Secure Distributed Data and Event Processing at Scale

Where are we now?

P. Eugster USI Lugano

Jun 30 DEBS'23

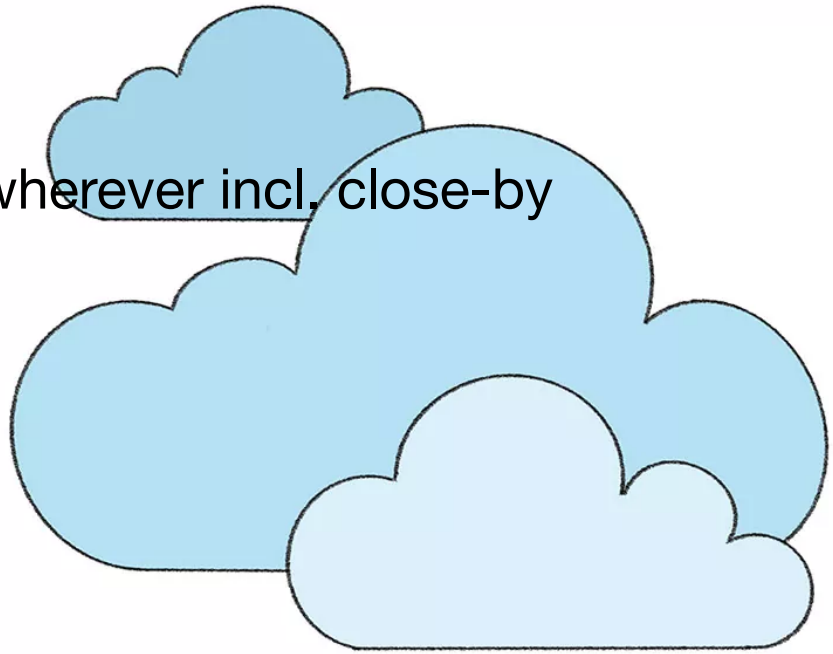
Roadmap

- Motivation: distributed data and event processing
- Problem: untrusted third-party infrastructure
- Solution fragments: security mechanisms
- Requirements: what do we need
- Tradeoffs: what comes first
- HYDRA: a carefully balanced approach
- Conclusions: lessons learned and open challenges

Motivation

Distributed data and event processing

- Rise of “remote execution”
  - Advent of cloud computing, edge
  - Massive resources available on demand, wherever incl. close-by
  - “Distribution by commodity”
- Further digitization of society
  - IoT, wearable computing, …
  - “Distribution by necessity”



Problem

Untrusted third-party infrastructure

- So you want to do distributed processing on large data sets?
  - Do not have your own cluster?
  - Just use cloud or edge data centers, or both!
  - And say goodbye to your data…
- Causes
  - Multitenancy, internal threats, big government
  - Use of “foreign” software, tool sprawl

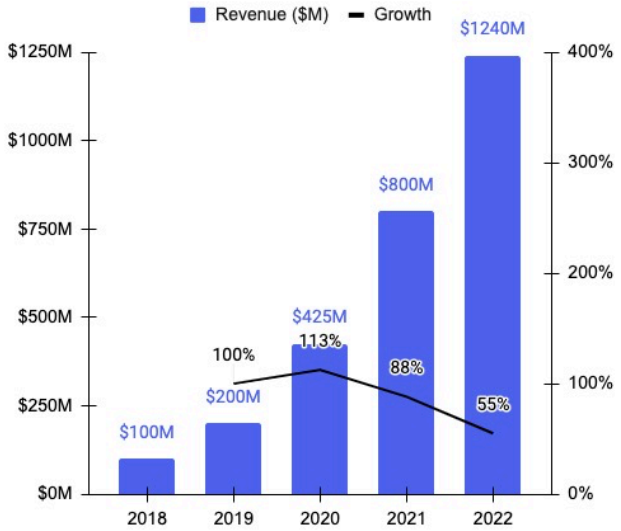
[IBM’22] https://www.ibm.com/reports/data-breach

- Almost half of data breaches happen in cloud, with average cost of $4.35M!!! [IBM’22]

