

RUHR-UNIVERSITÄT BOCHUM

ABE Cubed: Advanced Benchmarking Extensions for ABE Squared

CHES 2025 Conference

Sven Argo, Marloes Venema, Doreen Riepel, Tim Güneysu, Diego F. Aranha

17.09.2025

Chair for Security Engineering
Faculty of Computer Science
Ruhr University Bochum

Motivation

ABE Cubed

- **ABE**: powerful role-based access control on cryptographic level
- **ABE Squared**[1]: systematic benchmarks & optimizations
  - Advanced features not considered
  - No structural optimizations

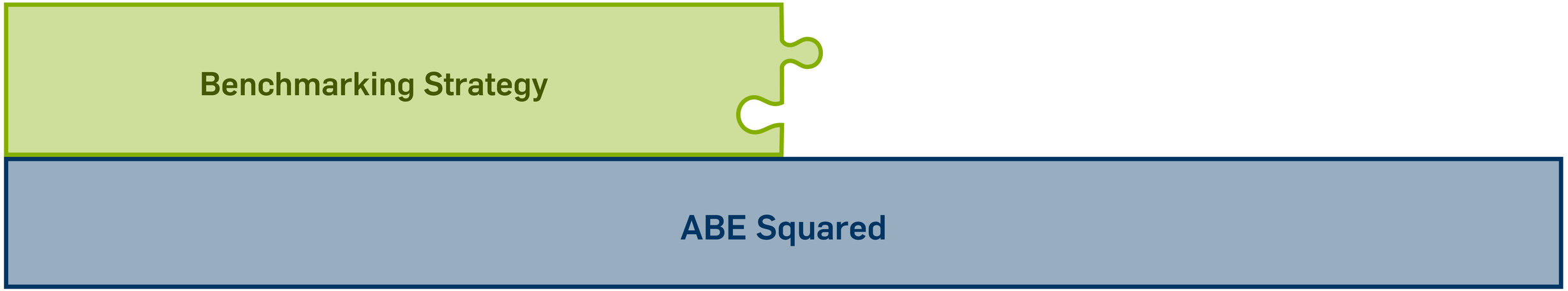
ABE Squared

[1] "ABE Squared: Accurately Benchmarking Efficiency of Attribute-Based Encryption", de la Piedra et al., TCHES 2022

Motivation

ABE Cubed

- **ABE**: powerful role-based access control on cryptographic level
- **ABE Squared**[1]: systematic benchmarks & optimizations
  - Advanced features not considered
  - No structural optimizations

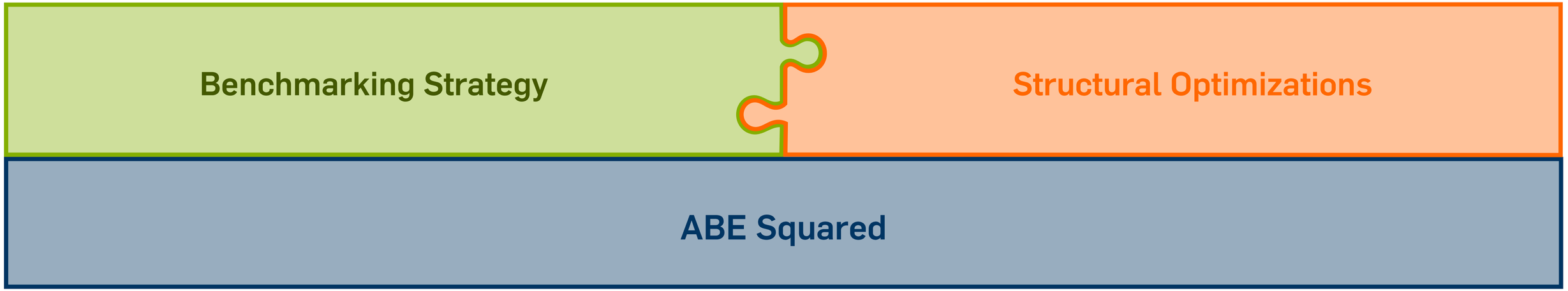


[1] "ABE Squared: Accurately Benchmarking Efficiency of Attribute-Based Encryption", de la Piedra et al., TCHES 2022

Motivation

ABE Cubed

- **ABE**: powerful role-based access control on cryptographic level
- **ABE Squared**[1]: systematic benchmarks & optimizations
  - Advanced features not considered
  - No structural optimizations



[1] "ABE Squared: Accurately Benchmarking Efficiency of Attribute-Based Encryption", de la Piedra et al., TCHES 2022

Agenda

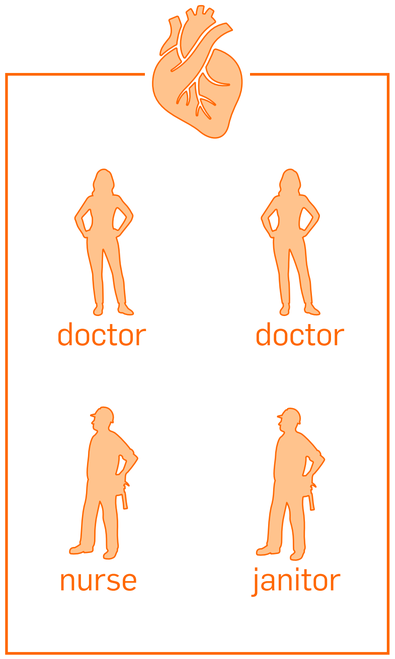
- ABE: Scenario, Building Blocks, Our Scheme
- Recap: ABE Squared
- Our Structural Optimizations
- Our Benchmarking Strategy
- Evaluation & Results
- Conclusion

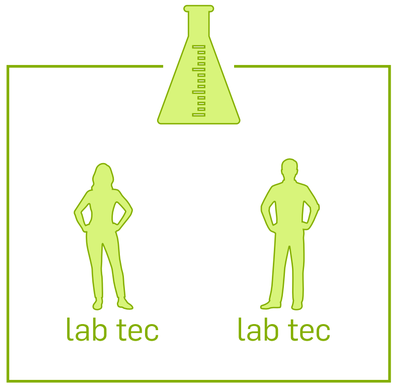
ABE: Scenario, Building Blocks, Our Scheme

