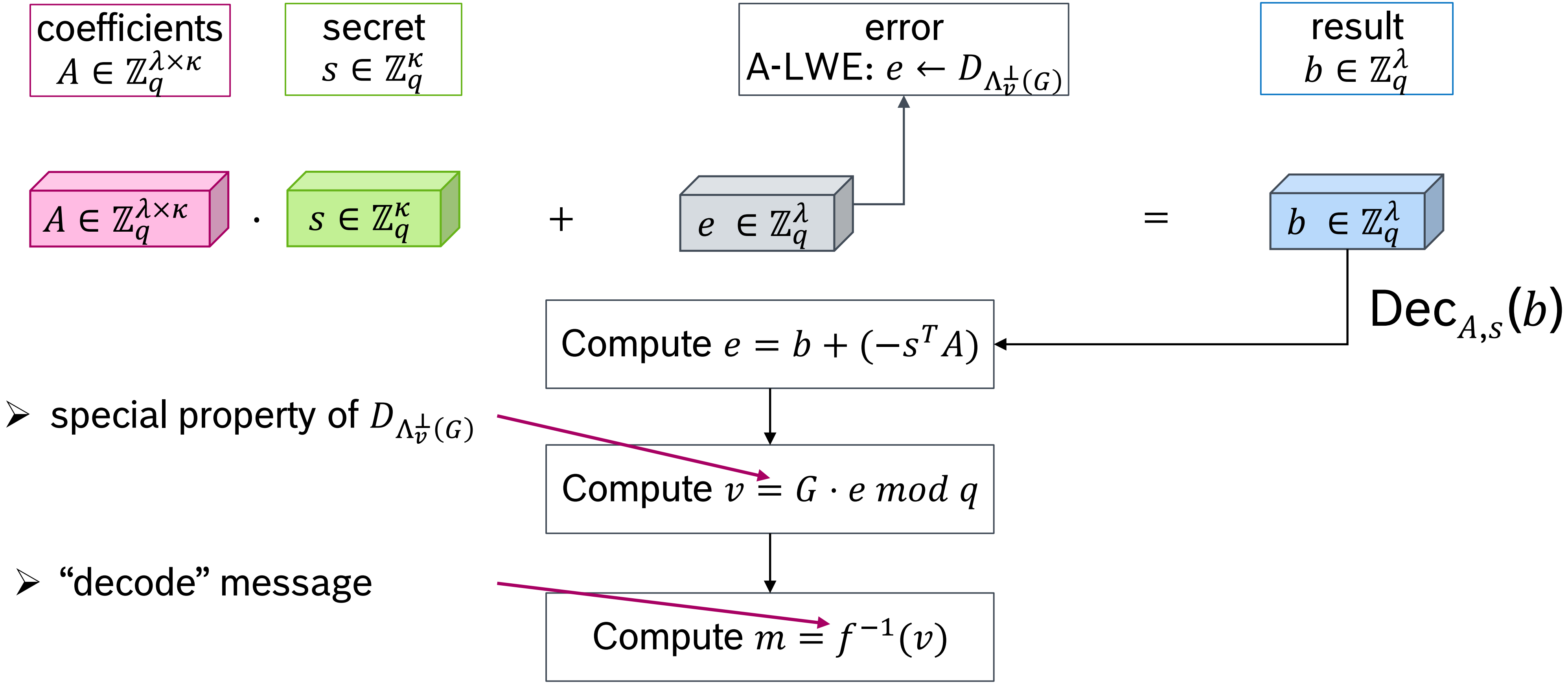
→ b = $\mathrm{Enc}(m)$

[3] R. El Bansarkhani, Ö. Dagdelen, J. Buchmann. *Augmented Learning with Errors: The Untapped Potential of the Error Term*. FC 2015.