Data Analytics and Confidential Computing

Relevance for society and economy

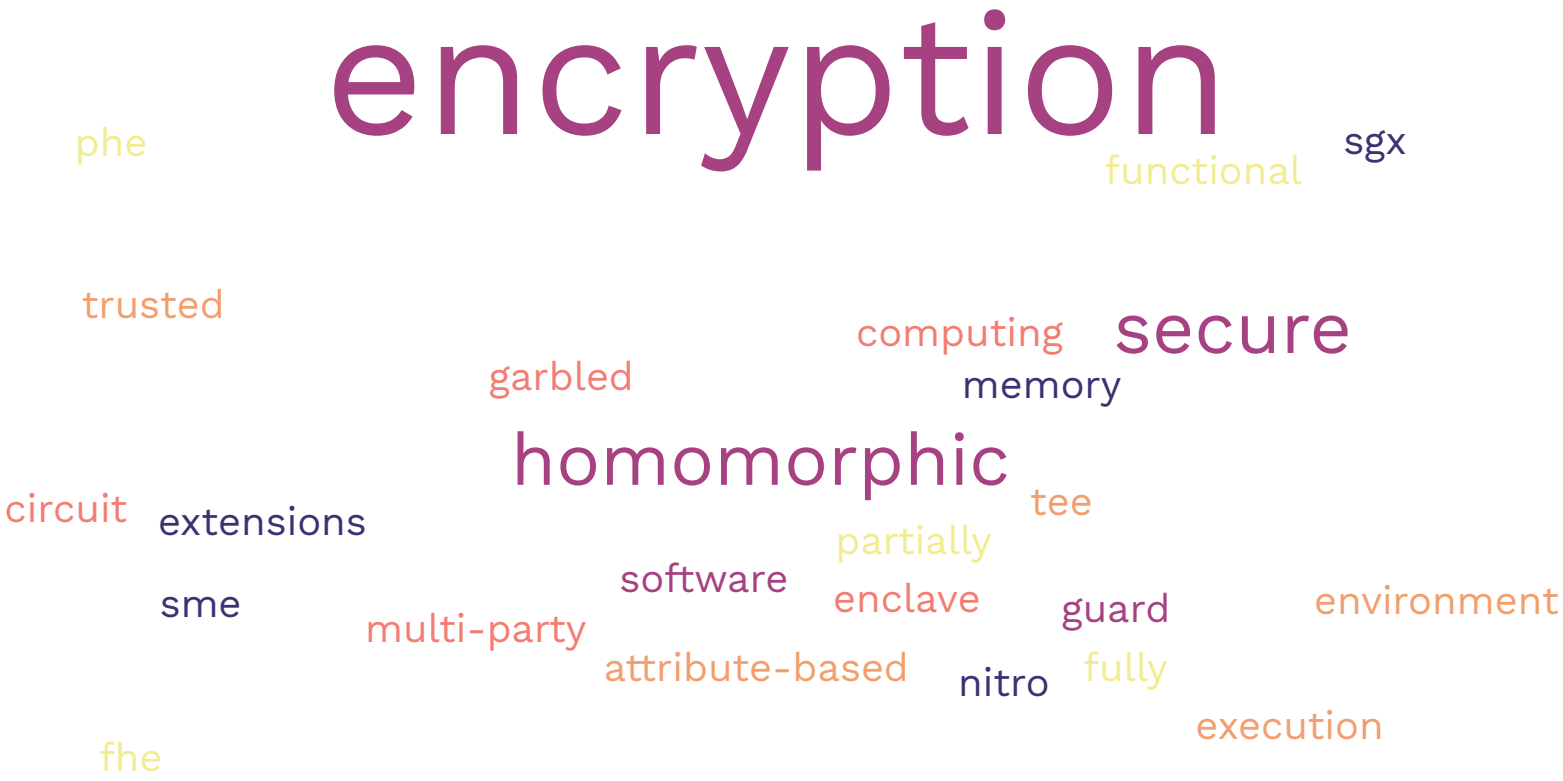
- 2022 revenue of DataBricks, a leader in data analytics, estimated at $1.24B up from $800M in 2021, 90% from platform based on Spark [Sabra'23].
- DataBricks/Spark lacks support for security **required** by many scenarios.
- Gartner predicts size of **confidential computing** market to be $54B by 2026 [Gartner'23].



[Sabra'23] https://sacra.com/c/databricks/

[Gartner'23] https://www.gartner.com/en/newsroom/press-releases/2021-03-23-gartner-identifies-top-security-and-risk-management-t

Solution Fragments



Mechanisms

Software (SW)

- Cryptographic primitives, e.g.,
  - Fully homomorphic encryption [Gentry'10]
  - Partially homomorphic encryption (PHE)
  - Functional encryption [Boneh et al.'11]
  - Attributed-based encryption [Shamir'84]
  - Garbled circuits [Yao'86]
  - …

[Gentry'10] C. Gentry. Computing Arbitrary Functions of Encrypted Data. Communications of the ACM 53(3): 97-105, 2010.

[Boneh et al.'11] D. Boneh, A. Sahai, and B. Waters: Functional Encryption: Definition and Challenges. TCC 2011.