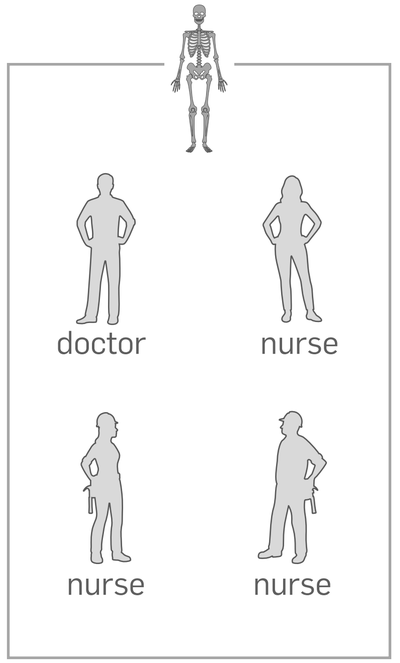
ABE Scenario

Multi-Authority Ciphertext-Policy Attribute-based Encryption



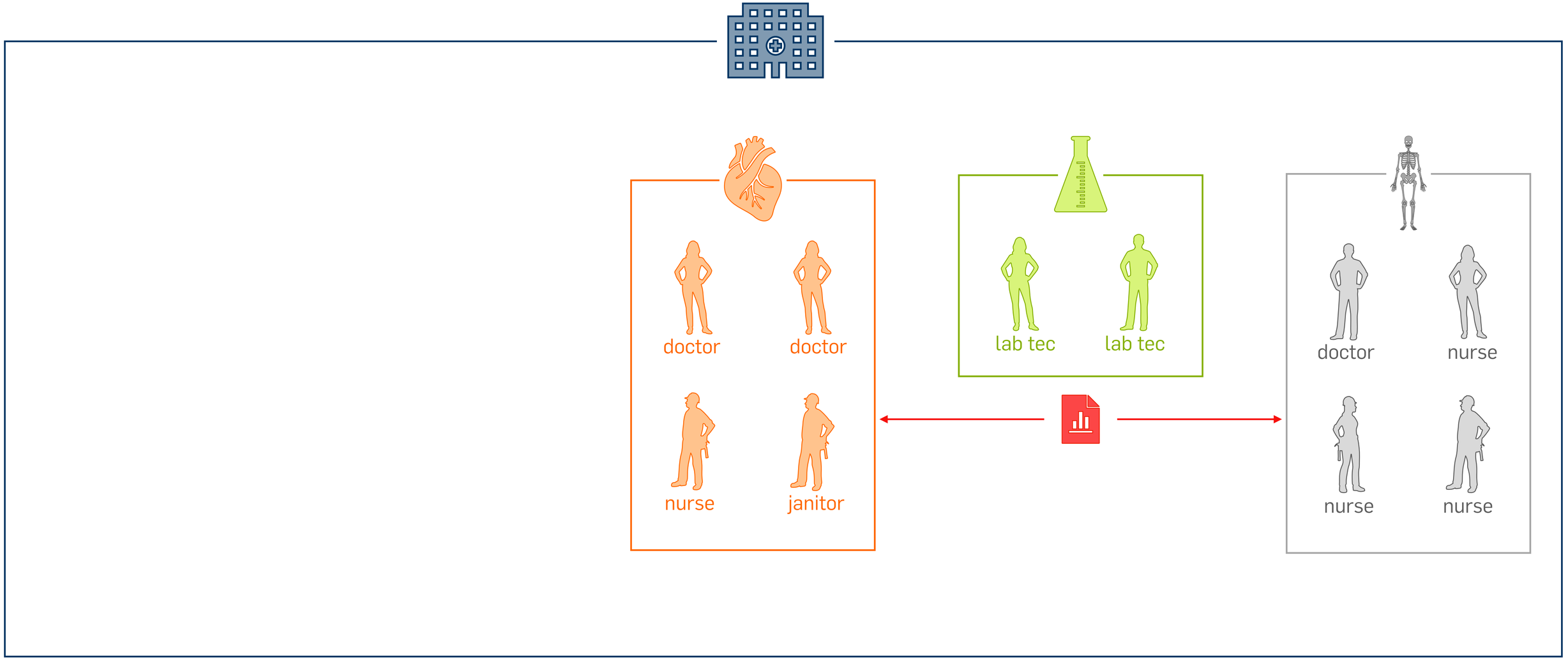






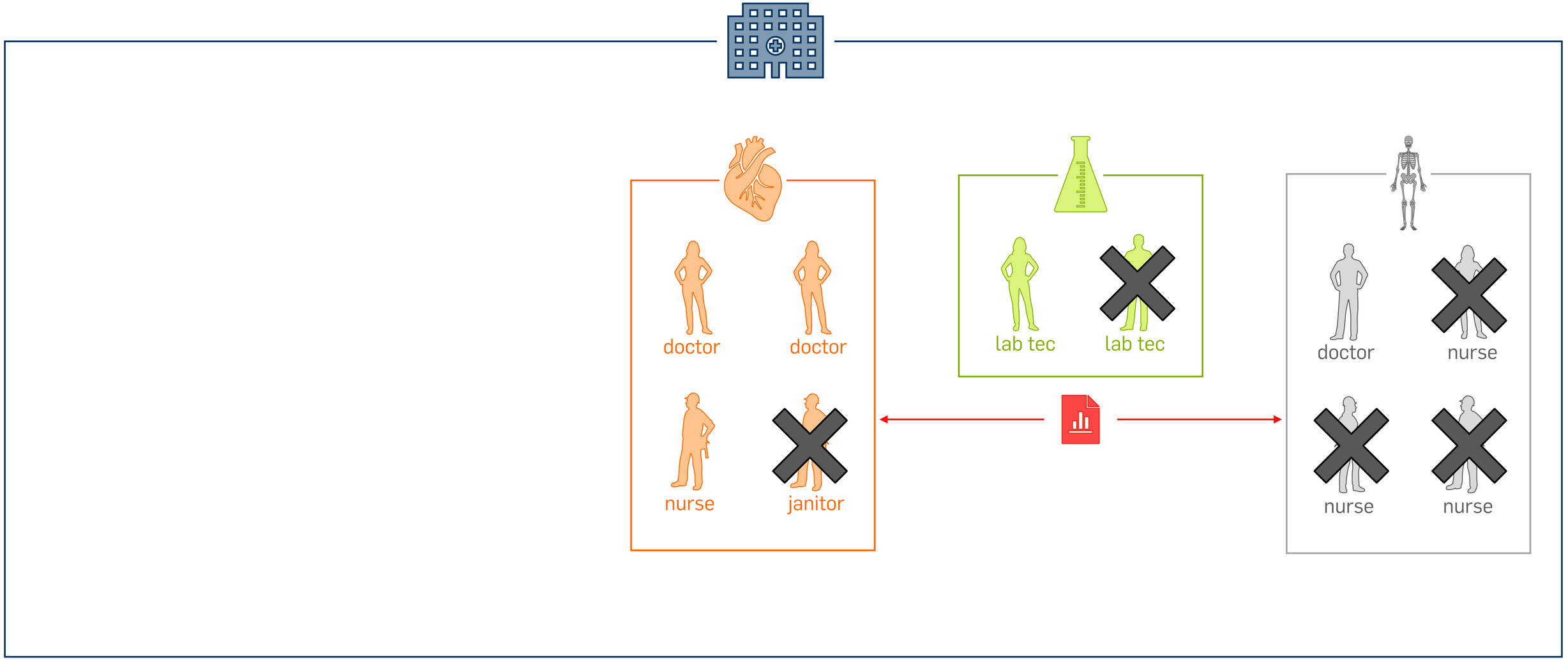
ABE Scenario

Multi-Authority Ciphertext-Policy Attribute-based Encryption



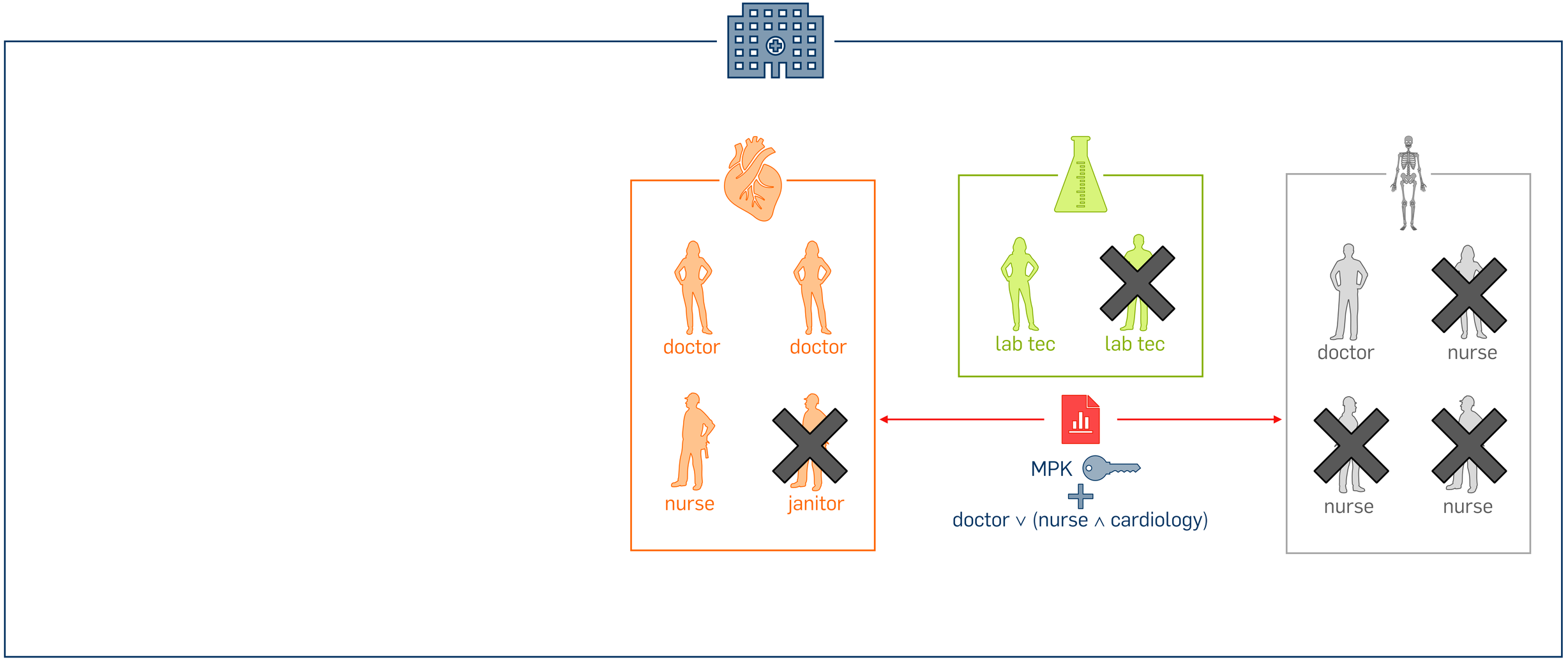
ABE Scenario

Multi-Authority Ciphertext-Policy Attribute-based Encryption



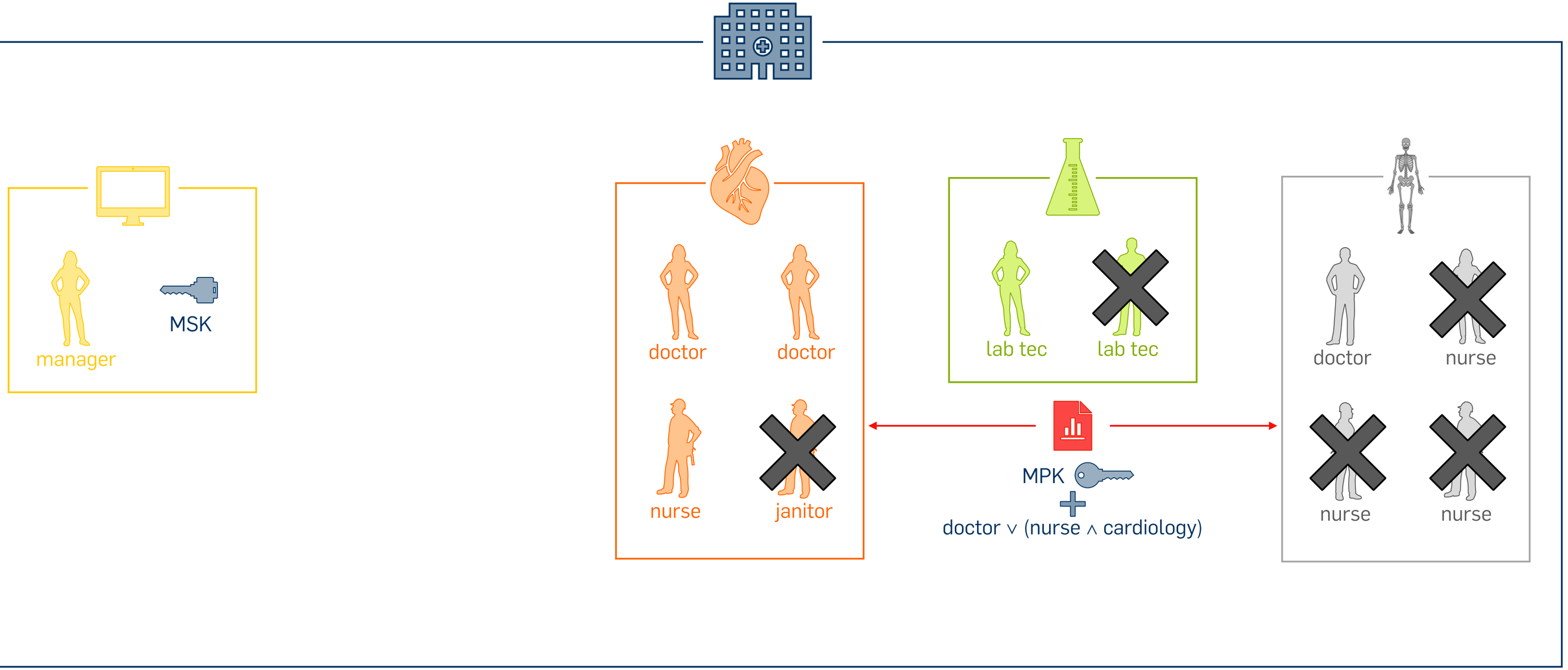
ABE Scenario

Multi-Authority Ciphertext-Policy Attribute-based Encryption



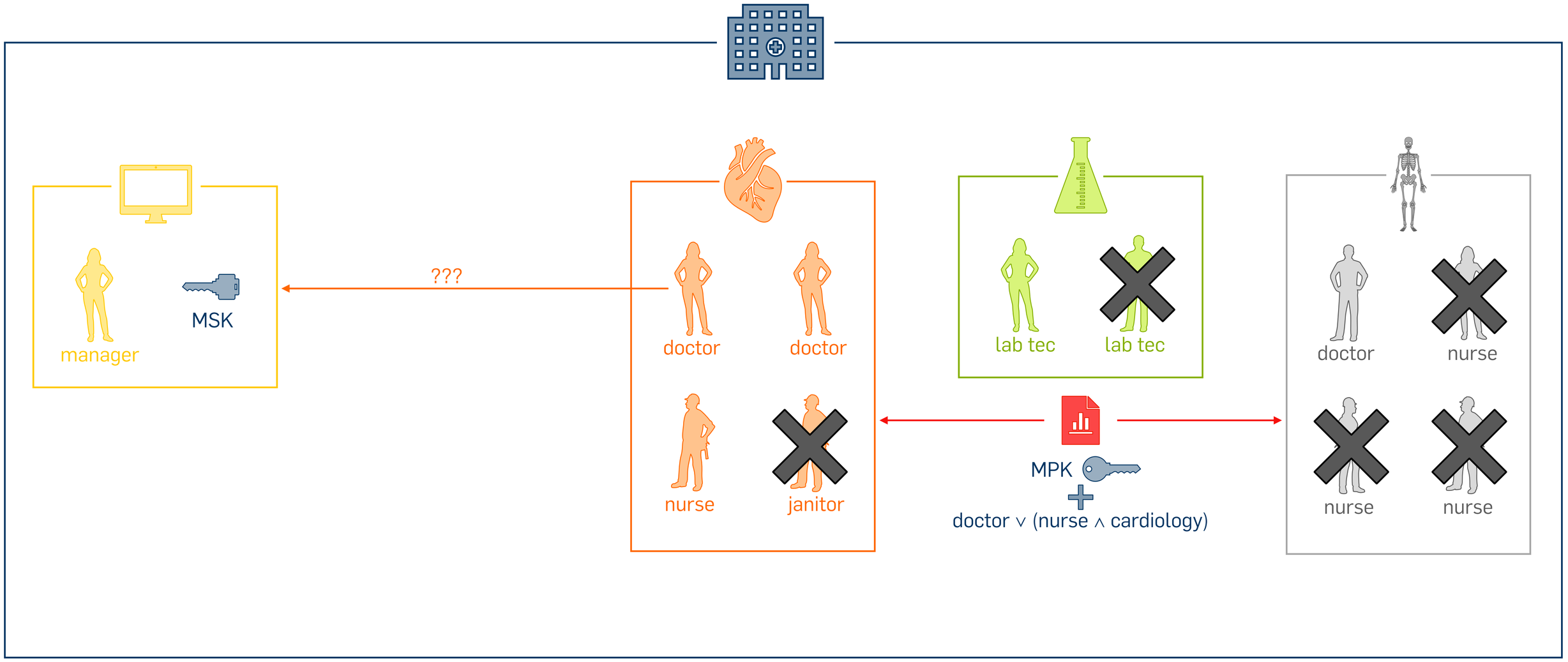
ABE Scenario

Multi-Authority Ciphertext-Policy Attribute-based Encryption



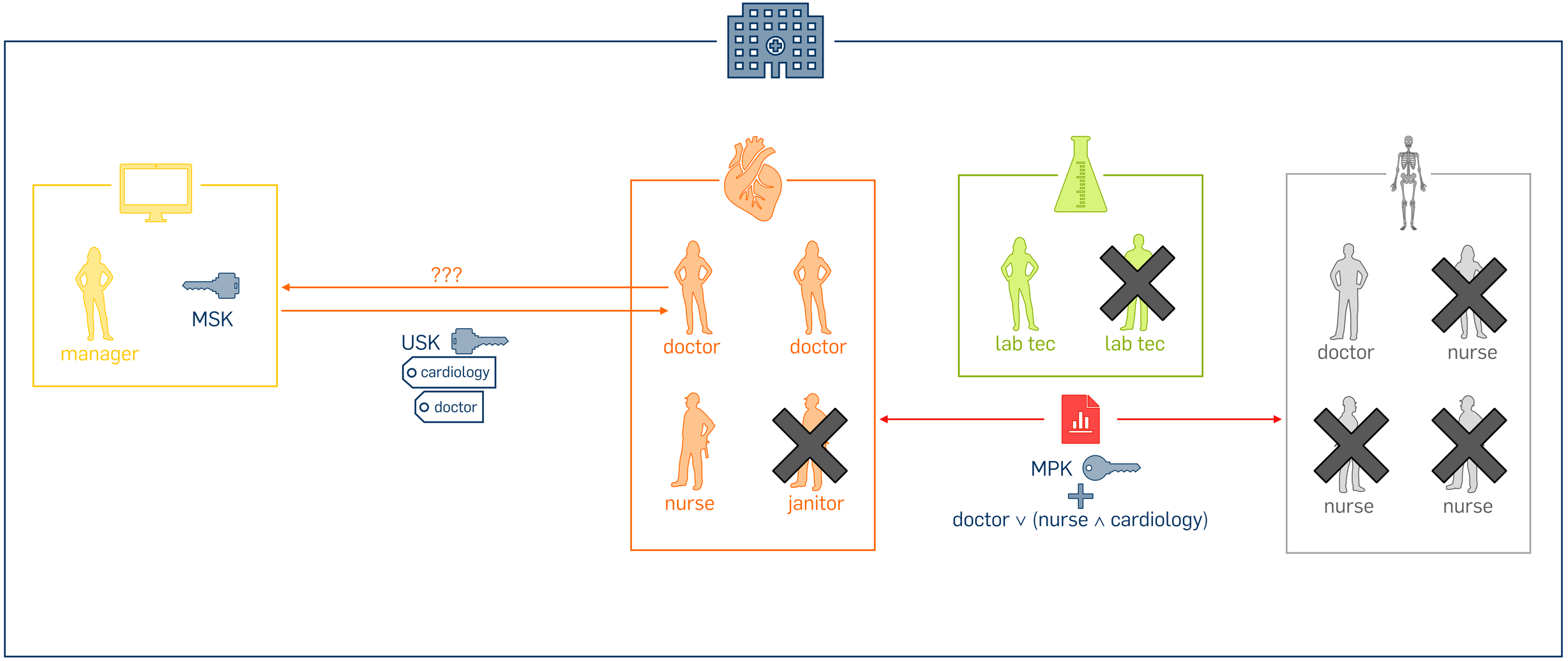
ABE Scenario

Multi-Authority Ciphertext-Policy Attribute-based Encryption



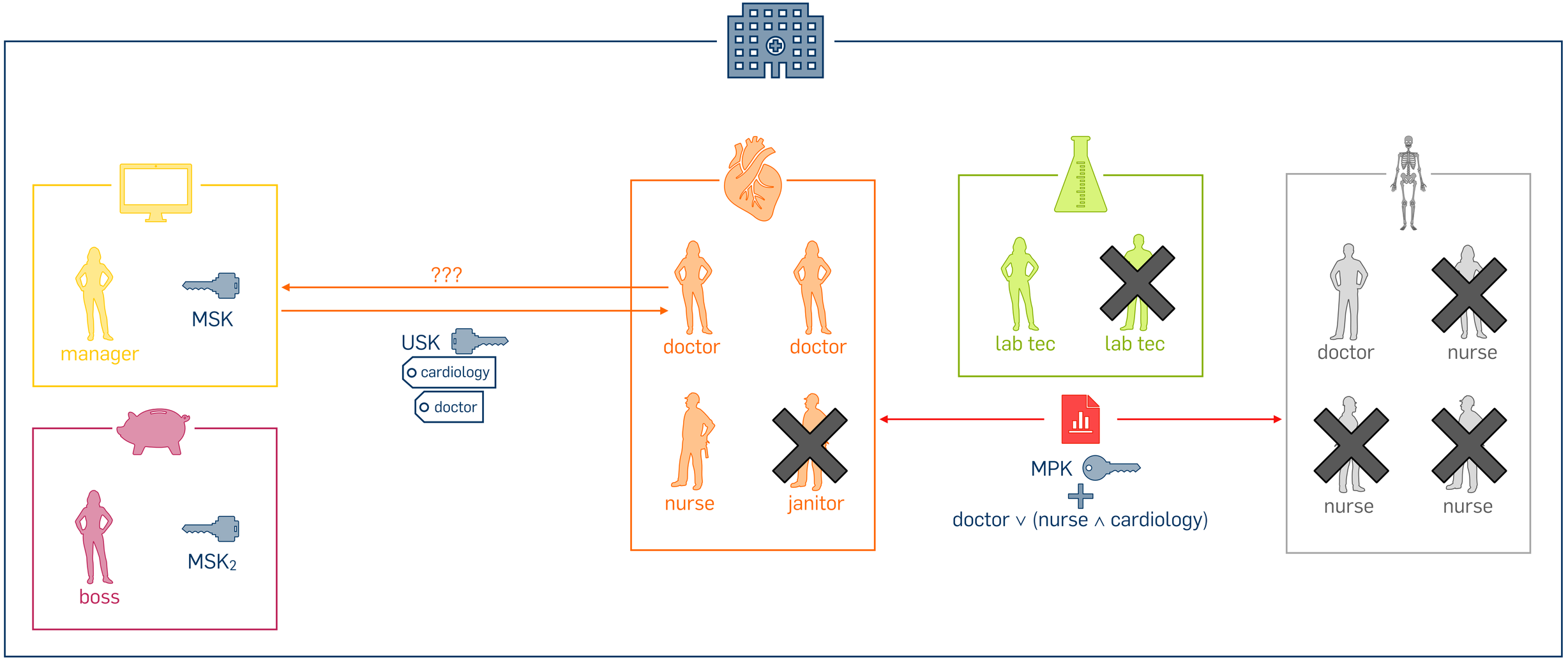
ABE Scenario

Multi-Authority Ciphertext-Policy Attribute-based Encryption



ABE Scenario

Multi-Authority Ciphertext-Policy Attribute-based Encryption



Building Blocks

“Linear Algebra in Exponents”

Building Blocks

“Linear Algebra in Exponents”

cyclic groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$

prime order p, $\cong \mathbb{Z}_p$

Building Blocks

“Linear Algebra in Exponents”

cyclic groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$

prime order p, $\cong \mathbb{Z}_p$

+

Pairing $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$

$$e(g^a, h^b) = e(g, h)^{a \cdot b}$$

Building Blocks

“Linear Algebra in Exponents”

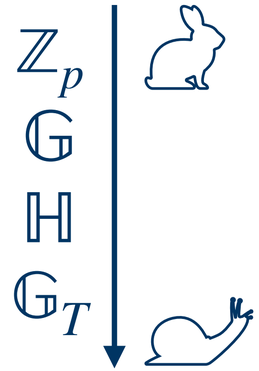
cyclic groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$

prime order p, $\cong \mathbb{Z}_p$

+

Pairing $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$

$$e(g^a, h^b) = e(g, h)^{a \cdot b}$$



Building Blocks

“Linear Algebra in Exponents”

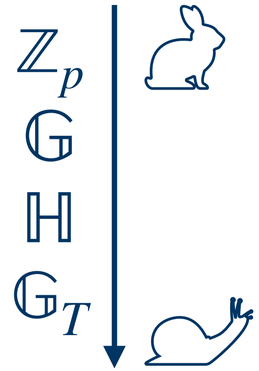
cyclic groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$

prime order p, $\cong \mathbb{Z}_p$

+

Pairing $e : \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$

$$e(g^a, h^b) = e(g, h)^{a \cdot b}$$



$$\text{USK} = \left\{ \begin{array}{c} g^{r_1}, h^{r_2} \\ k_1 = g^{\alpha + r_1 \cdot b} \\ \cdots \end{array} \right\}$$

$$\text{CT} = \left\{ \begin{array}{c} e(g, h)^{\alpha \cdot s} \cdot \mathcal{M} \\ h^{s_1}, g^{s_2} \\ c_1 = g^{\lambda_1 + s_1 \cdot b} \\ \cdots \end{array} \right\}$$

Building Blocks

“Linear Algebra in Exponents”

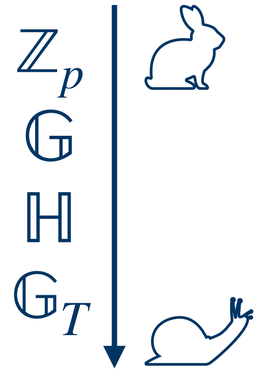
cyclic groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$

prime order p, $\cong \mathbb{Z}_p$

+

Pairing $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$

$$e(g^a, h^b) = e(g, h)^{a \cdot b}$$



$$\text{USK} = \left\{ \begin{array}{c} g^{r_1}, h^{r_2} \\ k_1 = g^{\alpha + r_1 \cdot b} \\ \dots \end{array} \right\}$$

$$\text{CT} = \left\{ \begin{array}{c} e(g, h)^{\alpha \cdot s} \cdot \mathcal{M} \\ h^{s_1}, g^{s_2} \\ c_1 = g^{\lambda_1 + s_1 \cdot b} \\ \dots \end{array} \right\}$$

Decrypt: $\dots \cdot e(k_1, h^{s_1}) \cdot e(c_1, h^{r_2}) \cdot \dots$

Our Scheme

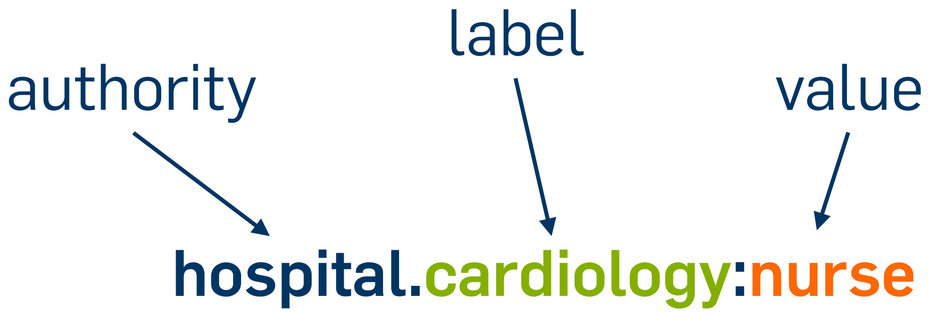
Advanced Features & Attributes

- Pairing-based
- Large-universe, CP-ABE scheme
- Multiple authorities
- Unlimited attribute re-uses
- Labeled negations (of values)

Our Scheme

Advanced Features & Attributes

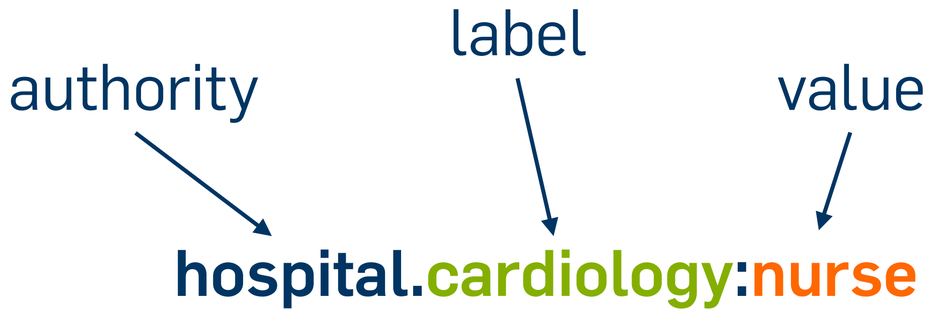
- Pairing-based
- Large-universe, CP-ABE scheme
- Multiple authorities
- Unlimited attribute re-uses
- Labeled negations (of values)

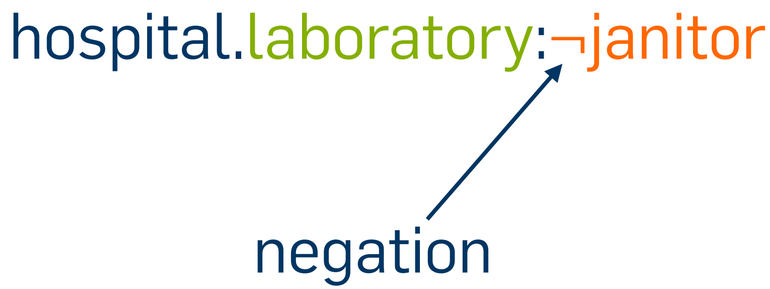


Our Scheme

Advanced Features & Attributes

- Pairing-based
- Large-universe, CP-ABE scheme
- Multiple authorities
- Unlimited attribute re-uses
- Labeled negations (of values)