BOSCH

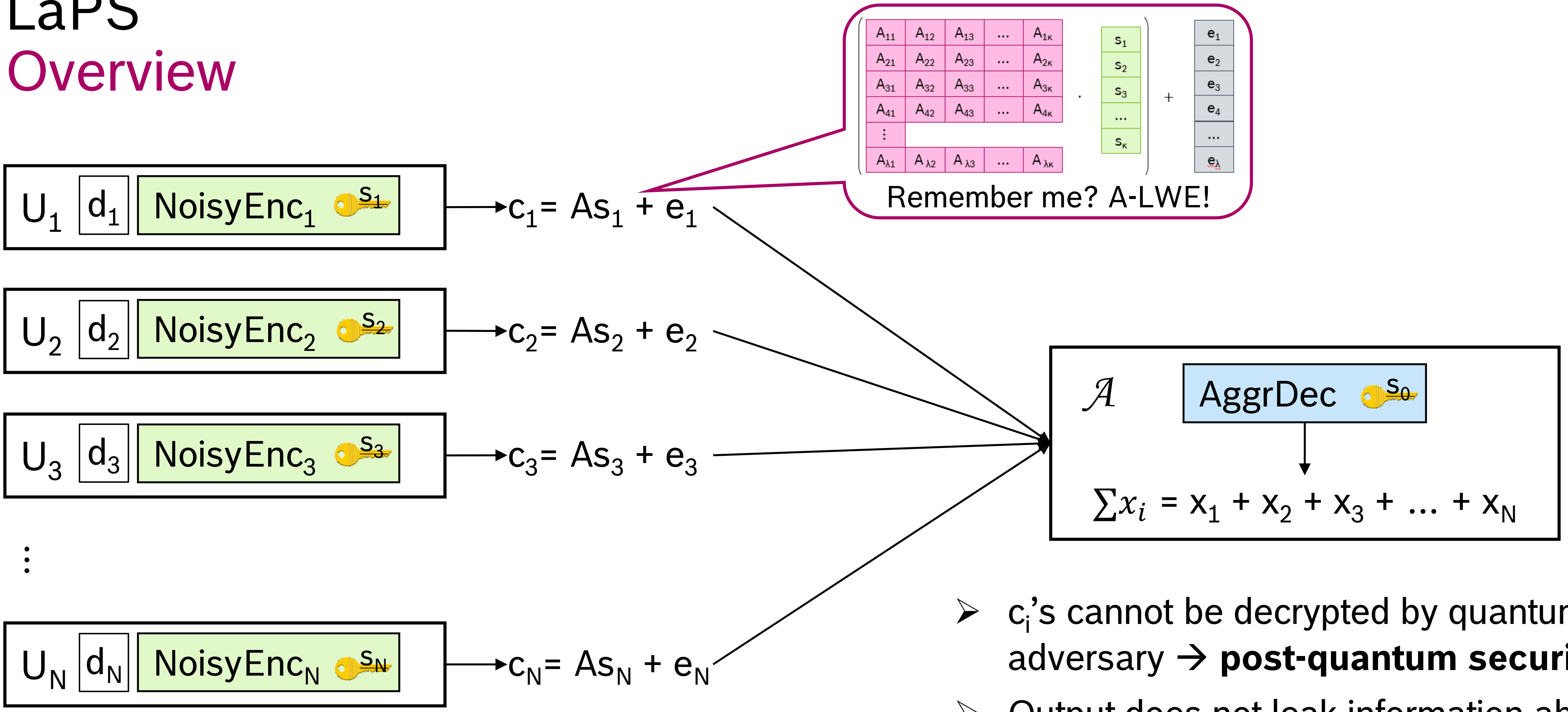
Augmented LWE (A-LWE) – Message Embedding [3]

Decryption



[3] R. El Bansarkhani, Ö. Dagdelen, J. Buchmann. *Augmented Learning with Errors: The Untapped Potential of the Error Term*. FC 2015.
BOSCH

LaPS
Overview



- c_i's cannot be decrypted by quantum adversary → **post-quantum security**
- Output does not leak information about individual d_i's nor x_i's → **privacy**

LaPS
Let's take a closer look

U_i - $(param, A, G, pk, s_i, d_i)$ **NoisyEnc$_i$**

$$x_i \leftarrow d_i + \eta_i$$

$$v_i = AHOM.Enc(pk, x_i)$$

Sample $e_i \leftarrow D_{\Lambda^{\perp}_{v_i}(G)}$

$$c_i \leftarrow A s_i + e_i$$

$\{c_i\}_{i=1}^{N}$

$\mathcal{A}$ - $(param, A, G, \{c_i\}_{i=1}^{N}, sk, s_0)$ **AggrDec**

$$c_{agg} := c_1 + \ldots + c_N$$
$$\leftrightarrow c_{agg} = A \cdot \sum_{i=1}^{N} s_i + \sum e_i$$