[Shamir'84] A. Shamir: Identity-Based Cryptosystems and Signature Schemes. Advances in Cryptology 1984.

[Yao'86] A. C.-C. Yao: How to Generate and Exchange Secrets. FOCS 1986.

Mechanisms

Hardware (HW)

- Trusted execution environments, e.g.,
  - Intel Secure Guard eXtensions (SGX)
  - AMD Secure Memory Encryption (SME)
  - ARM Trustzone
  - Amazon Nitro
  - …

Mechanisms

Hardware (HW)

- Trusted execution environments, e.g.,
  - Intel Secure Guard eXtensions (SGX)
  - AMD Secure Memory Encryption (SME)
  - ARM Trustzone
  - Amazon Nitro
  - …

Which one(s) to use? How?

Requirements?

More/higher/stronger is better

- Transparency
  - Very few data analysts are experts in security
- Efficiency
  - Many potential users deterred by high overhead w.r.t. security-agnostic execution
  - 10x slowdown critical
- Portability
  - Lock-in with cloud or platform providers is never good, may support different (HW) mechanisms
  - New mechanisms keep being proposed

Requirements (cont'd)?
Higher/stronger is better

- Interoperability
  - Important especially if combining cloud and edge in *data pipelines*, notably continuous processing
  - Interoperability ⇏ portability: if components can interact across platforms doesn't mean they can move
  - For simplicity we consider interoperability as platform-independence including portability
- Security
  - What is the attacker/threat model and guarantees given in the face of those? If little security is offered …
  - How strongly are they substantiated? Informal, "formal" on paper, mechanically verified, …?
- Expressivity
  - Supporting simpler computations is always easier

Tradeoffs

- Transparency vs interoperability: user-facing language?
  - SecureScala [Hauck et al.'16]
- Transparency vs efficiency: transparency for whom?
  - Cuttlefish [Savvides et al.'17]
- Security vs efficiency: what say you (user)?
  - Symmetria [Savvides et al.'20]
- Expressivity vs efficiency: continuous is always harder
  - STYX [Stephen et al.'16], C3PO [Savvides et al.'22]

[Hauck et al.'16] M. Hauck, S. Savvides, P. Eugster, M. Mezini, and G. Salvaneschi: SecureScala: Scala Embedding of Securecomputations. SCALA 2016.

[Savvides et al.'17] S. Savvides, J. J. Stephen, M. Saeida Ardekani, V. Sundaram, and P. Eugster: Secure Data Types: A Simple Abstraction for Confidentiality-Preserving Data Analytics. SoCC 2017.

[Savvides et al.'20] S. Savvides, D. Khandelwal, and P. Eugster: Efficient Confidentiality-Preserving Data Analytics over Symmetrically Encrypted Datasets. VLDB 2020.

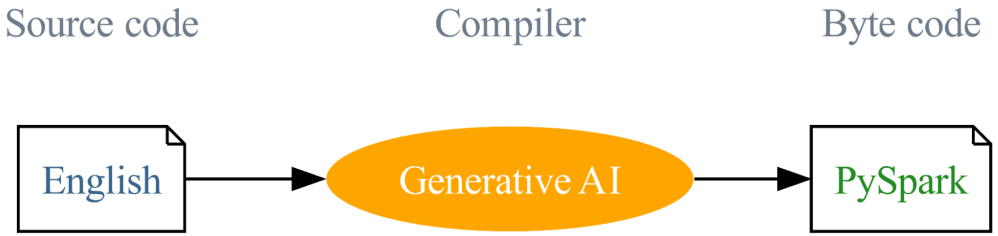
[Stephen et al.'16] J. J. Stephen, S. Savvides, V. Sundaram, M. Saeida Ardekani, and P. Eugster: STYX: Stream Processing with Trustworthy Cloud-based Execution. SoCC 2016.

[Savvides et al.'22] S. Savvides, S. Kumar, J. J. Stephen, and P. Eugster: C3PO. Cloud-based Confidentiality-preserving Continuous Query Processing. ACM TOPS 25(1), 2022.

Tradeoffs

- Transparency vs interoperability: user-facing language?
  - SecureScala [Hauck et al.'16]
- Transparency vs efficiency: transparency for
  - Cuttlefish [Savvides et al.'17]
- Security vs efficiency: what say you (user)?
  - Symmetria [Savvides et al.'20]
- Expressivity vs efficiency: continuous is always harder
  - STYX [Stephen et al.'16], C3PO [Savvides et al.'22]

Databricks introducing English as the new programming language for Apache Spark



[Stephen et al.'16] J. J. Stephen, S. Savvides, V. Sundaram, M. Saeida Ardekani, and P. Eugster: STYX: Stream Processing with Trustworthy Cloud-based Execution. SoCC 2016.