Recap: ABE Squared

ABE Squared [1]

Optimizations & Benchmarking

- **Focus**: optimize key generation and encryption
  - shifting groups $\mathbb{G} \leftrightarrow \mathbb{H}$
- **Decryption:** *“as much as possible”*
- **Evaluation**:
  - 1 to 100 unique attributes
  - 3 schemes (Wat11, RW13, AC17) in 12 variants
  - Various curves

[1] "ABE Squared: Accurately Benchmarking Efficiency of Attribute-Based Encryption", de la Piedra et al., TCHES 2022

Our Structural Optimizations

Our Structural Optimizations

Introduction

- **Bottlenecks in decryption**: exponentiations and pairings

$$e(g_1, h) \cdot e(g_2, h) = e(g_1 \cdot g_2, h) \qquad e(g, h)^x = e(g^x, h) = e(g, h^x)$$

Our Structural Optimizations

Introduction

- **Bottlenecks in decryption**: exponentiations and pairings

$$e(g_1, h) \cdot e(g_2, h) = e(g_1 \cdot g_2, h) \qquad e(g, h)^x = e(g^x, h) = e(g, h^x)$$

- **Optimization idea:**
  - “enforce” more shared argument pairings

Our Structural Optimizations

Introduction

- **Bottlenecks in decryption**: exponentiations and pairings

$$e(g_1, h) \cdot e(g_2, h) = e(g_1 \cdot g_2, h) \qquad e(g, h)^x = e(g^x, h) = e(g, h^x)$$

- **Optimization idea:**
  - “enforce” more shared argument pairings

$$r_1 \text{ ——— } c_1 \qquad s_1 \text{ ——— } k_1$$

$$r_2 \text{ ——— } c_2 \qquad s_2 \text{ ——— } k_2$$

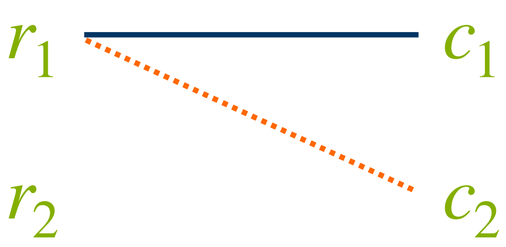
Our Structural Optimizations

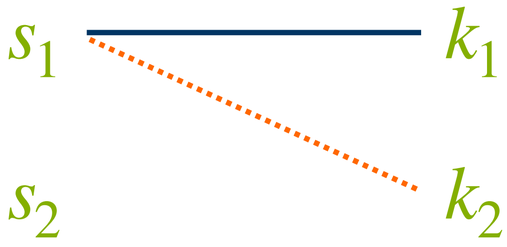
Introduction

- **Bottlenecks in decryption**: exponentiations and pairings

$$e(g_1, h) \cdot e(g_2, h) = e(g_1 \cdot g_2, h) \qquad e(g, h)^x = e(g^x, h) = e(g, h^x)$$

- **Optimization idea:**
  - “enforce” more shared argument pairings





Our Structural Optimizations

Rand. Re-Use for Keys

$\iota_{\text{auth}}(\mathcal{S})$:= “use **same** randomizer for **all** (attribute-)values unless label is also the same”

Security:
$r_{\text{val}} \cdot b_{\text{lbl}}$

Our Structural Optimizations

Rand. Re-Use for Keys

$\iota_{\text{auth}}(\mathcal{S})$:= “use **same** randomizer for **all** (attribute-)values unless label is also the same”

$\mathcal{S}$ =
A.x:a
A.x:b
A.y:a
A.y:c
A.z:d

Security:
$r_{\text{val}} \cdot b_{\text{lbl}}$

Our Structural Optimizations

Rand. Re-Use for Keys

$\iota_{\text{auth}}(\mathcal{S})$:= “use **same** randomizer for **all** (attribute-)values unless label is also the same”

$$\mathcal{S} = \begin{array}{ll} \text{A.x:a} & \longrightarrow r_a \\ \text{A.x:b} & \longrightarrow r_b \\ \text{A.y:a} & \longrightarrow r_a \\ \text{A.y:c} & \longrightarrow r_c \\ \text{A.z:d} & \longrightarrow r_d \end{array}$$

Security:
$r_{\text{val}} \cdot b_{\text{lbl}}$

Our Structural Optimizations

Rand. Re-Use for Keys

$\iota_{\text{auth}}(\mathcal{S})$:= “use **same** randomizer for **all** (attribute-)values unless label is also the same”

$\mathcal{S}$ =

A.x:a → r_a ⇢ ι_{auth} ⇢ r_1

A.x:b → r_b

A.y:a → r_a

A.y:c → r_c

A.z:d → r_d

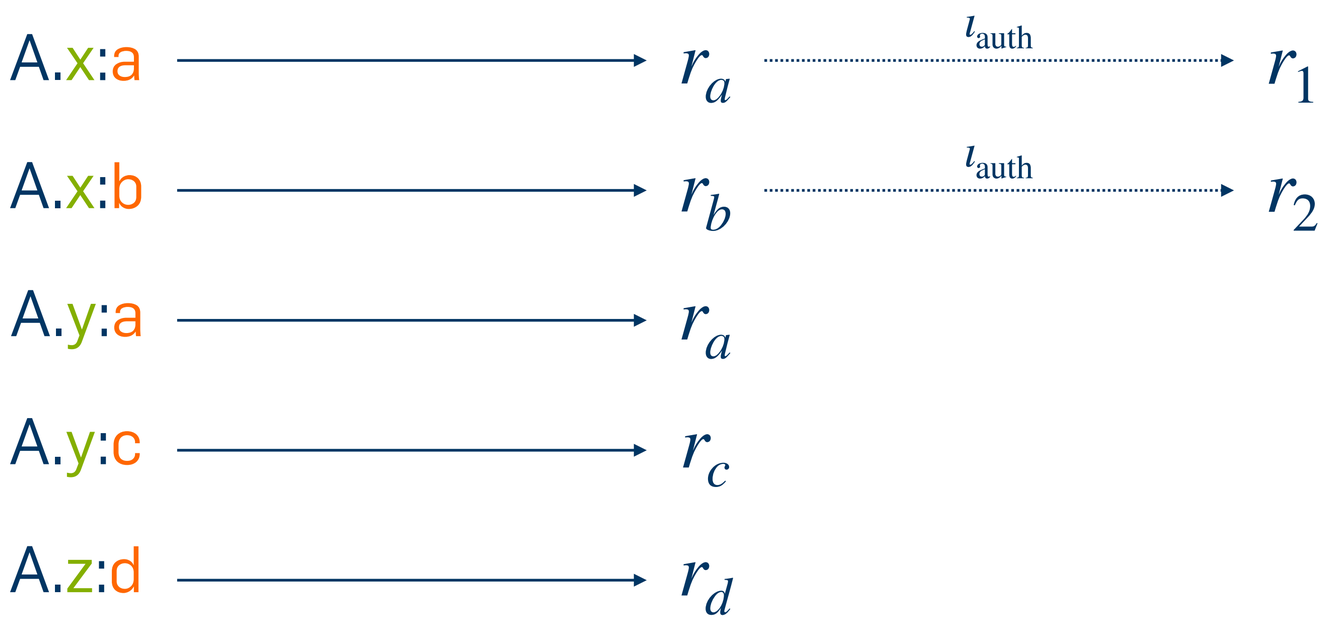
Security: $r_{\text{val}} \cdot b_{\text{lbl}}$

Our Structural Optimizations

Rand. Re-Use for Keys

$\iota_{\text{auth}}(\mathcal{S})$:= “use **same** randomizer for **all** (attribute-)values unless label is also the same”

$\mathcal{S}$ =



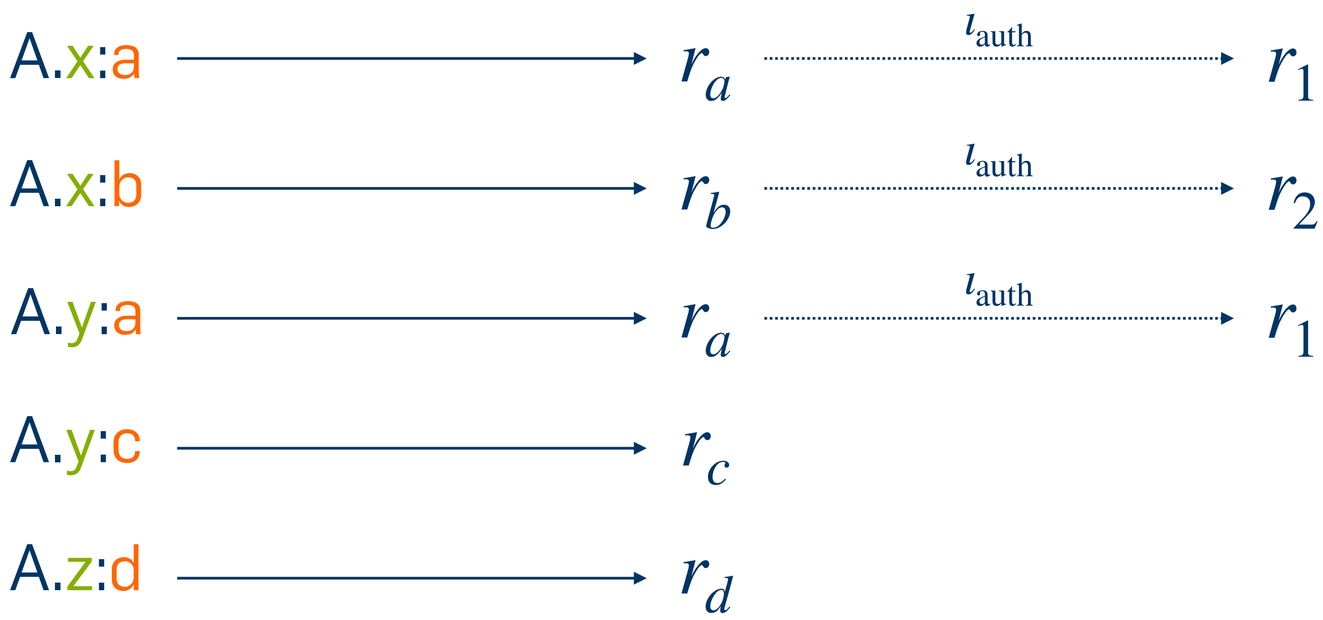
Security:
$r_{\text{val}} \cdot b_{\text{lbl}}$

Our Structural Optimizations

Rand. Re-Use for Keys

$\iota_{\text{auth}}(\mathcal{S})$:= “use **same** randomizer for **all** (attribute-)values unless label is also the same”

$\mathcal{S}$ =

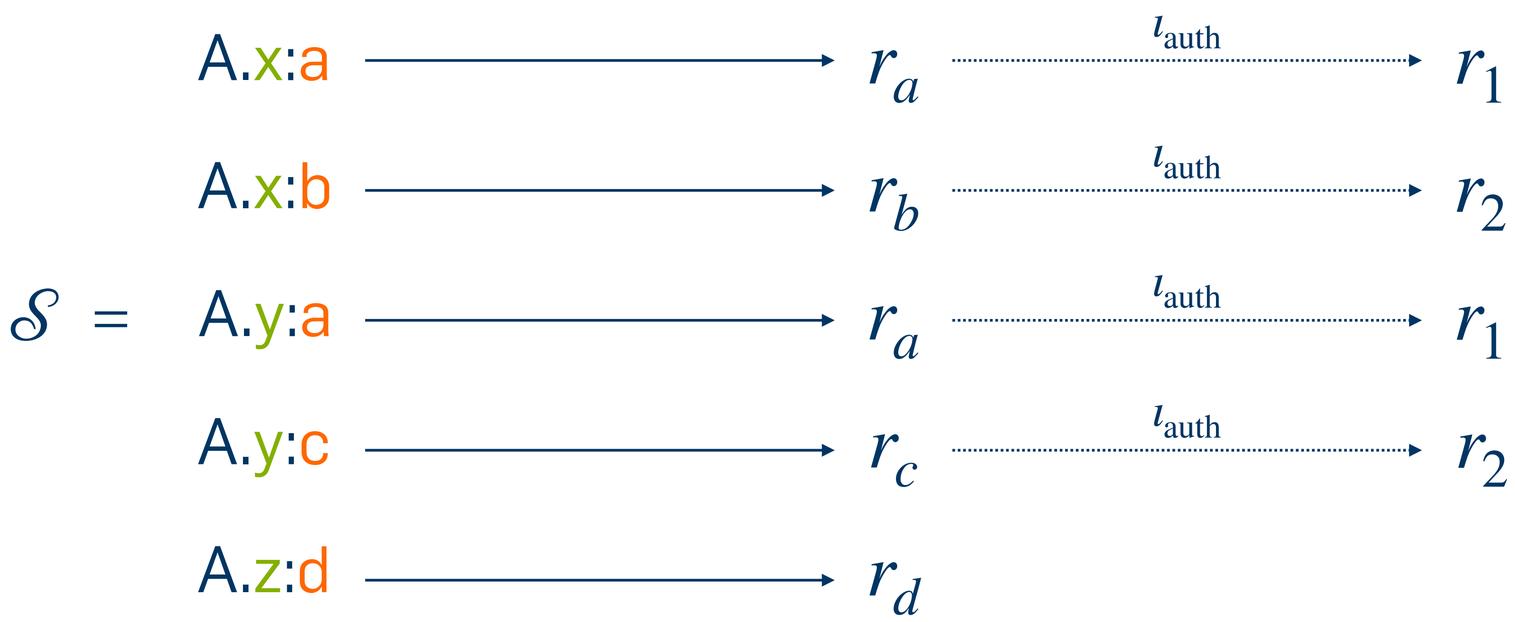


Security:
$r_{\text{val}} \cdot b_{\text{lbl}}$

Our Structural Optimizations

Rand. Re-Use for Keys

$\iota_{\text{auth}}(\mathcal{S})$:= “use **same** randomizer for **all** (attribute-)values unless label is also the same”



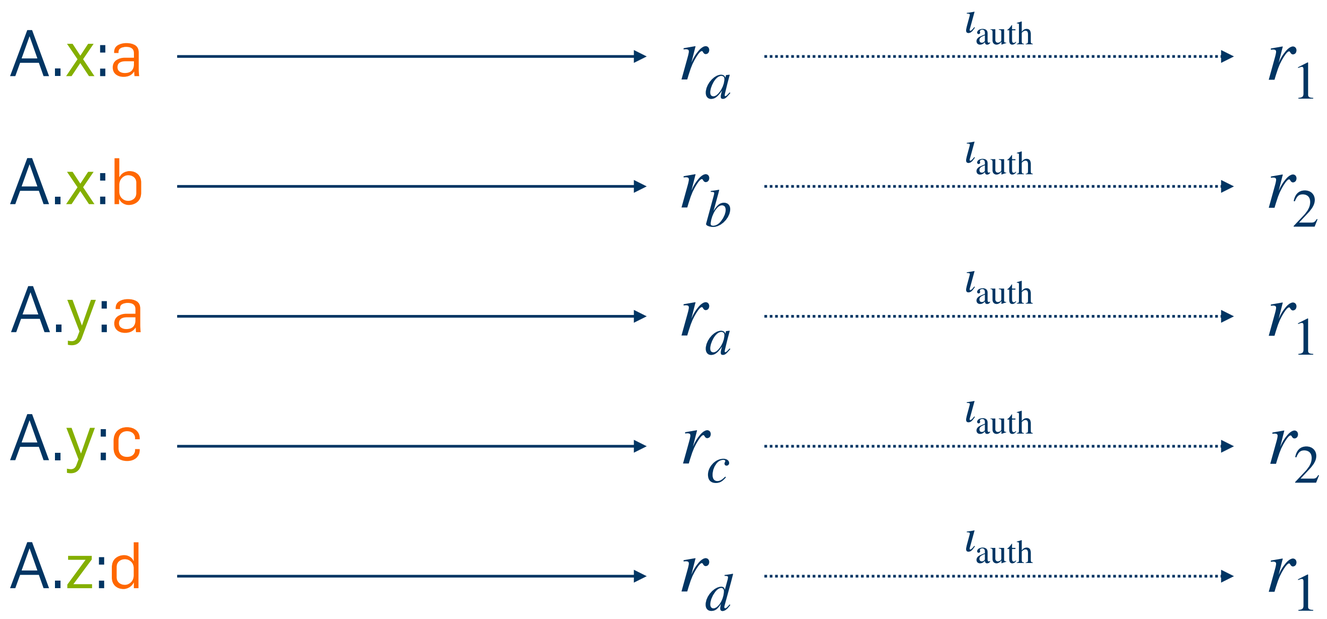
Security:
$r_{\text{val}} \cdot b_{\text{lbl}}$

Our Structural Optimizations

Rand. Re-Use for Keys

$\iota_{\text{auth}}(\mathcal{S})$:= “use **same** randomizer for **all** (attribute-)values unless label is also the same”

$\mathcal{S}$ =



Security:
$r_{\text{val}} \cdot b_{\text{lbl}}$

Our Structural Optimizations

Rand. Re-Use for Ciphertexts

$\tau_{\tilde{\rho}}([n_1])$:= “use **same** randomizer for **all** LSSS rows unless authority is also the same”

Security:
$s_j \cdot b_{\text{auth}}$
$s_j \cdot b_{\text{auth,lbl}}$

Our Structural Optimizations

Rand. Re-Use for Ciphertexts

$\tau_{\tilde{\rho}}([n_1])$:= “use **same** randomizer for **all** LSSS rows unless authority is also the same”

$$\mathbb{A} = \begin{array}{c} \text{A.x:a} \\ \wedge \\ \text{A.y:b} \\ \wedge \\ \text{B.y:a} \\ \wedge \\ \text{C.y:a} \\ \wedge \\ \text{D.x:a} \end{array}$$

Security:

$s_j \cdot b_{\text{auth}}$

$s_j \cdot b_{\text{auth,lbl}}$

Our Structural Optimizations

Rand. Re-Use for Ciphertexts

$\tau_{\tilde{\rho}}([n_1])$:= "use **same** randomizer for **all** LSSS rows unless authority is also the same"

$$\mathbb{A} = \begin{matrix} \text{A.x:a} \\ \wedge \\ \text{A.y:b} \\ \wedge \\ \text{B.y:a} \\ \wedge \\ \text{C.y:a} \\ \wedge \\ \text{D.x:a} \end{matrix} \longrightarrow \begin{matrix} s_1 \\ \\ s_2 \\ \\ s_3 \\ \\ s_4 \\ \\ s_5 \end{matrix}$$

Security:

$s_j \cdot b_{\text{auth}}$

$s_j \cdot b_{\text{auth,lbl}}$

Our Structural Optimizations

Rand. Re-Use for Ciphertexts

$\tau_{\tilde{\rho}}([n_1])$:= "use **same** randomizer for **all** LSSS rows unless authority is also the same"

$$\mathbb{A} = \begin{array}{l} \text{A.x:a} \\ \wedge \\ \text{A.y:b} \\ \wedge \\ \text{B.y:a} \\ \wedge \\ \text{C.y:a} \\ \wedge \\ \text{D.x:a} \end{array} \longrightarrow \begin{array}{l} s_1 \\ \\ s_2 \\ \\ s_3 \\ \\ s_4 \\ \\ s_5 \end{array} \overset{\tau_{\tilde{\rho}}}{\dashrightarrow} \begin{array}{l} s_1 \\ \\ s_2 \\ \\ s_1 \\ \\ s_1 \\ \\ s_1 \end{array}$$

Security:

$s_j \cdot b_{\text{auth}}$

$s_j \cdot b_{\text{auth,lbl}}$

Our Structural Optimizations

Rand. Splitting for Key (pos.)