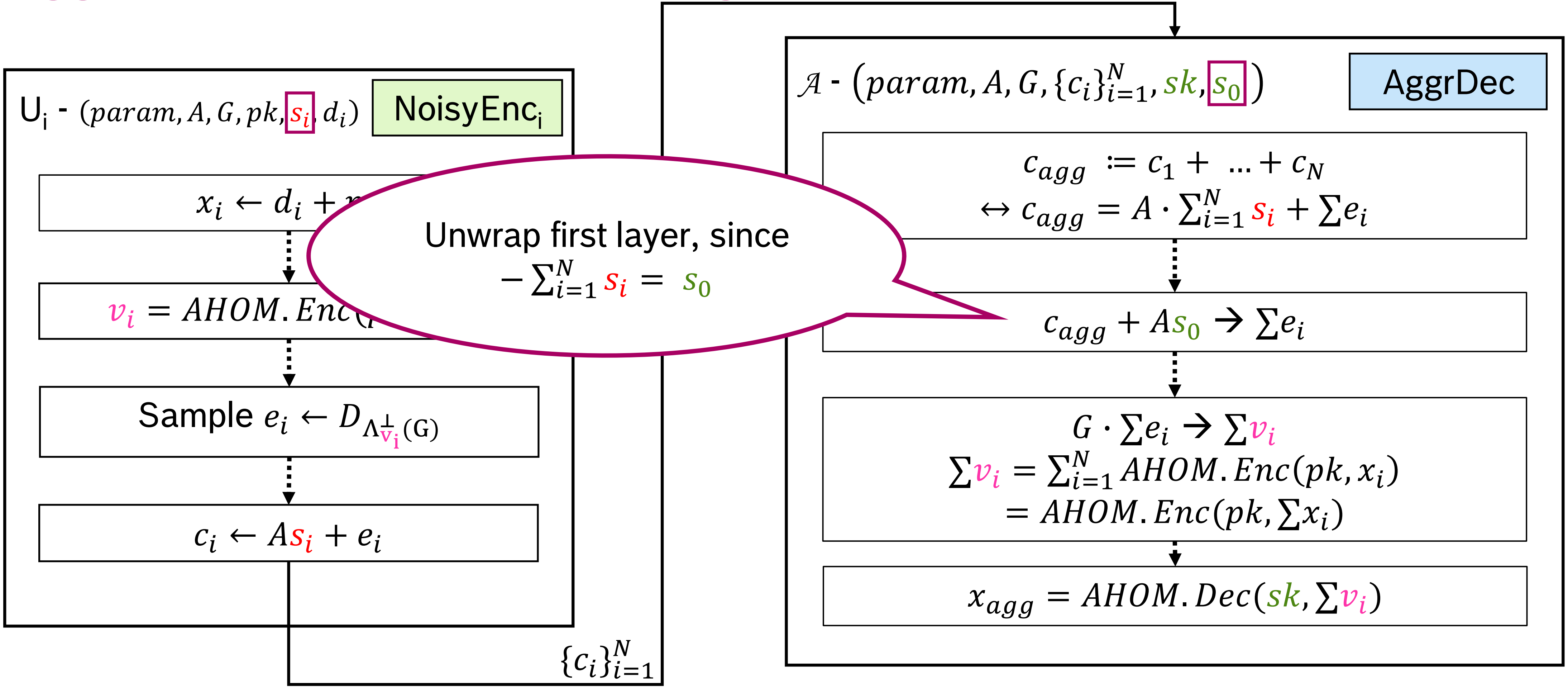
$$c_{agg} + A s_0 \rightarrow \sum e_i$$

$$G \cdot \sum e_i \rightarrow \sum v_i$$
$$\sum v_i = \sum_{i=1}^{N} AHOM.Enc(pk, x_i)$$
$$= AHOM.Enc(pk, \sum x_i)$$