[Savvides et al.'22] S. Savvides, S. Kumar, J. J. Stephen, and P. Eugster: C3PO. Cloud-based Confidentiality-preserving Continuous Query Processing. ACM TOPS 25(1), 2022.

Tradeoffs

- Transparency vs interoperability: user-facing language?
  - SecureScala [Hauck et al.'16]
- Transparency vs efficiency: transparency for whom?
  - Cuttlefish [Savvides et al.'17]
- Security vs efficiency: what say you (user)?
  - Symmetria [Savvides et al.'20]
- Expressivity vs efficiency: continuous is always harder
  - STYX [Stephen et al.'16], C3PO [Savvides et al.'22]

[Hauck et al.'16] M. Hauck, S. Savvides, P. Eugster, M. Mezini, and G. Salvaneschi: SecureScala: Scala Embedding of Securecomputations. SCALA 2016.

[Savvides et al.'17] S. Savvides, J. J. Stephen, M. Saeida Ardekani, V. Sundaram, and P. Eugster: Secure Data Types: A Simple Abstraction for Confidentiality-Preserving Data Analytics. SoCC 2017.

[Savvides et al.'20] S. Savvides, D. Khandelwal, and P. Eugster: Efficient Confidentiality-Preserving Data Analytics over Symmetrically Encrypted Datasets. VLDB 2020.

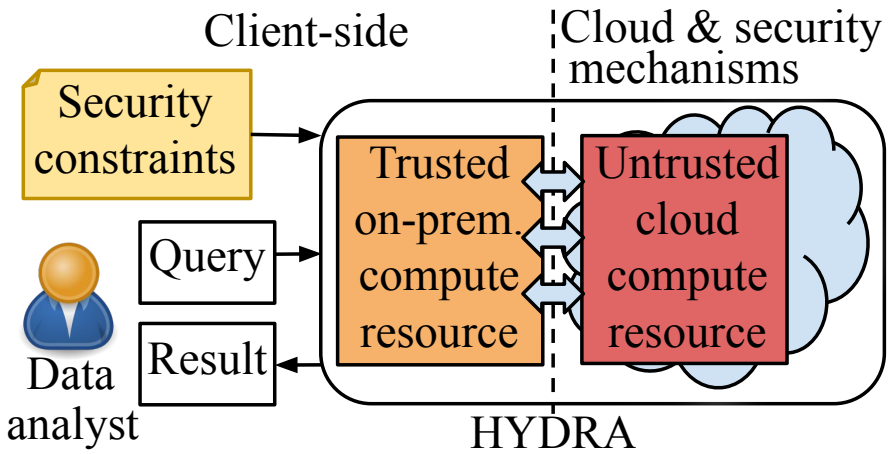
[Stephen et al.'16] J. J. Stephen, S. Savvides, V. Sundaram, M. Saeida Ardekani, and P. Eugster: STYX: Stream Processing with Trustworthy Cloud-based Execution. SoCC 2016.

[Savvides et al.'22] S. Savvides, S. Kumar, J. J. Stephen, and P. Eugster: C3PO. Cloud-based Confidentiality-preserving Continuous Query Processing. ACM TOPS 25(1), 2022.

HYDRA

Hybrid approach to confidentiality-preserving data analytics

- Transparency: data analysts write queries *agnostically to security constraints*.
- Efficiency: *over 10x faster* than state of the art [Mangipudi et al.'23].
- Interoperability: *combining* software mechanisms (homomorphic encryption schemes, e.g., Paillier [Paillier'99]) and hardware mechanisms (trusted execution environments, e.g., Intel SGX) from *extensible sets.*
- Security: *secure end-to-end* with *formally verified* guarantees.
- (Expressivity: focus on distributed data analytics)



[Paillier'99] P. Paillier: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. EUROCRYPT 1999.

[Mangipudi et al.'23] S. Mangipudi, P. Chuprikov, P. Eugster, M. Viering and S. Savvides: Generalized Policy-based Non-interference for Efficient Confidentiality-Preservation. PLDI 2023.

Transparency

- Full transparency is unrealistic, undesired
- **Can define/split different roles (SecureScala, Cuttlefish)**
  1. *Data analysts*: know queries; but not computer scientists, not expert programmers
     - Should be able to write queries agnostically to security
  2. *Security experts*: know security mechanisms, attacks, cloud platforms, etc.
     - Define (static) security constraints and policies
  3. *Data managers*: know data semantics, understand security basics
     - Define (static) security requirements for data together with 2.