$\tau([n_1]) :=$ “use **same** randomizer for **all** LSSS rows unless authority **and** label is also the same”

Security:

$s_j \cdot b_{\text{auth}}$

$s_j \cdot b_{\text{auth,lbl}}$

Our Structural Optimizations

Rand. Splitting for Key (pos.)

$\tau([n_1]) :=$ "use **same** randomizer for **all** LSSS rows unless authority **and** label is also the same"

$$k_{\text{auth,lbl,val}} = \alpha_{\text{auth}} + r_{\text{GID}} \cdot b_{\text{auth}} + r_{\text{auth},\iota} \cdot (b^{(0)}_{\text{auth,lbl}} + x_{\text{val}} \cdot b^{(1)}_{\text{auth,lbl}})$$

Security:

$s_j \cdot b_{\text{auth}}$

$s_j \cdot b_{\text{auth,lbl}}$

Our Structural Optimizations

Rand. Splitting for Key (pos.)

$\tau([n_1]) :=$ "use **same** randomizer for **all** LSSS rows unless authority **and** label is also the same"

$$k_{\text{auth,lbl,val}} = \alpha_{\text{auth}} + r_{\text{GID}} \cdot b_{\text{auth}} + r_{\text{auth},\iota} \cdot (b^{(0)}_{\text{auth,lbl}} + x_{\text{val}} \cdot b^{(1)}_{\text{auth,lbl}})$$

Security:

$s_j \cdot b_{\text{auth}}$

$s_j \cdot b_{\text{auth,lbl}}$

Our Structural Optimizations

Rand. Splitting for Key (pos.)

$\tau([n_1]) :=$ "use **same** randomizer for **all** LSSS rows unless authority **and** label is also the same"

$$k_{\text{auth,lbl,val}} = \alpha_{\text{auth}} + r_{\text{GID}} \cdot b_{\text{auth}} + r_{\text{auth},\iota} \cdot (b^{(0)}_{\text{auth,lbl}} + x_{\text{val}} \cdot b^{(1)}_{\text{auth,lbl}})$$

$$\rightarrow k^{1}_{\text{auth},\iota} = \alpha_{\text{auth}} + r_{\text{GID}} \cdot b_{\text{auth}} + r_{\text{auth},\iota} \cdot b'_{\text{auth}}$$

$$\rightarrow k^{2}_{\text{auth,lbl},\iota} = r_{\text{auth},\iota} \cdot (b^{(0)}_{\text{auth,lbl}} + x_{\text{val}} \cdot b^{(1)}_{\text{auth,lbl}})$$

Security:

$s_j \cdot b_{\text{auth}}$

$s_j \cdot b_{\text{auth,lbl}}$

Our Structural Optimizations

Rand. Splitting for Key (pos.)

$\tau([n_1]) :=$ "use **same** randomizer for **all** LSSS rows unless authority **and** label is also the same"

$$k_{\text{auth,lbl,val}} = \alpha_{\text{auth}} + r_{\text{GID}} \cdot b_{\text{auth}} + r_{\text{auth},\iota} \cdot (b^{(0)}_{\text{auth,lbl}} + x_{\text{val}} \cdot b^{(1)}_{\text{auth,lbl}})$$

$$\rightarrow k^{1}_{\text{auth},\iota} = \alpha_{\text{auth}} + r_{\text{GID}} \cdot b_{\text{auth}} + r_{\text{auth},\iota} \cdot b'_{\text{auth}}$$

$$\rightarrow k^{2}_{\text{auth,lbl},\iota} = r_{\text{auth},\iota} \cdot (b^{(0)}_{\text{auth,lbl}} + x_{\text{val}} \cdot b^{(1)}_{\text{auth,lbl}})$$

Security:

$s_j \cdot b_{\text{auth}}$

$s_j \cdot b_{\text{auth,lbl}}$

Our Structural Optimizations

Rand. Splitting for Key (pos.)

$\tau([n_1]) :=$ "use **same** randomizer for **all** LSSS rows unless authority **and** label is also the same"

$$k_{\text{auth,lbl,val}} = \alpha_{\text{auth}} + r_{\text{GID}} \cdot b_{\text{auth}} + r_{\text{auth},\iota} \cdot (b^{(0)}_{\text{auth,lbl}} + x_{\text{val}} \cdot b^{(1)}_{\text{auth,lbl}})$$

$$k^1_{\text{auth},\iota} = \alpha_{\text{auth}} + r_{\text{GID}} \cdot b_{\text{auth}} + r_{\text{auth},\iota} \cdot b'_{\text{auth}} \dashrightarrow s_{\tau_{\tilde{\rho}}(j)}$$

$$k^2_{\text{auth,lbl},\iota} = r_{\text{auth},\iota} \cdot (b^{(0)}_{\text{auth,lbl}} + x_{\text{val}} \cdot b^{(1)}_{\text{auth,lbl}}) \dashrightarrow s_{\tau(j)}$$

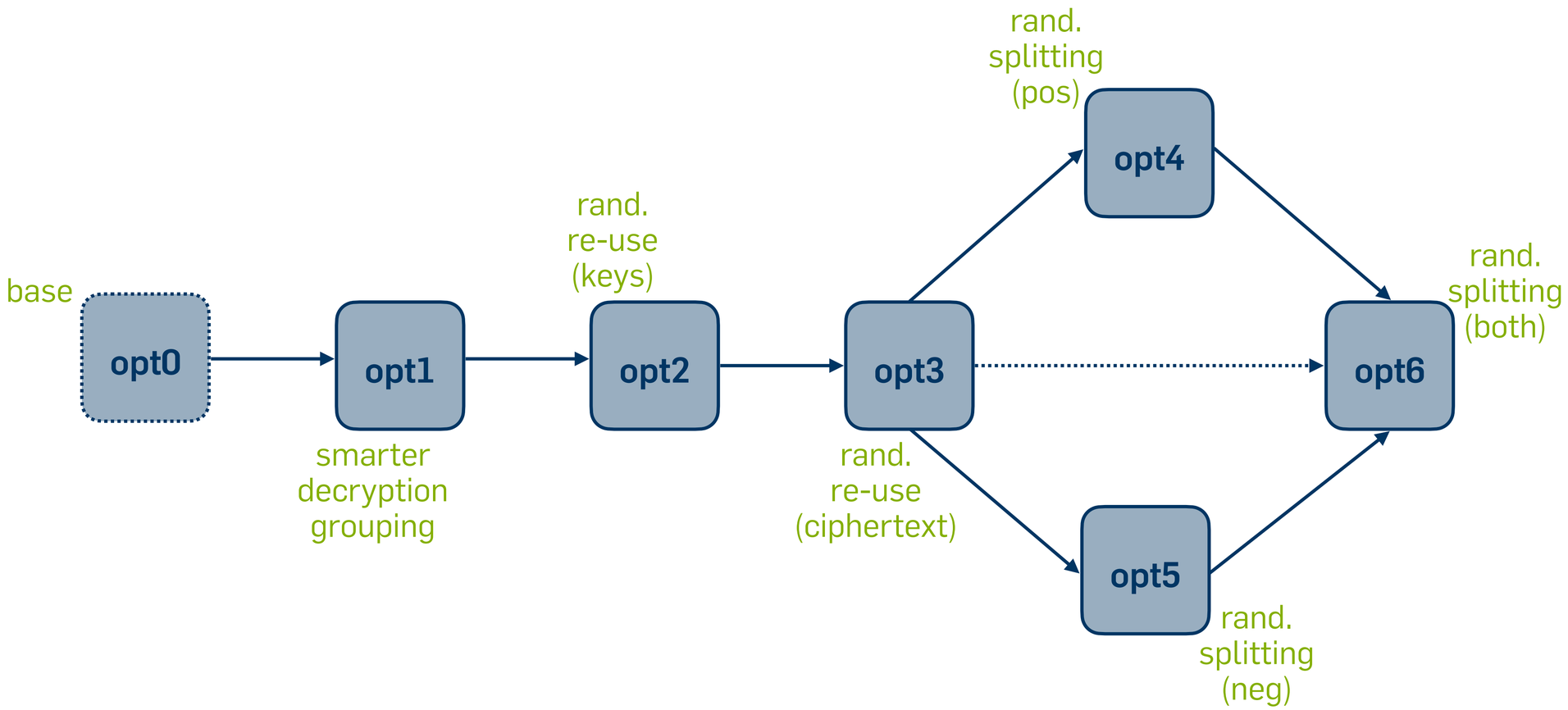
Security:

$s_j \cdot b_{\text{auth}}$

$s_j \cdot b_{\text{auth,lbl}}$

Our Structural Optimizations

Overview



Our Benchmarking Strategy

Our Benchmarking Strategy

The “ABE Squared” Strategy

- **ABE Squared**: everything is **unique**, nothing is **negated**

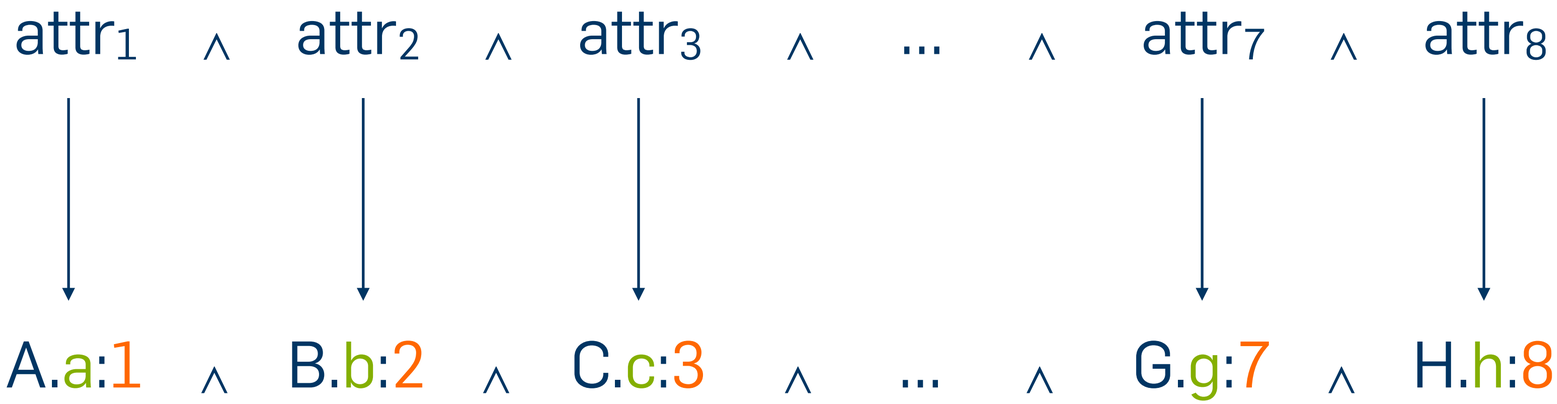
$$\text{attr}_1 \wedge \text{attr}_2 \wedge \text{attr}_3 \wedge \ldots \wedge \text{attr}_7 \wedge \text{attr}_8$$

- **ABE Cubed**:

Our Benchmarking Strategy

The “ABE Squared” Strategy

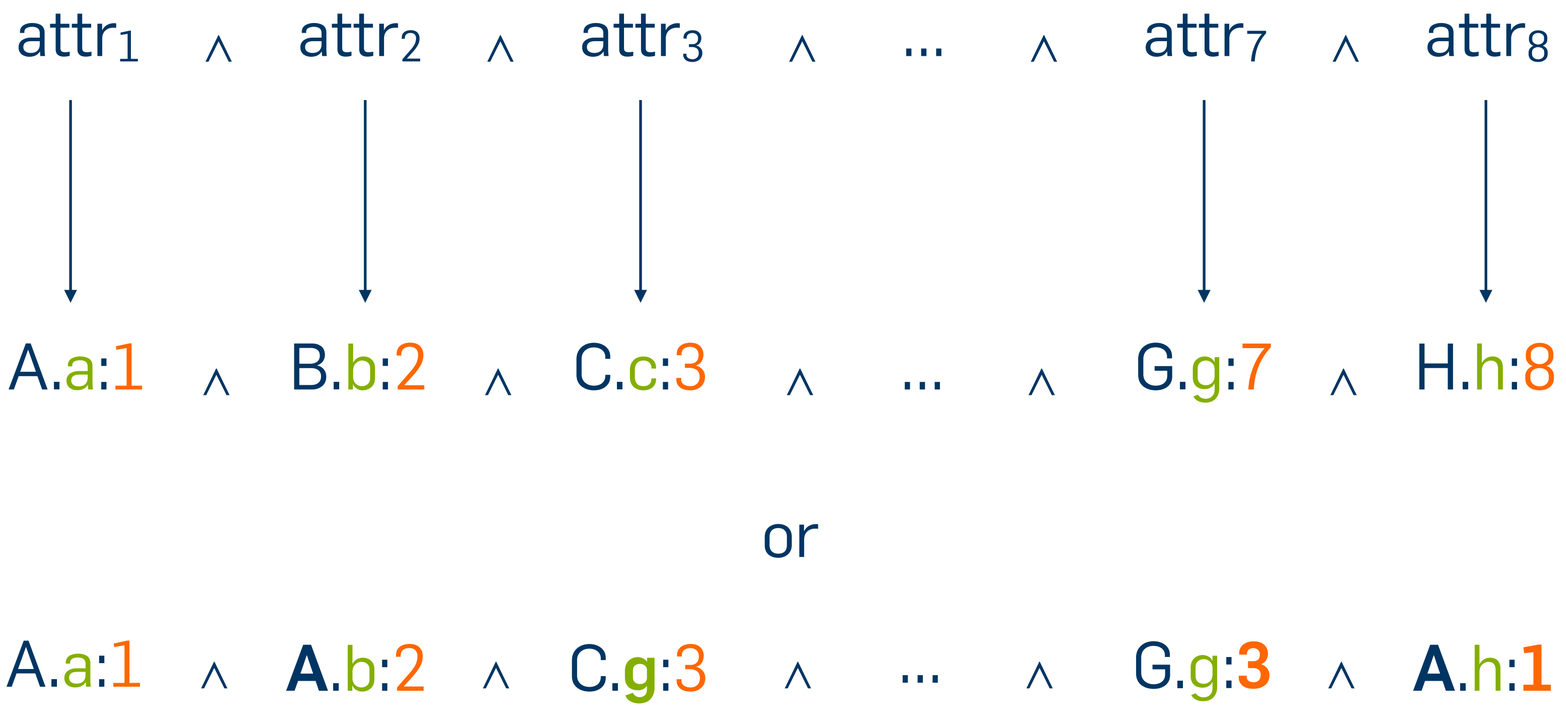
- **ABE Squared**: everything is **unique**, nothing is **negated**
- **ABE Cubed**:



Our Benchmarking Strategy

The “ABE Squared” Strategy

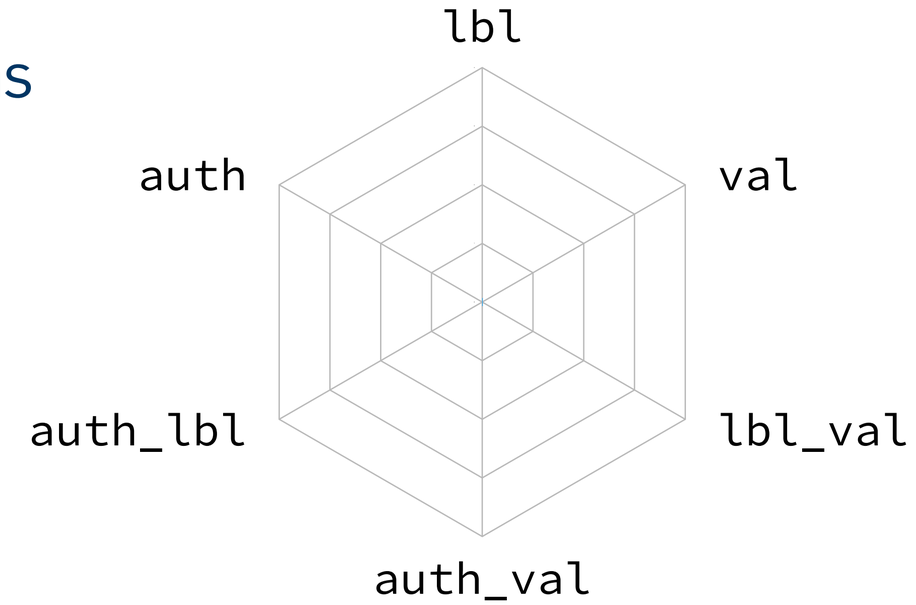
- **ABE Squared**: everything is **unique**, nothing is **negated**
- **ABE Cubed**:



Our Benchmarking Strategy

Duplication Dimensions

- Six dimensions to systematically introduce duplications
- Parameter: *n* (many divisors)