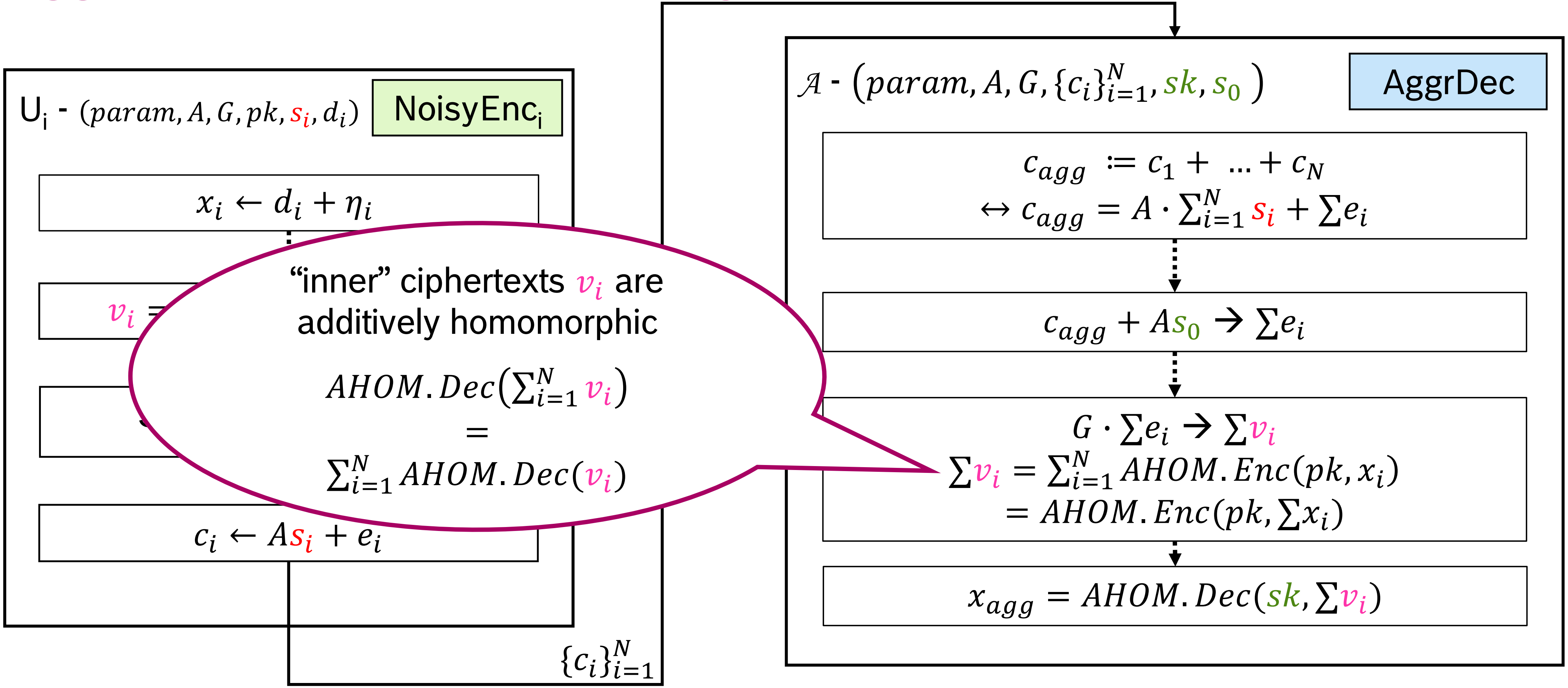
$$x_{agg} = AHOM.Dec(sk, \sum v_i)$$

Research and Technology Center | Daniela Becker CR/RTC3.2-NA | 02/19/2018

LaPS: Correctness
AggrDec – Double Unwrapping



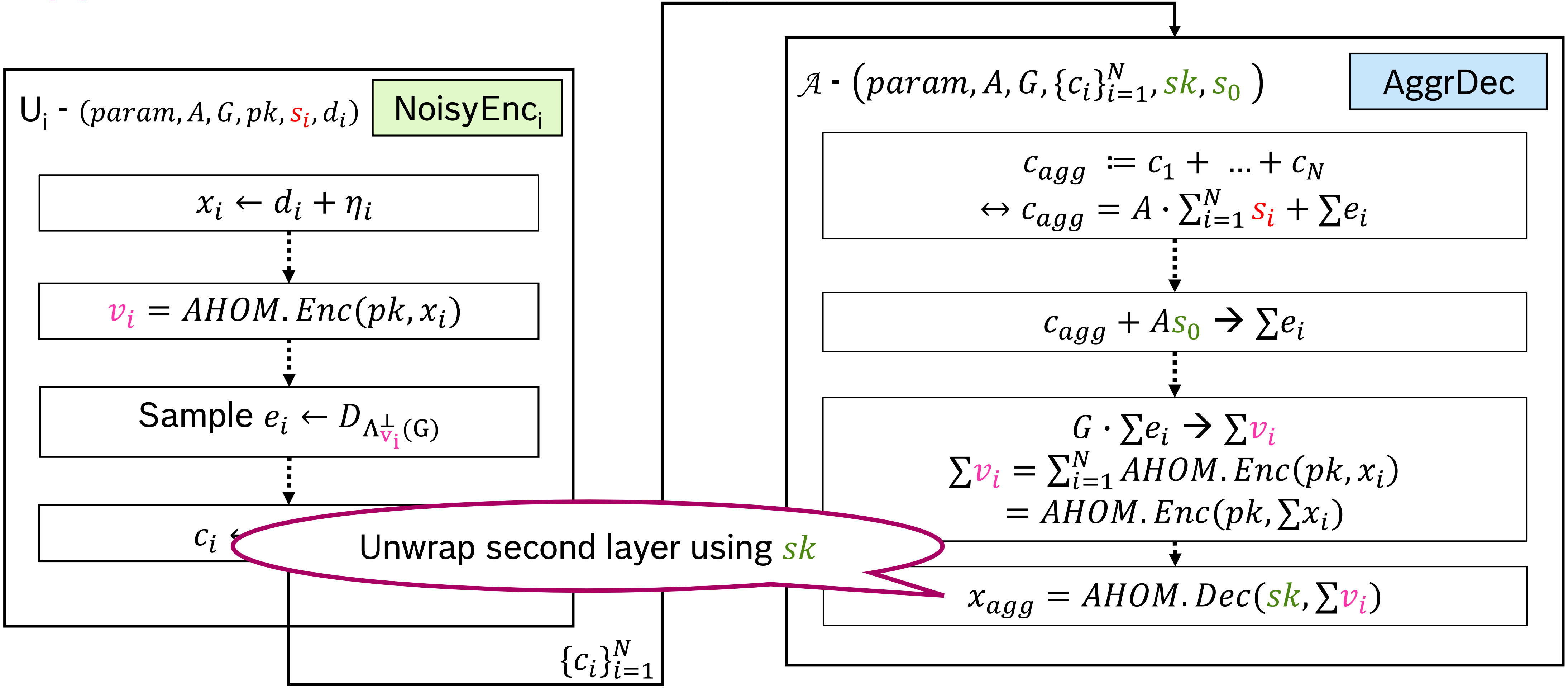
LaPS: Correctness
AggrDec – Double Unwrapping



BOSCH

LaPS: Correctness

AggrDec – Double Unwrapping



LaPS: Correctness
AggrDec – Double Unwrapping

Correctness:

As long as

➢ $AHOM.Dec\left(\sum_{i=1}^{N} v_i\right) = \sum_{i=1}^{N} x_i$

and since

➢ $G \cdot e_i \bmod q = v_i$, where $e_i \leftarrow D_{\Lambda_{v_i}}^{\perp}(G)$,

the aggregator indeed outputs the noisy sum aggregate of the users' values. We require $AHOM$ to be additively homomorphic - therefore the sum of the homomorphic ciphertexts $\sum_{i=1}^{N} v_i$ corresponds to an encryption of the sum of the underlying plaintexts $\sum_{i=1}^{N} x_i$.

$\mathcal{A}$ - $\left(param, A, G, \{c_i\}_{i=1}^{N}, sk, s_0\right)$ **AggrDec**

$$c_{agg} := c_1 + \ldots + c_N$$
$$\leftrightarrow c_{agg} = A \cdot \sum_{i=1}^{N} s_i + \sum e_i$$

$$c_{agg} + A s_0 \rightarrow \sum e_i$$

$$G \cdot \sum e_i \rightarrow \sum v_i$$
$$\sum v_i = \sum_{i=1}^{N} AHOM.Enc(pk, x_i)$$
$$= AHOM.Enc\left(pk, \sum x_i\right)$$