Efficiency

- SW can outperform HW (Cuttlefish)
  - E.g., PHE > SGX sometimes
  - **Want SW *and* HW**
- Differences between SW (Symmetria)
- Researchers frantically working on new SW solutions
- Researchers and engineers frantically working on new HW solutions
- Want to be able to integrate *new* solutions without complete redesign

Operation	PHE scheme	Plaintext	PHE	SGX
filter (_ = _)	AES-ECB [Daemen and Rijmen 2002]	5.4 s	5.8 s	8.0 s
filter range	OPE [Boldyreva et al. 2009]	5.7 s	8.7 s	10.7 s
filter match	SWP [Song et al. 2000]	5.4 s	8.8 s	9.4 s
groupby	AES-ECB [Daemen and Rijmen 2002]	6.2 s	7.2 s	57.7 s
sort	OPE [Boldyreva et al. 2009]	7.2 s	13.0 s	41.0 s
select (_ + _)	Paillier [Paillier 1999]	5.2 s	19.4 s	7.8 s
select (_ × _)	ElGamal [ElGamal 1985]	5.1 s	22.4 s	7.8 s

Interoperability

- **Support *extensible* sets of HW and SW mechanisms**
  - Domains $\mathscr{D}$: capture HW mechanisms and infrastructure providers
    - Can treat same HW mechanism differently in different infrastructures
  - Crypto(graphic) schemes $\mathscr{S}$: capture SW mechanisms
- Why not just 1 dimension/set of mechanisms?
  - Isn't SGX just AES(-NI) in HW?
    - Different characteristics and use
  - Benefits of combining HW with SW mechanisms for *same* data

Security

- Extensible sets $\mathscr{D}$ and $\mathscr{S}$
- How to integrate mechanisms with different guarantees?
  - E.g., different threat models, known attacks [Fei et al.'21]
- Basis: honest-but-curious (HbC) attacker model
  - Focus on confidentiality
- **Let security experts define what they trust and where (Cuttlefish)**
- **Leverage formal techniques to verify guarantees end-to-end (SecureScala)**

[Fei et al.'21] S. Fei, Z. Yan, W. Ding, and H. Xie: Security Vulnerabilities of SGX and Countermeasures: A Survey. ACM CSUR 54(6), 2021.

Security Policy $\mathbb{S}$

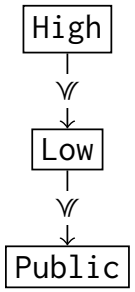
- Custom finite set of labels/levels $\mathscr{L}$ with lowest element $\bot$ [Denning'76]
- Arranged according to lattice $\langle \mathscr{L}, \preccurlyeq \sqcup , \sqcap \rangle$
  - Partial order $\preccurlyeq$ reflexive antisymmetric binary relation on $\mathscr{L}$
    - $l_1 \preccurlyeq l_2$: confidentiality requirements imposed by l_2 are at least as strict as l_1; always safe to replace l_1 with l_2 when labelling data
  - Least upper bound operator $\sqcup$: $\mathscr{L} \times \mathscr{L} \rightarrow \mathscr{L}$
  - Greatest lower bound operator $\sqcap$: $\mathscr{L} \times \mathscr{L} \rightarrow \mathscr{L}$
- Security policy $\mathbb{S}$ *assigning* levels to combinations of SW and HW mechanisms $\mathbb{S} : \mathscr{L} \rightarrow 2^{\mathscr{D} \times \mathscr{S}_{\varnothing}}$

[Denning'76] D. E. Denning: A Lattice Model of Secure Information Flow. Communications of the ACM, 1976.

Example Security Policy Setup

All steps performed once only

1. Define secrecy levels



Cf. US DoD

2. Assign mechanisms to levels

Label $\mathcal{L}$	Domain $\mathcal{D}$	Scheme $\mathcal{S}_\varnothing$
High	CLNT, SGX	$\varnothing$
High	CLD	AES-GCM
Low	CLD	SWP, AES-ECB, Paillier, ElGamal, OPE