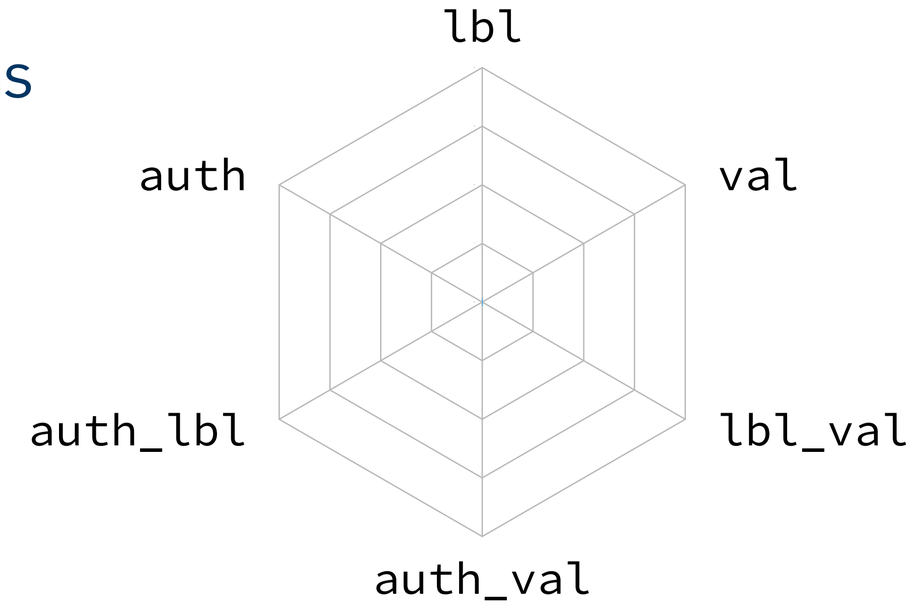


Our Benchmarking Strategy

Duplication Dimensions

- Six dimensions to systematically introduce duplications
- Parameter: *n* (many divisors)

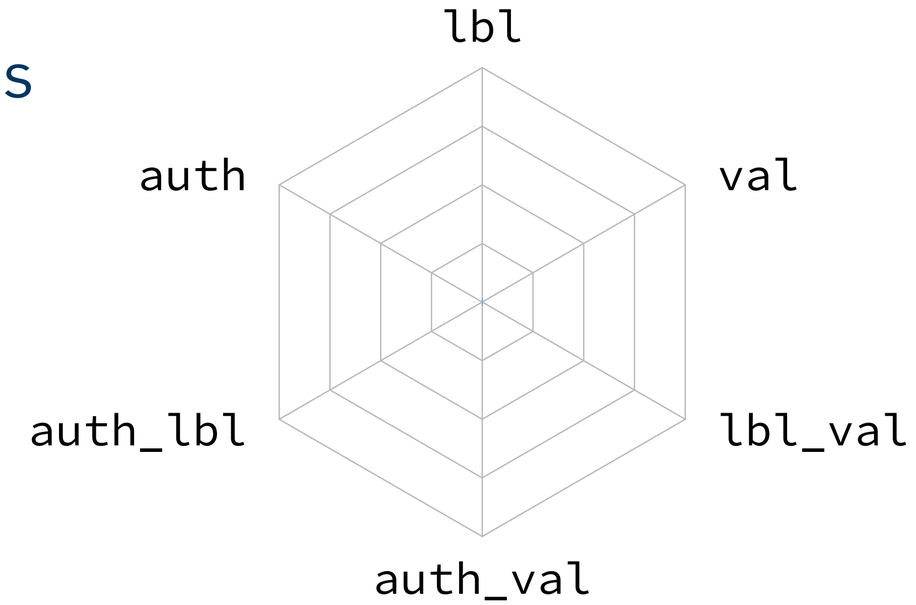
Example: `val`, n=4



Our Benchmarking Strategy

Duplication Dimensions

- Six dimensions to systematically introduce duplications
- Parameter: *n* (many divisors)



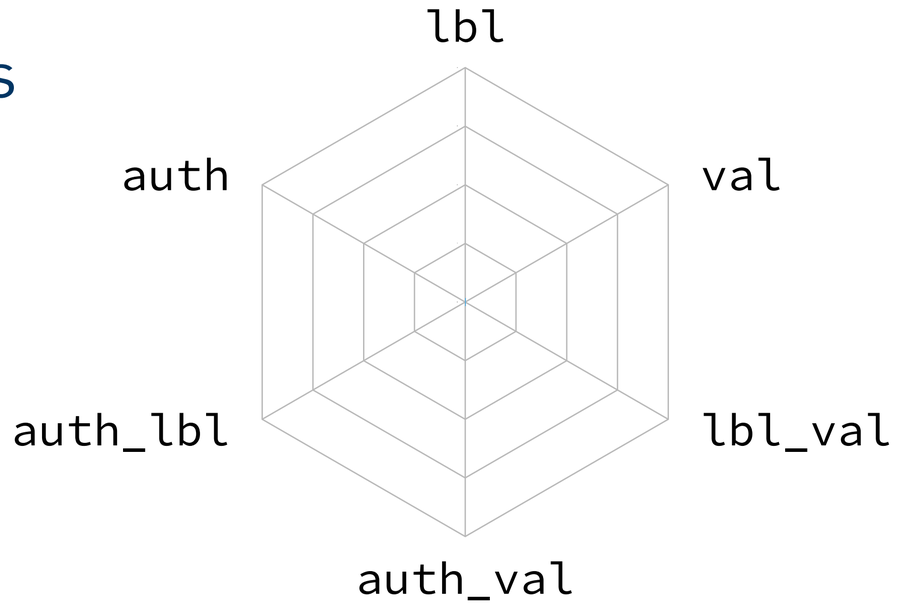
Example: `val`, n=4

1	A.w:	B.x:	C.y:	D.z:

Our Benchmarking Strategy

Duplication Dimensions

- Six dimensions to systematically introduce duplications
- Parameter: *n* (many divisors)



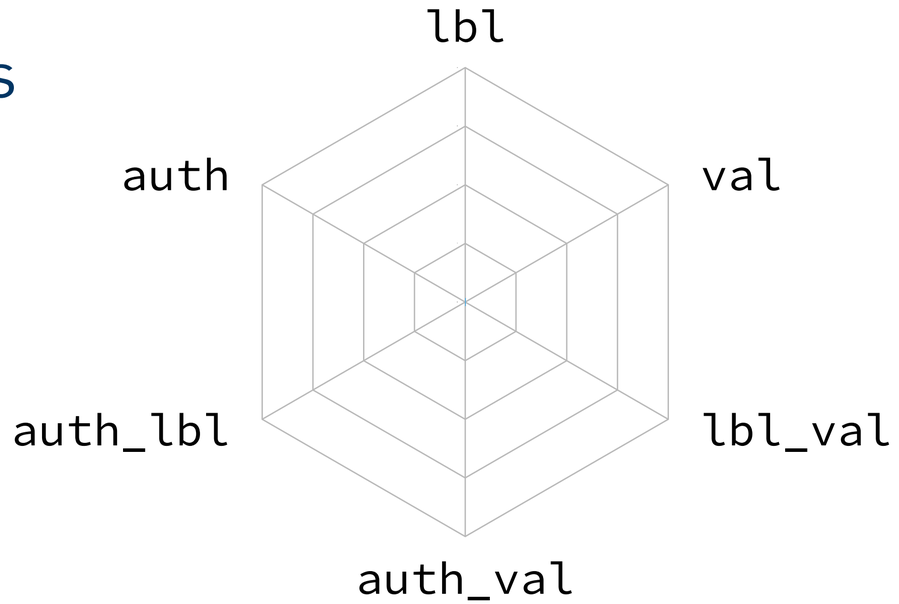
Example: `val`, n=4

1 A.w:1 B.x:1 C.y:1 D.z:1

Our Benchmarking Strategy

Duplication Dimensions

- Six dimensions to systematically introduce duplications
- Parameter: *n* (many divisors)



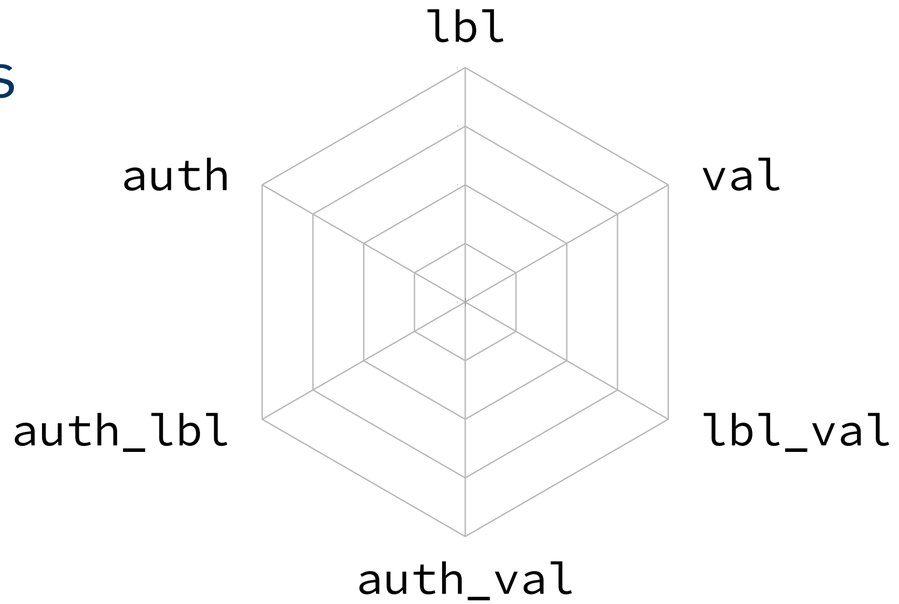
Example: `val`, n=4

1	A.w:1	B.x:1	C.y:1	D.z:1
2	A.w:	B.x:	C.y:	D.z:

Our Benchmarking Strategy

Duplication Dimensions

- Six dimensions to systematically introduce duplications
- Parameter: *n* (many divisors)



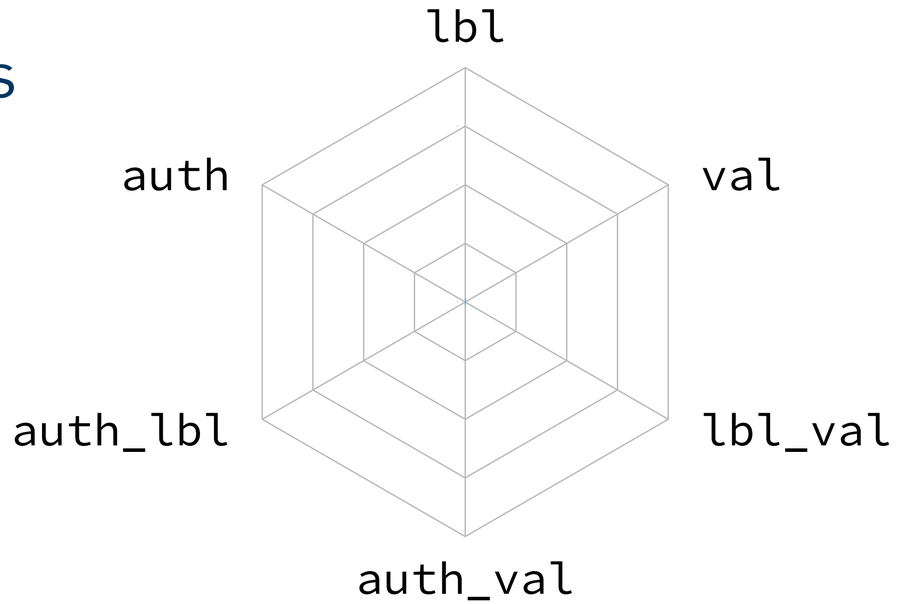
Example: `val`, n=4

1	A.w:**1**	B.x:**1**	C.y:**1**	D.z:**1**
2	A.w:**1**	B.x:**1**	C.y:**2**	D.z:**2**

Our Benchmarking Strategy

Duplication Dimensions

- Six dimensions to systematically introduce duplications
- Parameter: *n* (many divisors)



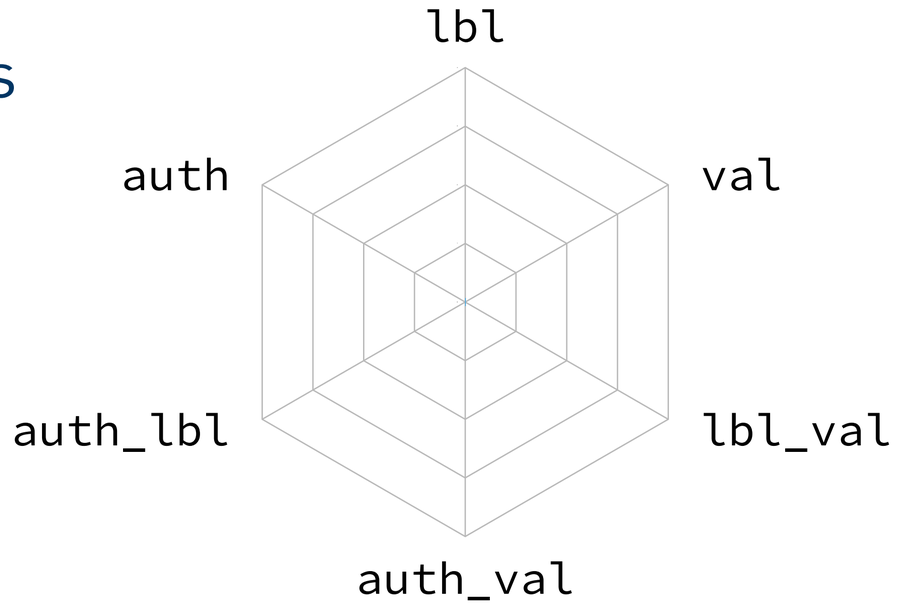
Example: `val`, n=4

1	A.w:**1**	B.x:**1**	C.y:**1**	D.z:**1**
2	A.w:**1**	B.x:**1**	C.y:**2**	D.z:**2**
4	A.w:	B.x:	C.y:	D.z:

Our Benchmarking Strategy

Duplication Dimensions

- Six dimensions to systematically introduce duplications
- Parameter: *n* (many divisors)



Example: `val`, n=4

1	A.w:**1**	B.x:**1**	C.y:**1**	D.z:**1**
2	A.w:**1**	B.x:**1**	C.y:**2**	D.z:**2**
4	A.w:**1**	B.x:**2**	C.y:**3**	D.z:**4**

Our Benchmarking Strategy

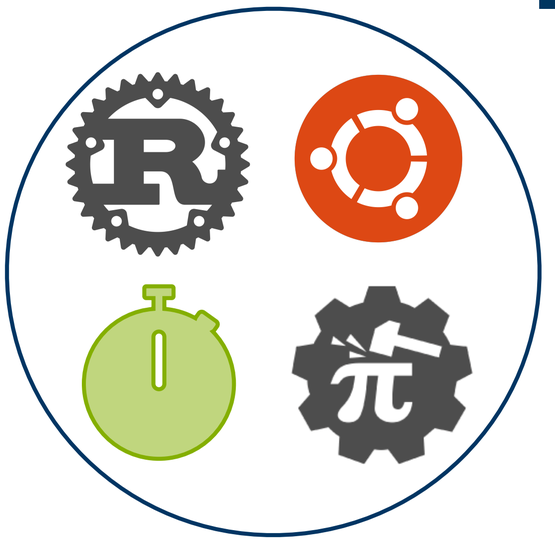
Overview

- **Phase 1a**:
  - Everything globally unique (⇒ ABE Squared)
- **Phase 1b**:
  - Everything globally unique
  - Various negation degrees d: $|\mathcal{S}| = d \cdot |\text{policy}|$
- **Phase 2**:
  - Explore all six dimensions

Evaluation & Results

Evaluation

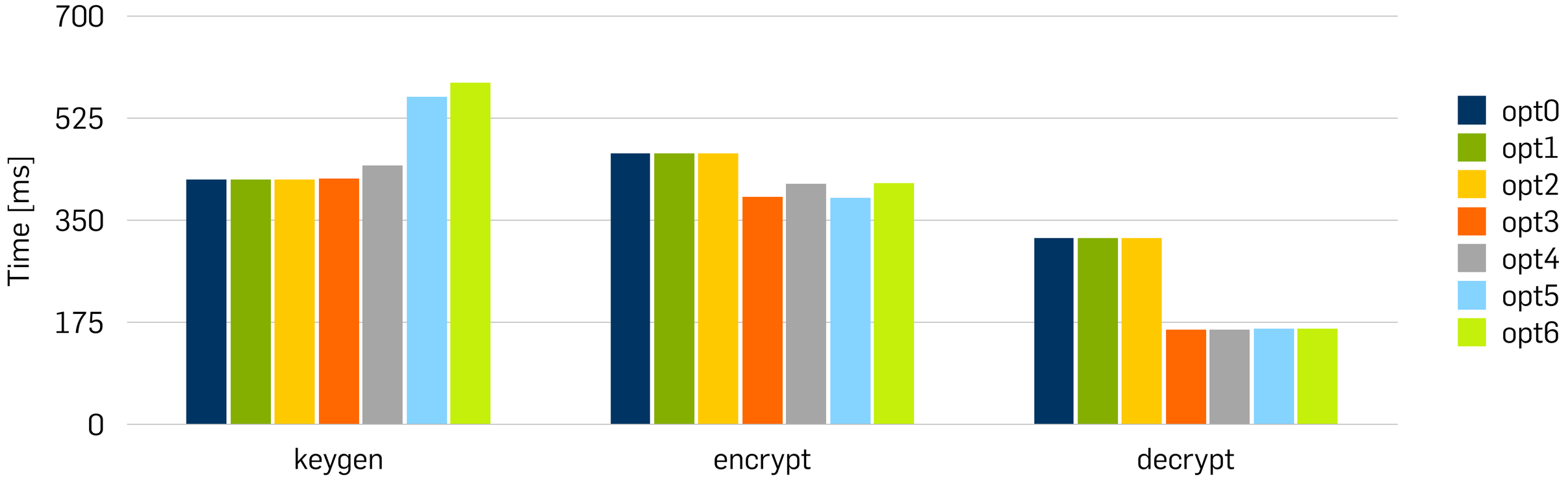
Results



- **Fully functional** implementation in Rust
  - arkworks: BLS12-381 pairing-friendly curve
  - Criterion: benchmarking framework
- **Setup**: AMD Server, Ubuntu 22.04 VM, 8 ph./16 v. cores, 32GB RAM
- 6 opts × 6 dimensions × #divisors(60) × … = **huge data set**
  - >18.000 data points (in artifact)