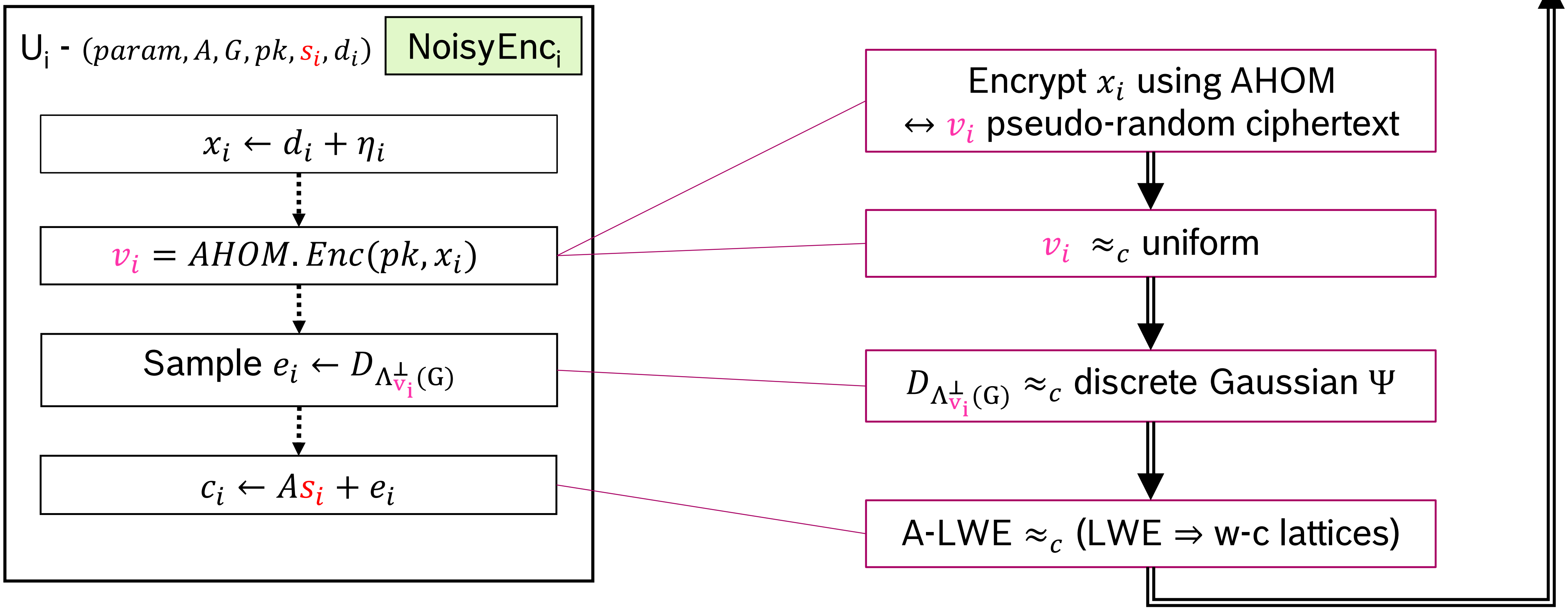
$$x_{agg} = AHOM.Dec\left(sk, \sum v_i\right)$$

LaPS: Security Guarantees

NoisyEnc$_i$ – Let's take a closer look

For security we want:

Break NoisyEnc$_i$ $\leftrightarrow$ break w-c lattice problem



BOSCH

LaPS: Security Guarantees
NoisyEnc$_i$ – Let's take a closer look

For security we want:

Break NoisyEnc$_i$ ↔ break w-c lattice problem

U$_i$ - $(param, A, G, pk, s_i, d_i)$ **NoisyEnc$_i$**

$x_i \leftarrow d_i + \eta_i$

↓

$v_i = AHOM.Enc(pk, x_i)$

↓

Sample $e_i \leftarrow D_{\Lambda^{\perp}_{v_i}(G)}$

↓

$c_i \leftarrow As_i + e_i$

<u>Theorem 1 (Semantic Security):</u>

Let the output of $AHOM.Enc$ be indistinguishable from random [...]. Then, the ciphertexts generated by NoisyEnc in are *semantically secure* assuming the hardness of worst case lattice problems.

<u>Theorem 2 (Aggregator Obliviousness Security):</u>

Let the output of $AHOM.Enc$ be indistinguishable from random [...]. LaPS satisfies *aggregator oblivious security* assuming the hardness of worst case lattice problems.