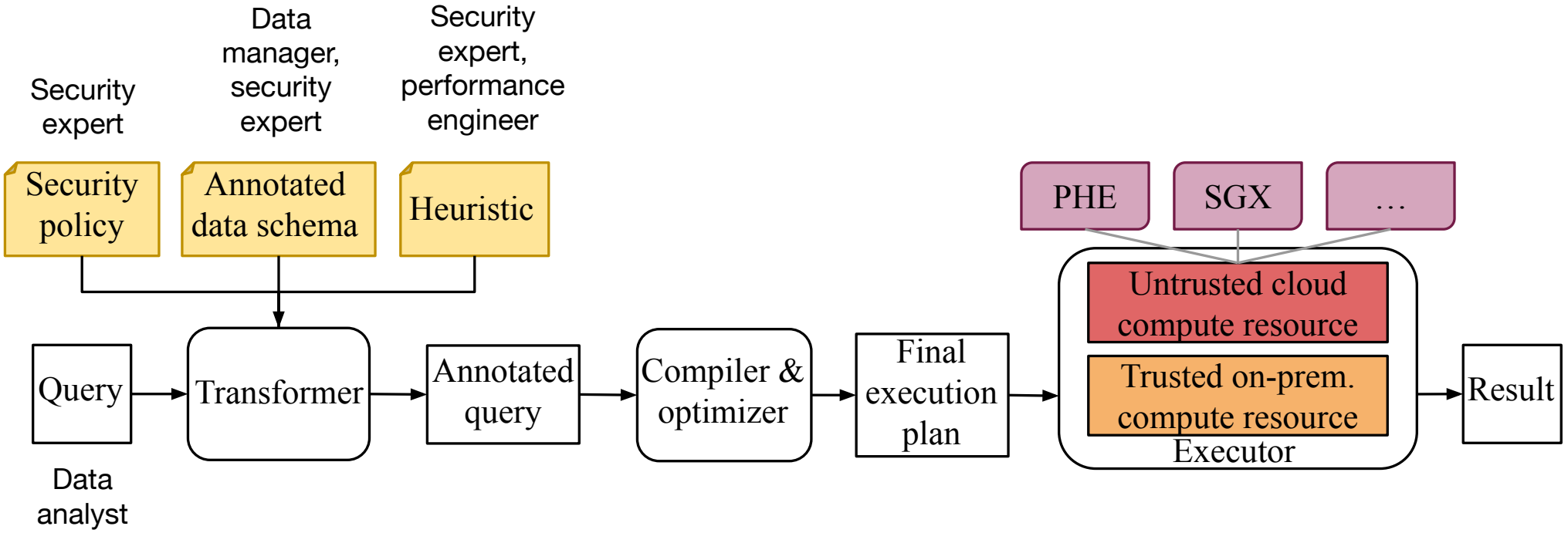
E.g., data at level `High` can be handled only on the client side (`CLNT`), by Intel SGX (`SGX`), or in the cloud (no SGX) only when using AES-GCM encryption

3. Label data with levels

Relation	Field	Type	Label	Security type (inferred)
Customers	custId	Str	Low	(Str$^{\text{AES-ECB}}$,Low)
	bal	Dbl	High	(Dbl$^{\text{AES-GCM}}$,High)
	…			
Orders	orderId	Str	High	(Str$^{\text{AES-GCM}}$,High)
	custKey	Str	Low	(Str$^{\text{AES-ECB}}$,Low)
	price	Str	High	(Dbl$^{\text{AES-GCM}}$,High)
	date	Int	Low	(Int$^{\text{OPE}}$,Low)
	…			

Security types automatically inferred at query transformation, e.g. AES-GCM encryption for `price` tags in `Orders`

Complete Workflow



Core Language

Lambda with relations and relational operators

$$
\begin{array}{rrcl}
\text{Type} & \kappa & ::= & \overline{\kappa} \rightarrow_{d} \kappa \mid \{\overline{f : (p^{s}, l)}\} \mid T\{\overline{f : (p^{s}, l)}\} \mid (p^{s}, l) \\
\text{Prim type} & p & ::= & \texttt{Int} \mid \texttt{Dbl} \mid \texttt{Str} \mid \texttt{Bool} \mid \ldots \\
\text{Scheme} & s & ::= & \varnothing \mid \texttt{AES-GCM} \mid \texttt{ElGamal} \mid \texttt{Paillier} \mid \ldots \quad \text{(in one-to-one correspondence with } \mathcal{S}_{\varnothing}) \\
\text{Domain} & d & ::= & \texttt{CLNT} \mid \texttt{SGX} \mid \texttt{SEV} \mid \texttt{CLD} \mid \ldots \quad \text{(in one-to-one correspondence with } \mathcal{D}) \\
\text{Value} & v & ::= & T\{\overline{f : v}\} \mid \{\overline{f : v}\} \mid c^{s} \mid \lambda[d](\overline{x : \kappa}).\, e \mid f \\
\text{Expression} & e & ::= & v \mid x \mid e(\overline{e}) \mid \oplus(\overline{e}) \mid \{\overline{f : e}\} \mid e.f \mid \texttt{table}(name) \mid \theta(\overline{e}) \mid \texttt{encr}(e, s) \mid \texttt{decr}(e) \mid [e]_{d} \\
\text{Prim ops} & \oplus & ::= & + \mid - \mid \ldots \mid \wedge \mid \vee \mid \ldots \\
\text{Query ops} & \theta & ::= & \texttt{filter} \mid \texttt{proj} \mid \texttt{cross} \mid \texttt{agg} \mid \ldots
\end{array}
$$