Evaluation

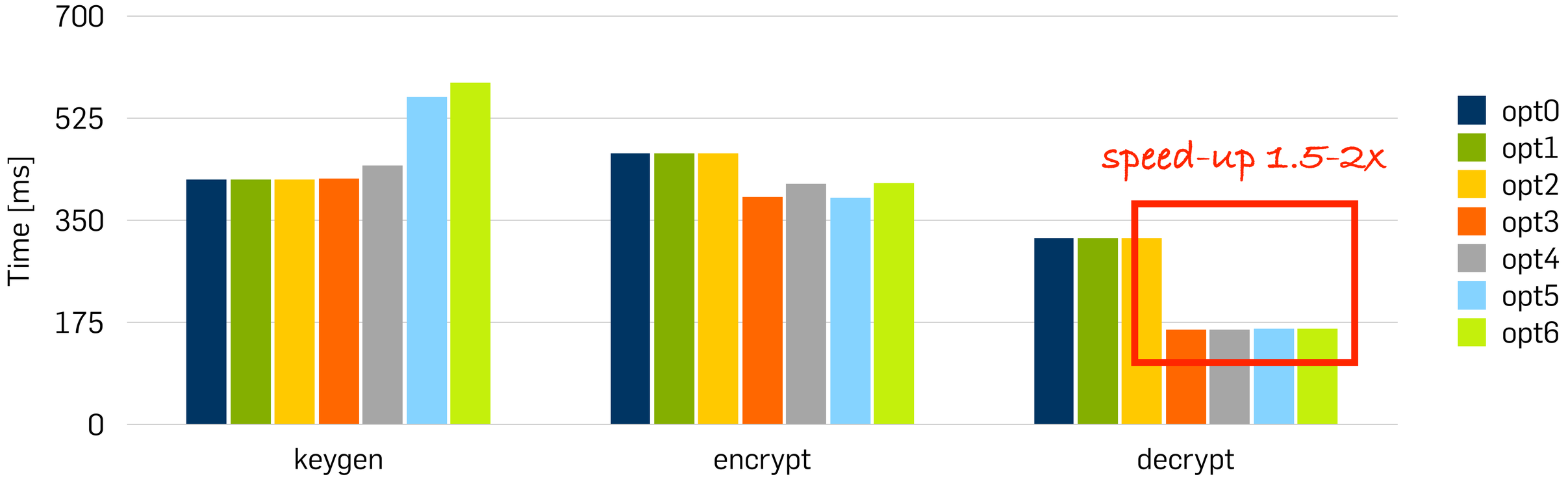
Phase 1a



Policy length 100, positive literals, all unique

Evaluation

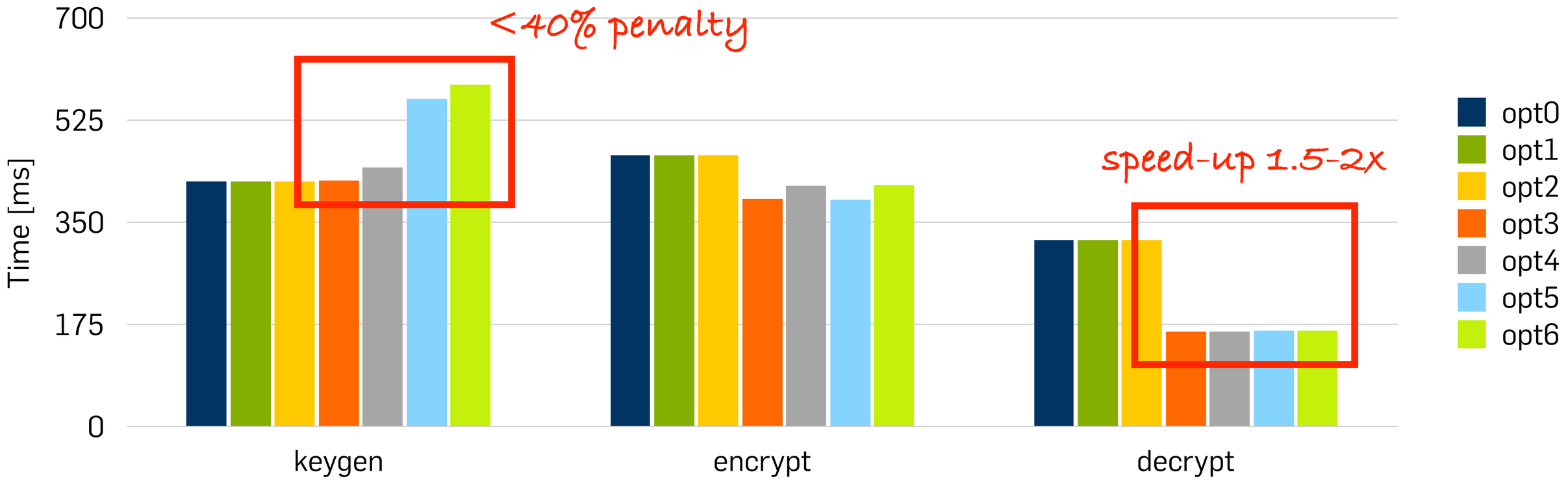
Phase 1a



Policy length 100, positive literals, all unique

Evaluation

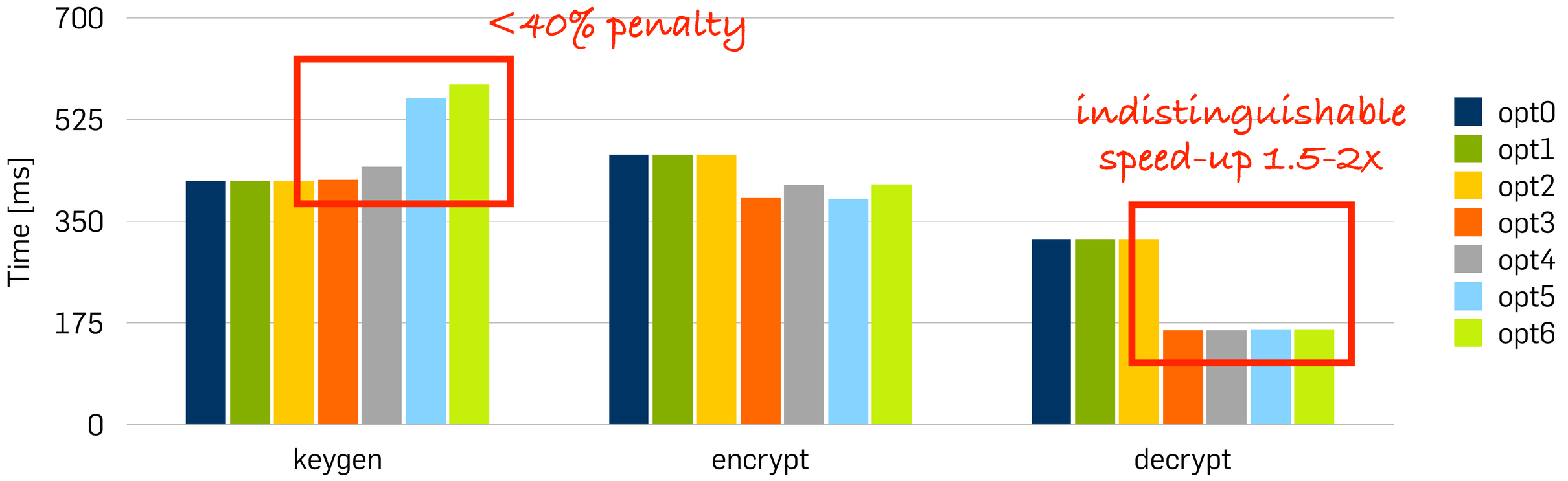
Phase 1a



Policy length 100, positive literals, all unique

Evaluation

Phase 1a



Policy length 100, positive literals, all unique

Evaluation

Phase 2 - Isolating what Matters

	no_dup			auth			lbl			val			auth_lbl			auth_val			lbl_attr		
Scheme	KG	E	D	KG	E	D	KG	E	D	KG	E	D	KG	E	D	KG	E	D	KG	E	D
opt0	-	-	-																		
opt1	-	-	-																		
opt2	-	-	-																		
opt3	-	-	-																		
opt4	-	-	-																		
opt5	-	-	-																		
opt6	-	-	-																		

Policy length 60, positive literals

Evaluation

Phase 2 - Isolating what Matters

	no_dup			auth			lbl			val			auth_lbl			auth_val			lbl_attr		
Scheme	KG	E	D	KG	E	D	KG	E	D	KG	E	D	KG	E	D	KG	E	D	KG	E	D
opt0	-	-	-	✓									✓	✓		✓					
opt1	-	-	-	✓									✓	✓		✓		✓			
opt2	-	-	-	✓		✓							✓	✓		✓		✓			
opt3	-	-	-	✓	✓	✓							✓	✓	✓	✓	✓	✓			
opt4	-	-	-	✓	✓	✓							✓	✓	✓	✓	✓	✓			
opt5	-	-	-	✓	✓	✓							✓	✓	✓	✓	✓	✓			
opt6	-	-	-	✓	✓	✓							✓	✓	✓	✓	✓	✓			

Policy length 60, positive literals

Evaluation

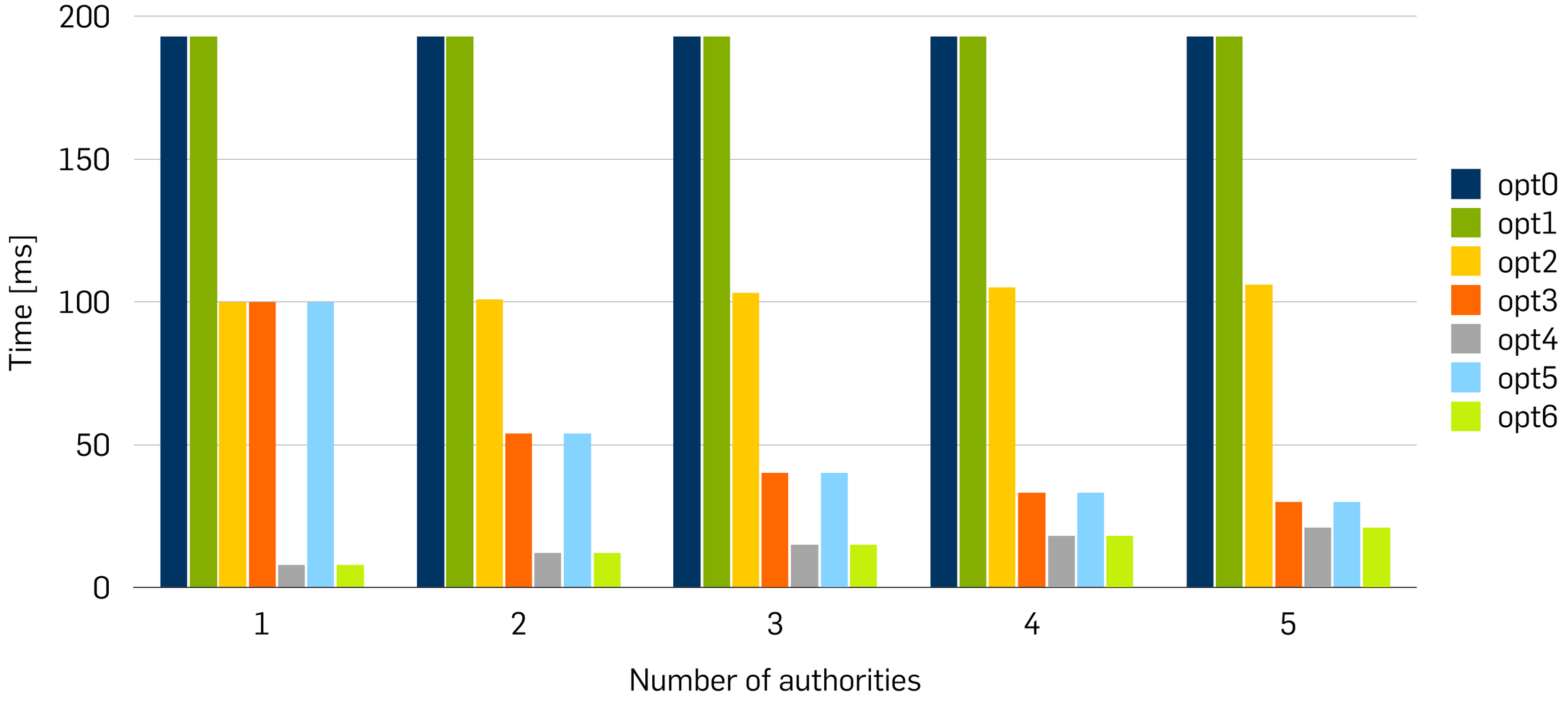
Phase 2 - Isolating what Matters

	no_dup			auth			~~lbl~~			~~val~~			auth_lbl			auth_val			~~lbl_attr~~		
Scheme	KG	E	D	KG	E	D	KG	E	D	KG	E	D	KG	E	D	KG	E	D	KG	E	D
opt0	-	-	-	✓									✓	✓		✓					
opt1	-	-	-	✓									✓	✓		✓		✓			
opt2	-	-	-	✓		✓							✓	✓		✓		✓			
opt3	-	-	-	✓	✓	✓							✓	✓	✓	✓	✓	✓			
opt4	-	-	-	✓	✓	✓							✓	✓	✓	✓	✓	✓			
opt5	-	-	-	✓	✓	✓							✓	✓	✓	✓	✓	✓			
opt6	-	-	-	✓	✓	✓							✓	✓	✓	✓	✓	✓			