LaPS Instantiation
Experimental Results

Instantiation using $\mathcal{M}_\chi$ → *discrete Laplace mechanism* and $\mathcal{AHOM}$ → *reduced* BGV* encryption scheme

	BEFORE [1]		AFTER (this work)**	
NoisyEnc$_i$	$p \in \{0,1\}$:	0.6 ms	$p \leq 5$: $p \leq 37$: $p \leq 65537$:	3.58 ms 3.62 ms 3.73 ms
AggrDec	$p \in \{0,1\}$:	300 ms	$p \leq 5$: $p \leq 37$: $p \leq 65537$:	1.87 ms 1.88 ms 1.96 ms

*) Original BGV Scheme [4], adapted from [5] and reduced to homomorphic addition (, i.e. no multiplication)

**) Runtime results [ms] for LaPS instance for 1000 users, 80-bit security

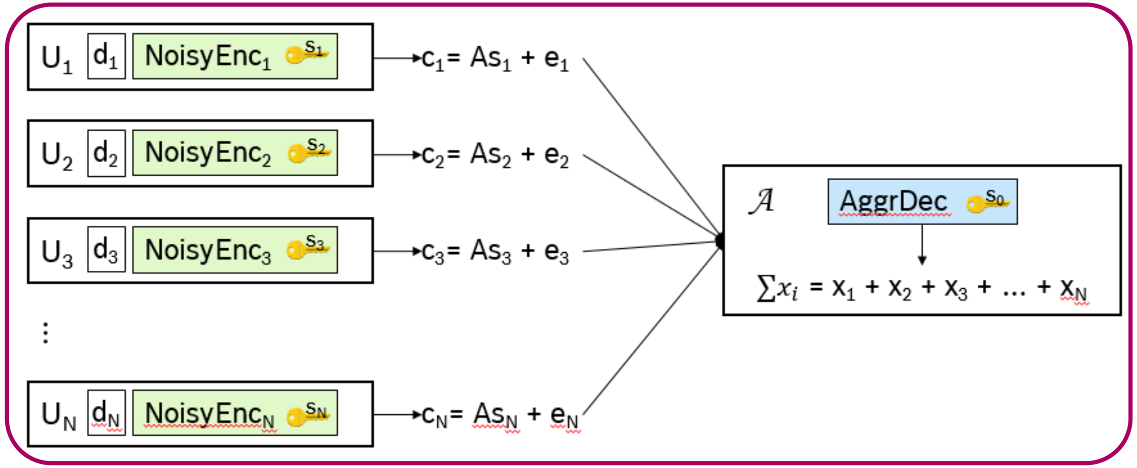
MacBook, macOS Sierra, single 2.5 GHz Intel Core i7 and 16GB memory; averaged over 1000 runs

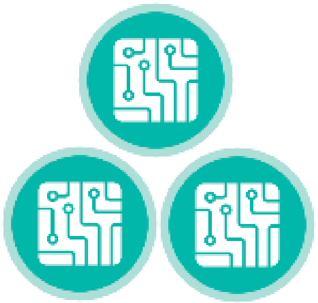
[4] Z. Brakerski and V. Vaikuntanathan, *Efficient Fully Homomorphic Encryption from (Standard) LWE*. ECCC 2011.

[5] I. Damgard, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. *Practical Covertly Secure MPC for Dishonest Majority or: Breaking the SPDZ Limits*. ESORICS 2013.

Summary

- Lattice-based Private Stream Aggregation
  - Plug-and-play deployment of additively homomorphic encryption
- Strong security & privacy guarantees
  - (Augmented) LWE-assumption provides post-quantum security
  - Formal Differential Privacy analysis
- Significant efficiency improvements compared to previous work
  - 150 times faster decryption
  - ≈66000 times larger plaintext space



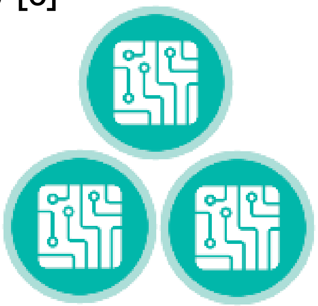


Summary

- Lattice-based Private Stream Aggregation
- Strong security & privacy guarantees
- Significant efficiency improvements compared to previous work

Outlook

- Dynamic joins and leaves / user failures
- Enhance scheme to aggregator unforgeability / public verifiability of aggregate result
  - E.g. by combining with Homomorphic Aggregate Signature scheme (HAS) [6]



[6] Z. Jing. *A homomorphic aggregate signature scheme based on lattice*. MPE 2014.

THANK YOU

QUESTIONS?

DANIELA.BECKER2@US.BOSCH.COM