- Blue parts not used by data analysts
- Green parts only used during evaluation

Transformation

1. Original security-agnostic query

```
1 agg(filter(cross(table(Customers),
2                  filter(table(Orders),
3                         λ(rO: {/* Orders */}). rO.date < 16052002)),
4            λ(rCO: {/* Customers + Orders */}). rCO.custId == rCO.custKey),
5     custId, 0,
6     λ(rP: {price: Dbl}, acc: Int).
7         acc + rP.price)
```

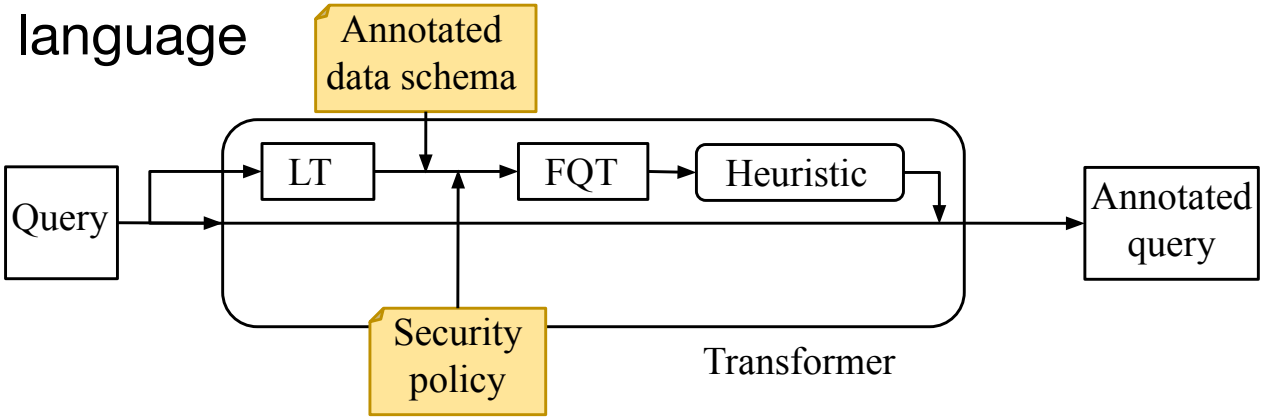
Transformation

2. Automatically transformed query with security annotations

```
1 agg(filter(cross(table(Customers),
2                  filter(table(Orders),
3                         λ[SGX](rO: {/* Orders */}). rO.date < 0x..OPE)),
4            λ[SGX](rCO: {/* Customers + Orders */}). rCO.custId == rCO.custKey),
5     custId, 0x..AES-GCM,
6     λ[SGX](rP: {price: (DblAES-GCM, High)}, acc: (DblAES-GCM, High)).
7            encr(decr(acc) + decr(rP.price), AES-GCM))
```

Transformation Heuristics and Framework

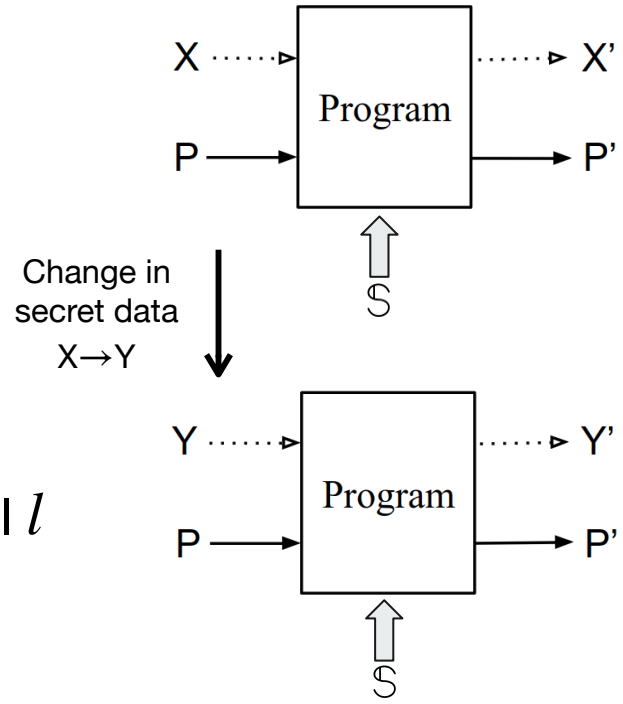
- Formal framework denoting well-formed transformations
  - (Partially) enforced through APIs and libraries for writing heuristics
- Information flow type system verifies transformed queries
  - Also queries written in full language