Policy length 60, positive literals

Evaluation

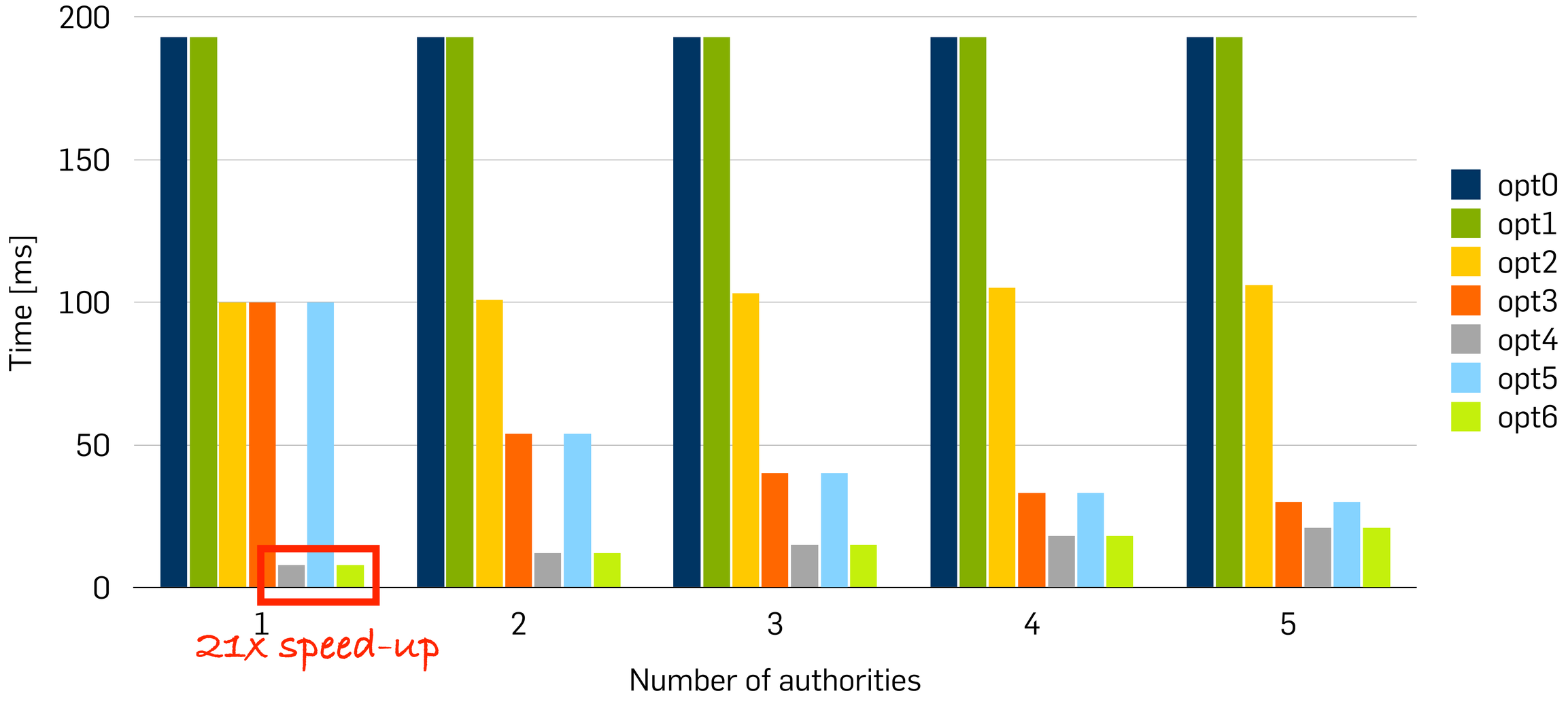
Phase 2



Decryption, policy length = 60, positive literals

Evaluation

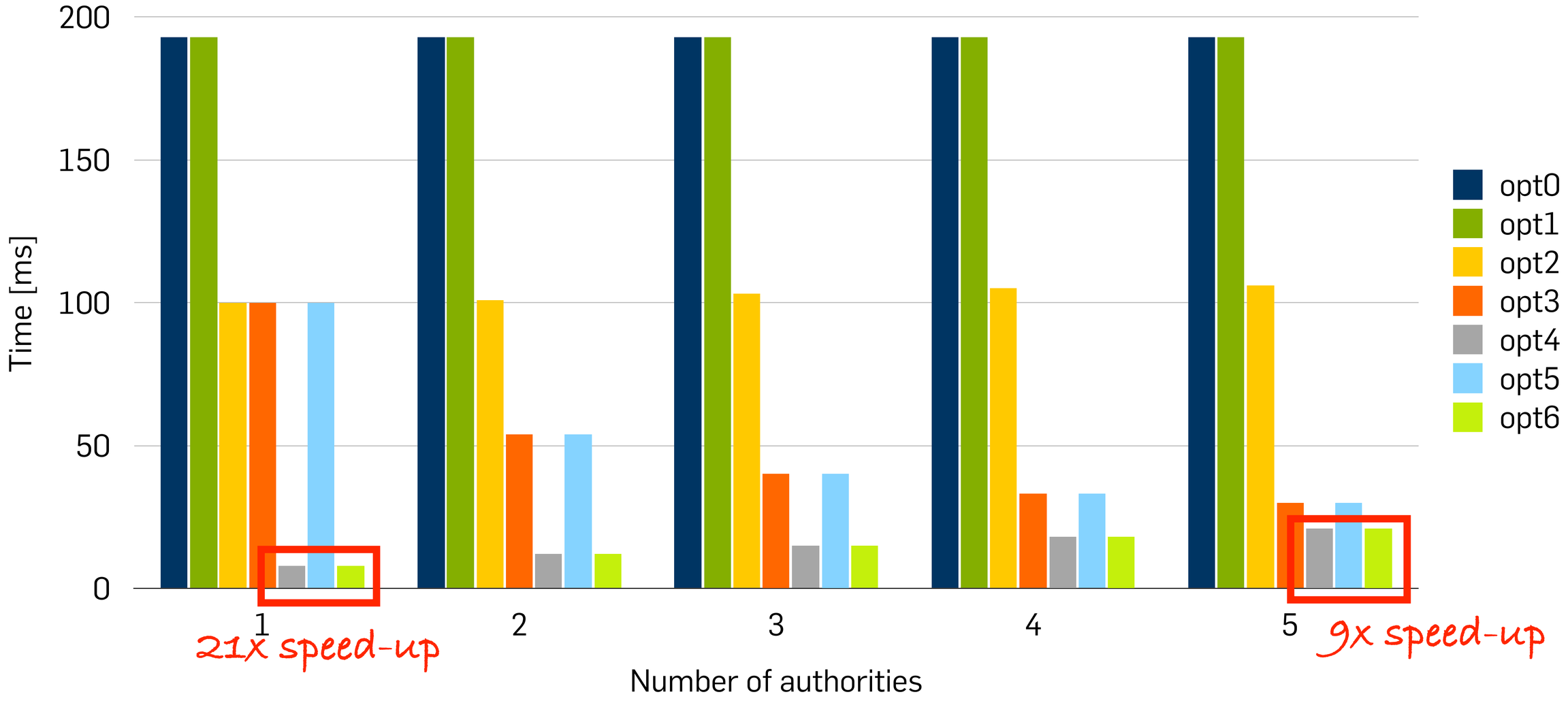
Phase 2



Decryption, policy length = 60, positive literals

Evaluation

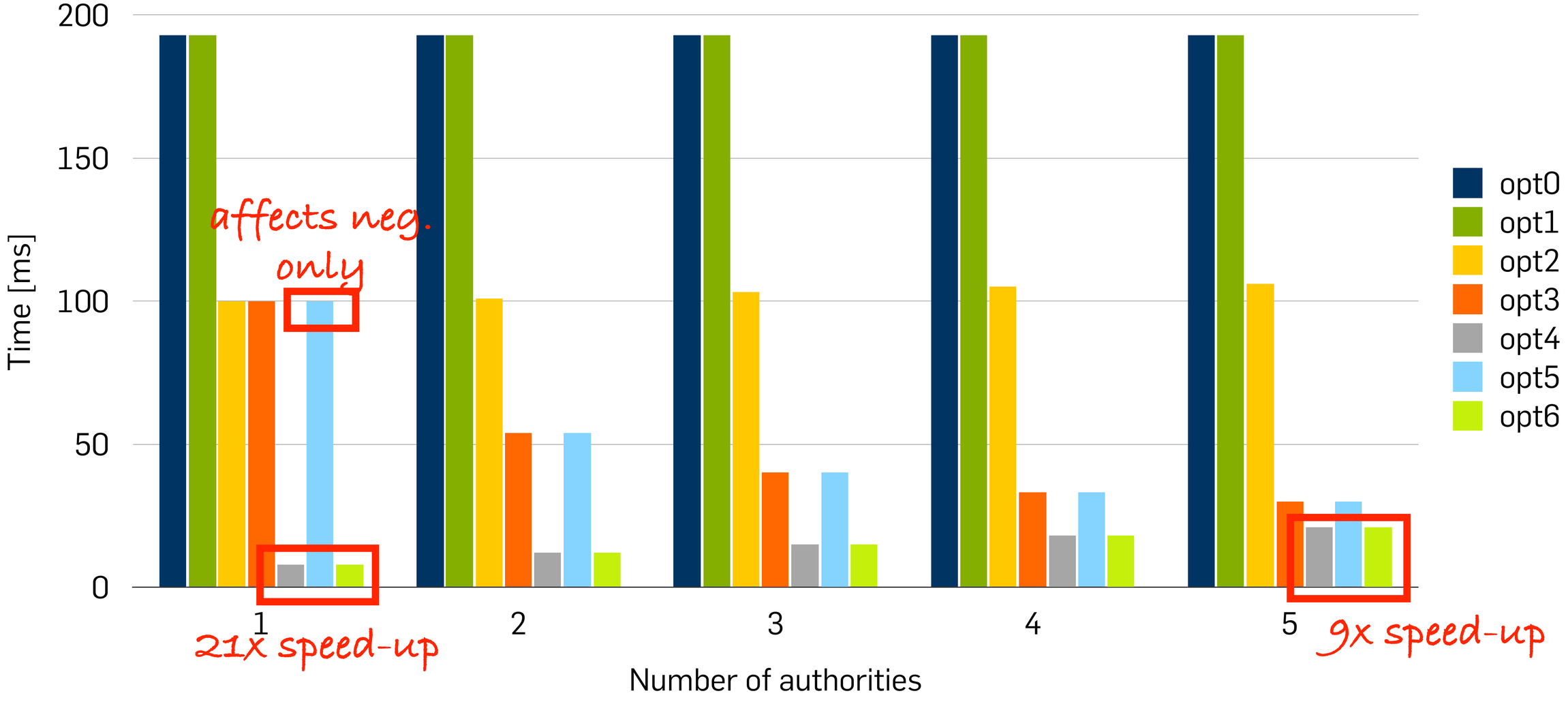
Phase 2



Decryption, policy length = 60, positive literals

Evaluation

Phase 2



Decryption, policy length = 60, positive literals

Evaluation

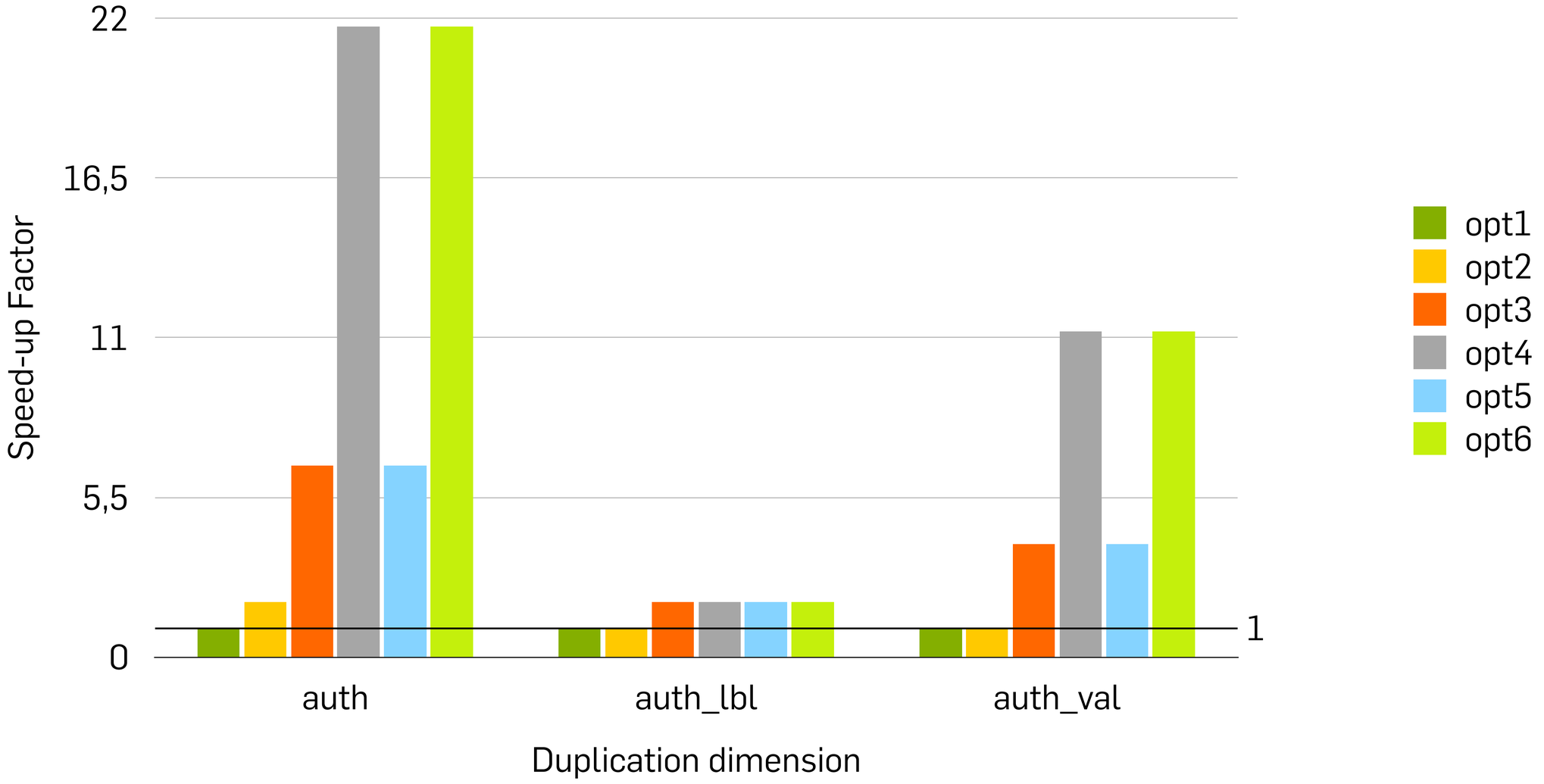
Phase 2



Decryption, policy length = 60, positive literals

Evaluation

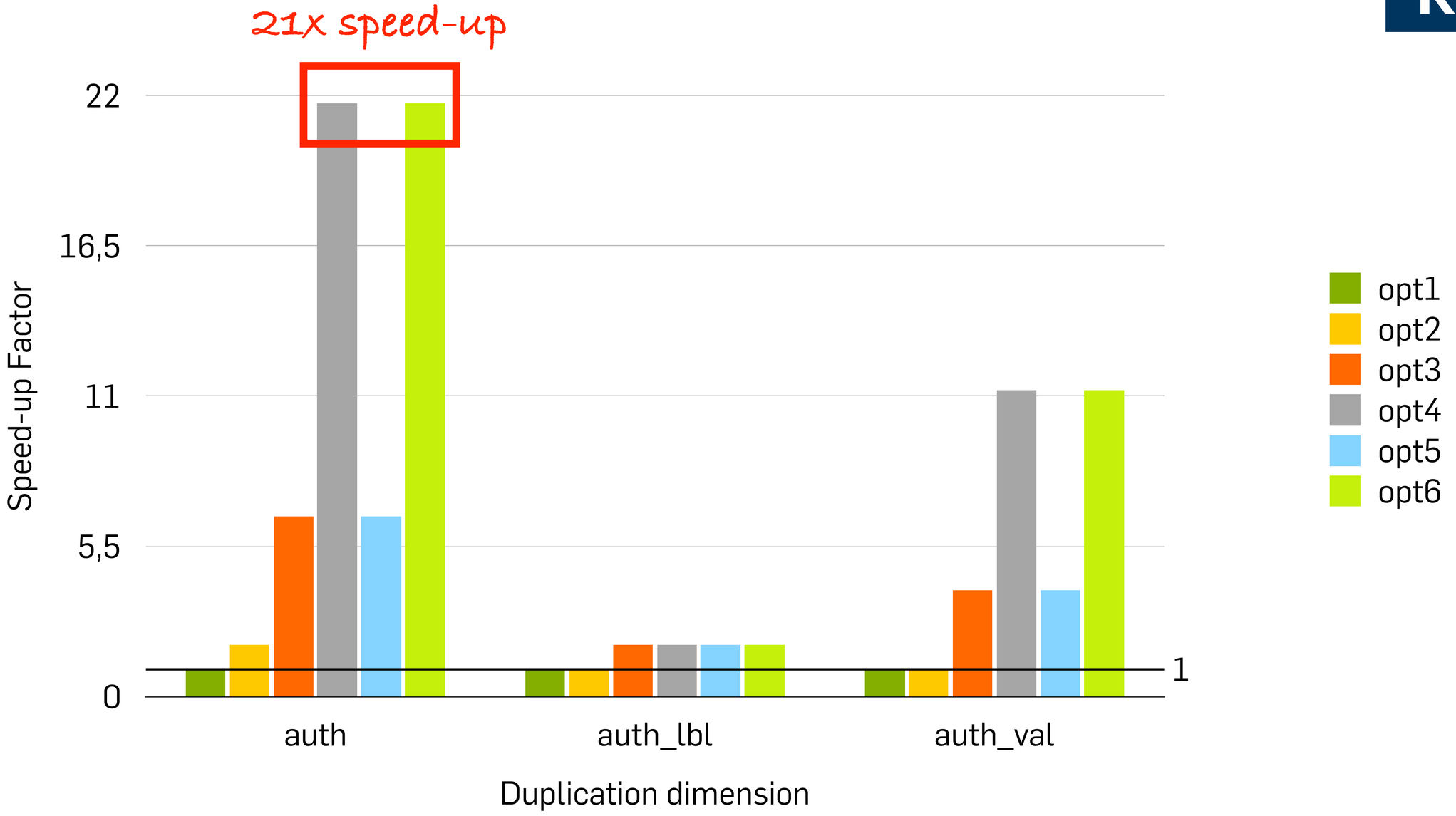
Phase 2



Decryption, policy length 60, positive literals

Evaluation

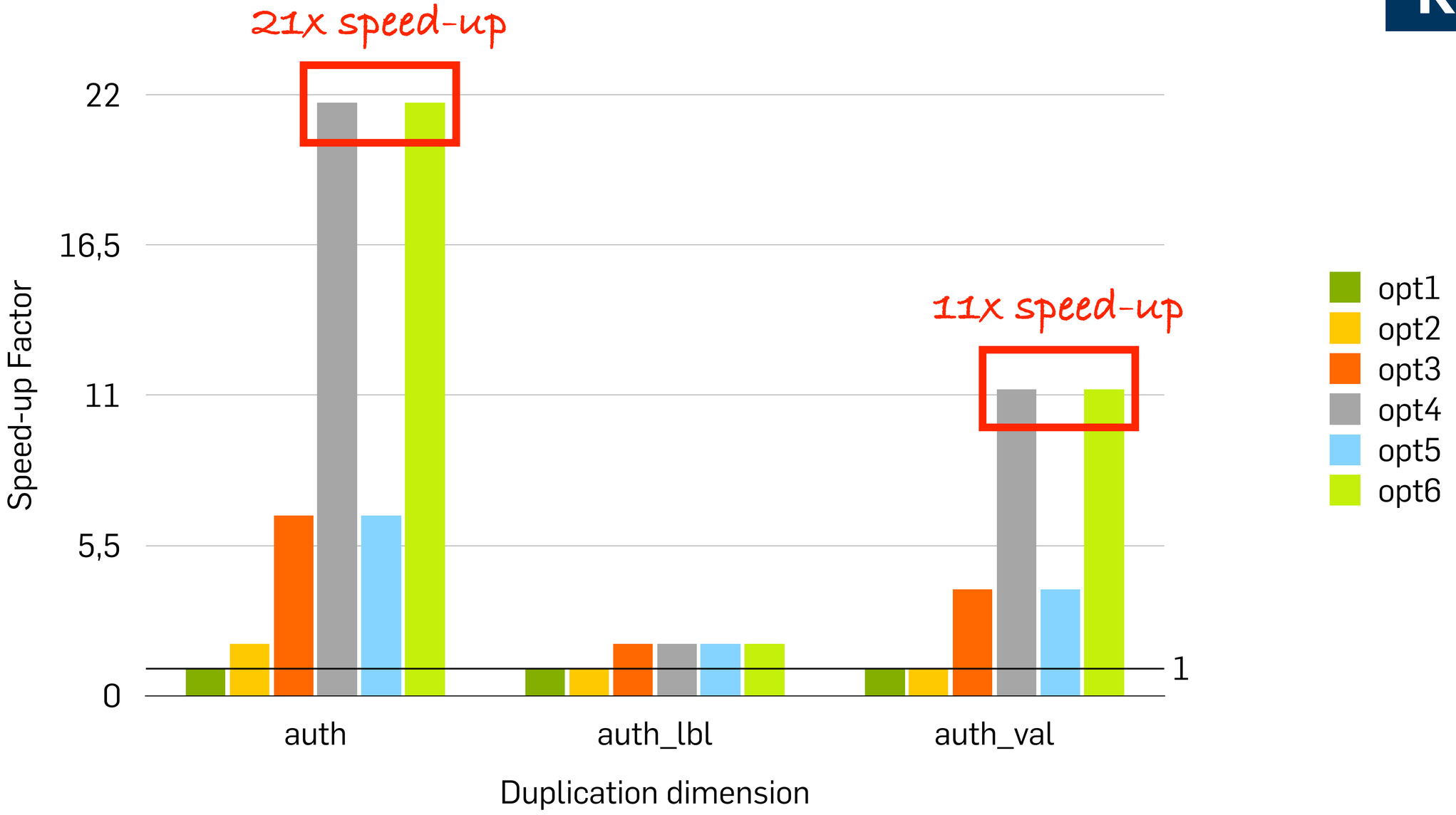
Phase 2



Decryption, policy length 60, positive literals

Evaluation

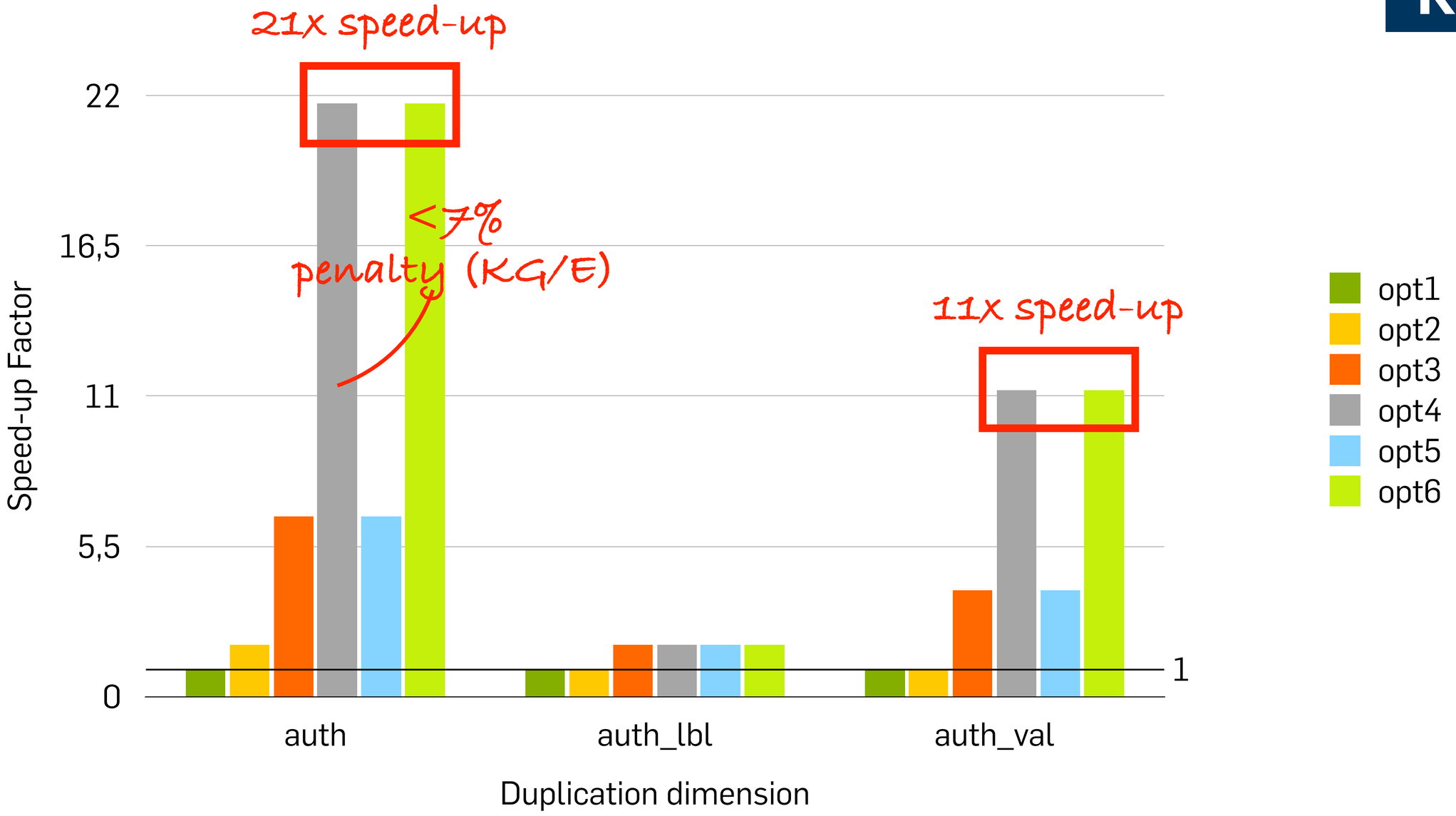
Phase 2



Decryption, policy length 60, positive literals

Evaluation

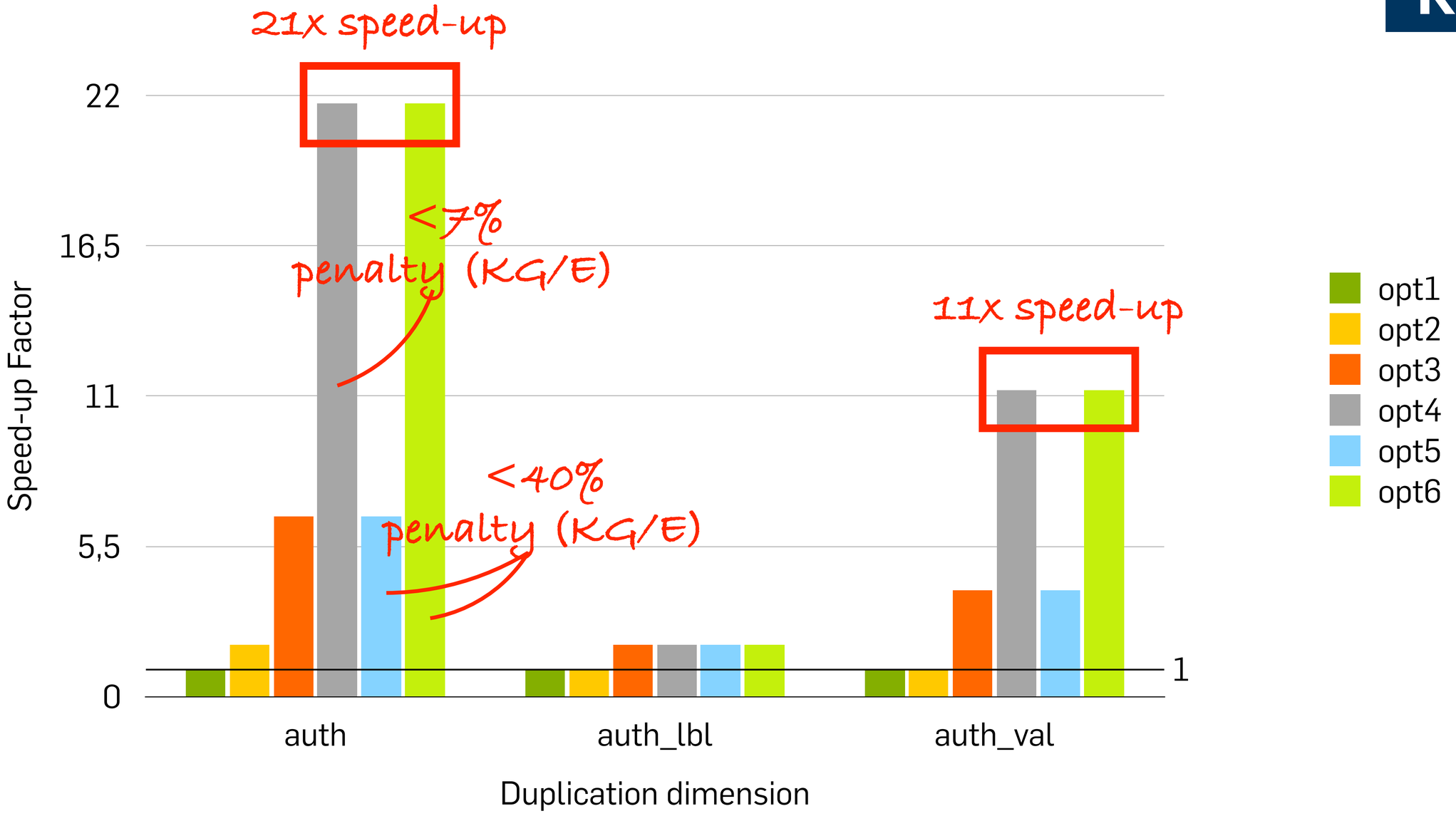
Phase 2



Decryption, policy length 60, positive literals

Evaluation

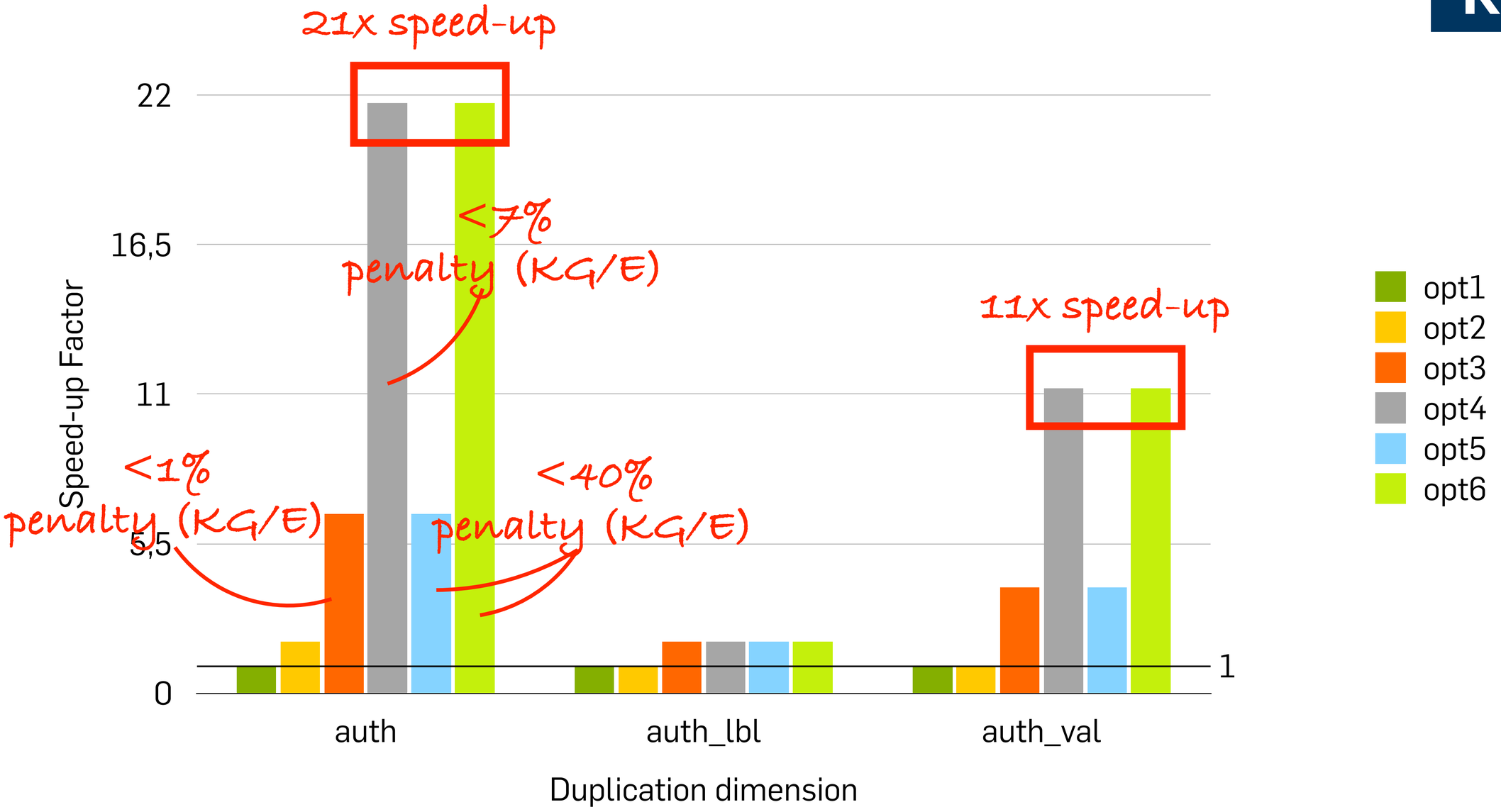
Phase 2



Decryption, policy length 60, positive literals

Evaluation

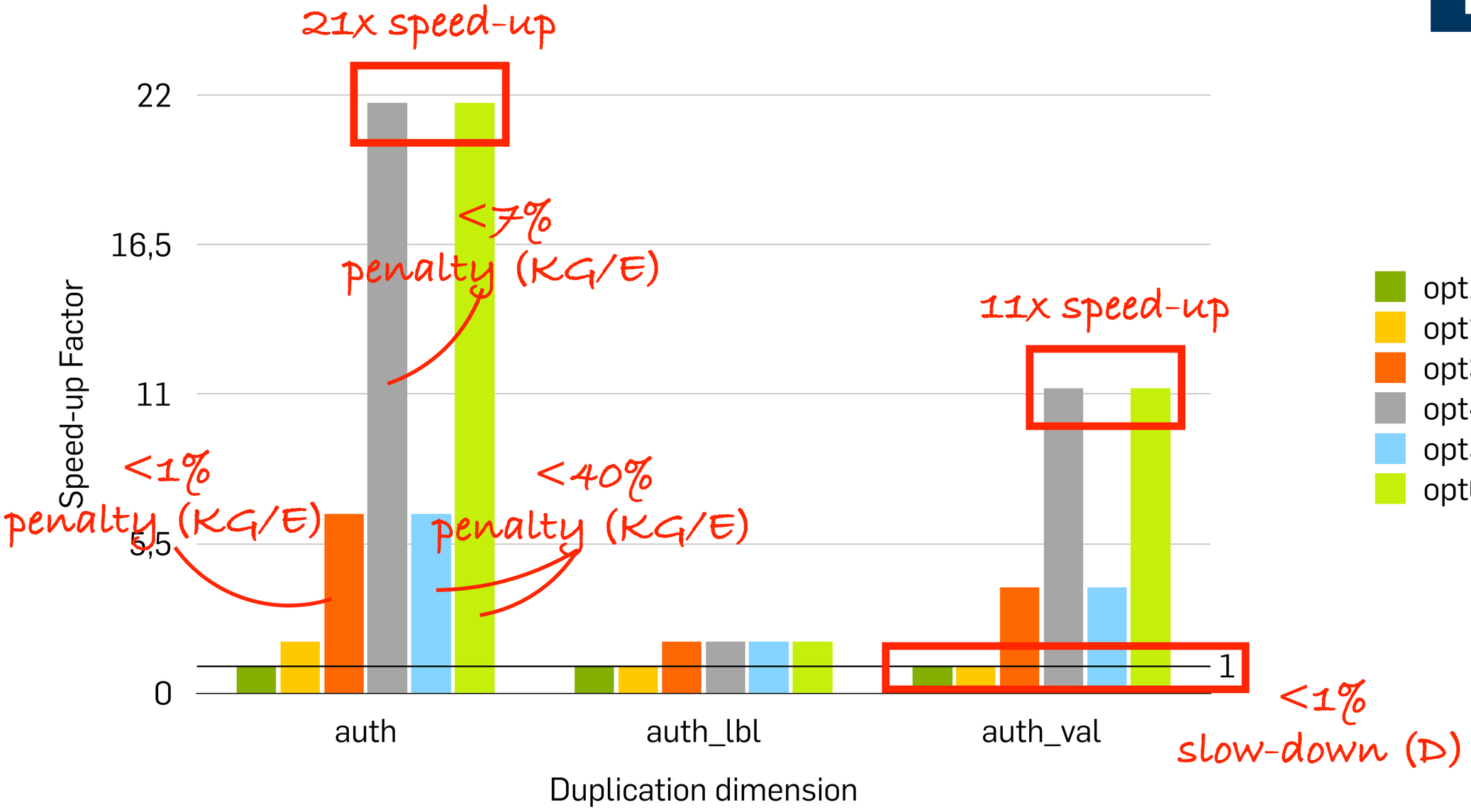
Phase 2



Decryption, policy length 60, positive literals

Evaluation

Phase 2

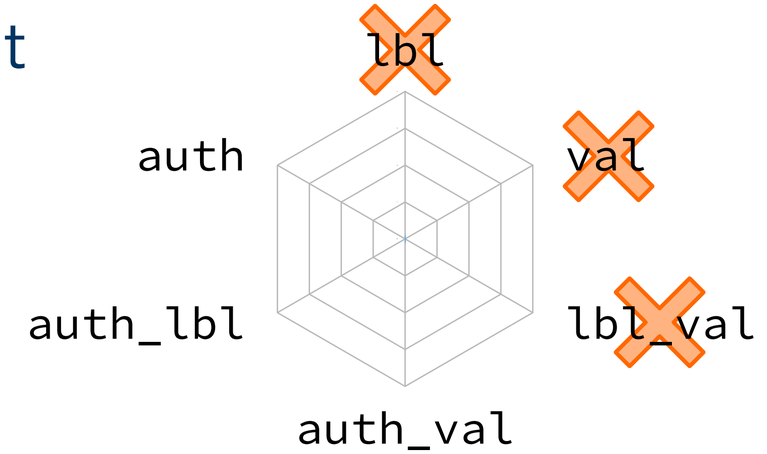


Decryption, policy length 60, positive literals

Evaluation

Summary

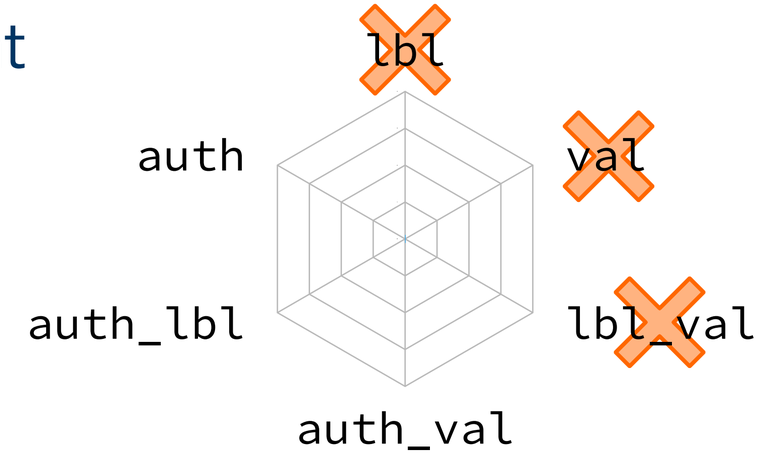
- Only dimension `auth`, `auth_lbl` and `auth_val` relevant
- Positive literals
  - *opt3*: up to **6x** speed-up with <1% penalty on KG/E
  - *opt4*: up to **21x** speed-up with <7% penalty on KG/E
- Negative literals
  - *opt5*/*opt6* have some undesirable trade-offs



Evaluation

Summary

- Only dimension `auth`, `auth_lbl` and `auth_val` relevant
- Positive literals
  - *opt3*: up to **6x** speed-up with <1% penalty on KG/E
  - *opt4*: up to **21x** speed-up with <7% penalty on KG/E
- Negative literals
  - *opt5*/*opt6* have some undesirable trade-offs

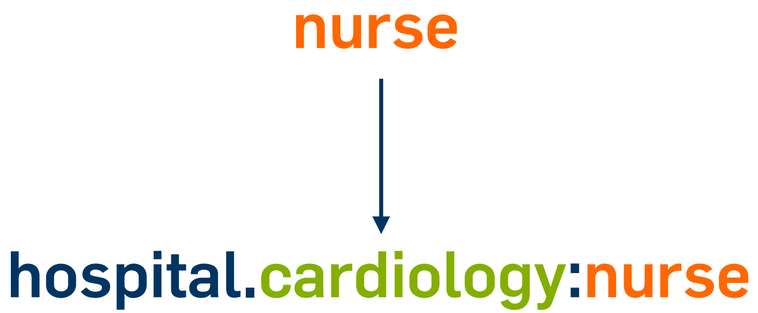


Conclusion

Conclusion

Key Take-Aways

- Expressive MA-CP-ABE scheme
  - **New optimizations**: fewer randomizers ⇒ fewer pairings
- **New benchmarking strategy**
  - Six duplication dimensions (+ negations)
- **Full implementation** and evaluation in Rust
- Up to **21x speed-up** with minimal overhead
  - Negations dampen speed-ups ⇒ future work



Full Paper



Thank you!

Sven Argo

Chair for Security Engineering
Faculty of Computer Science
Ruhr University Bochum