Guarantees

Informal

- Based on *noninterference* [Goguen&Meseguer'82]
  - Generalized to our security policy, thus $\mathbb{S}$-noninterference
- Partially ordered set of adversaries $\langle \mathscr{A}, \subseteq \rangle$
  - $A \in \mathscr{A}$ downward-closet set $A \subseteq \mathscr{D} \times \mathscr{S}_{\varnothing}$ it can break
  - If $\mathbb{S}(l) \cap A = \varnothing$ then A is unable to see any inputs at level l
- Adversary who can't break anything at level l can't see difference in outputs of computation if inputs at level l or higher are changed



[Goguen&Meseguer'82] A. Goguen and J. Meseguer: Security Policies and Security Models. S&P 1982.

Guarantees
Formal

(Ω table store, ρ plaintext schema, κ type w/ annotations)

Definition 4 (Level-l $\mathbb{S}$-noninterference $\mathbb{S}$-NI$(e)_{\rho,d,l}$). *Expression e has $\mathbb{S}$-NI$(e)_{\rho,d,l}$ property dubbed* level-l $\mathbb{S}$-noninterference *if and only if for any two stores Ω_1 and Ω_2 satisfying ρ, $\Omega_1 \sim^{l}_{\rho} \Omega_2$, and any two values v_1 and v_2, $e \xrightarrow[\Omega_1]{}{}^{*} v_1$ and $e \xrightarrow[\Omega_2]{}{}^{*} v_2$, it holds that $v_1 \sim^{d,l} v_2$.*

Definition 5 ($\mathbb{S}$-noninterference $\mathbb{S}$-NI$(e)_{\rho,d}$). *Expression e has* $\mathbb{S}$-noninterference property *$\mathbb{S}$-NI$(e)_{\rho,d}$ if and only if it has level l $\mathbb{S}$-noninterference property $\mathbb{S}$-NI$(e)_{\rho,d,l}$ for every l in $\mathcal{L}$.*

Theorem 1 (Soundness). *If there exists non-function κ, s.t., $\rho \vdash_d e : \kappa$ w.r.t. $\mathbb{S}$ then $\mathbb{S}$-NI$(e)_{\rho,d}$.*

Theorem 2 (Subject Reduction). *Let $\Vdash_{/d} \Omega : \rho$, $\rho \wr \Gamma \Vdash_d \mathbb{e} : \kappa$ and $\mathbb{e} \underset{\Omega}{\Longrightarrow} \mathbb{e}'$ then $\rho \wr \Gamma \Vdash_d \mathbb{e}' : \kappa$.*

Guarantees

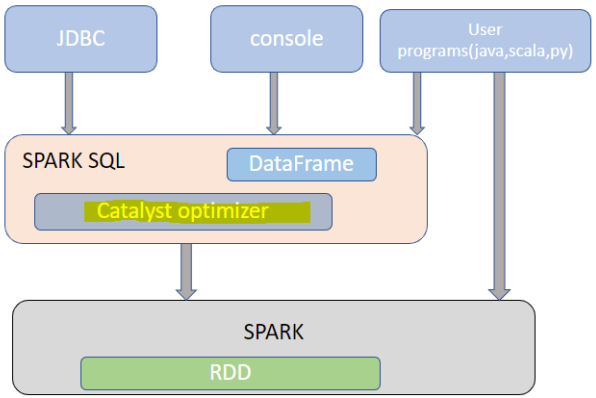
Transformation

Definition 1 (query transformation). *Function* $\tau[\![\cdot,\cdot]\!]$ *is a* query transformation *iff* $\forall \rho, e,$ *and* $(\rho', e') = \tau[\![\rho, e]\!]$, *we have* $e \rightsquigarrow e'$ *and* $\rho \rightsquigarrow_S \rho'$.

Theorem 3 (Transformation correctness). *For query transformation* $\tau[\![\cdot,\cdot]\!]$, *schemata* ρ, *and expression* e, *let* $(\rho', e') = \tau[\![\rho, e]\!]$. *If* $\rho' \vdash_d e' : \kappa$ *for some domain* d *and type* κ, *and also* $e \underset{\Omega}{\longrightarrow}^* v$ *for some table store* Ω *satisfying* ρ, *then for any* $\Omega' = \mathsf{encrVal}(\Omega, \rho')$, *there exists* v', *s.t.,* $e' \underset{\Omega'}{\longrightarrow}^* v'$ *and* $\mathsf{decrVal}(v') = v$.

Implementation

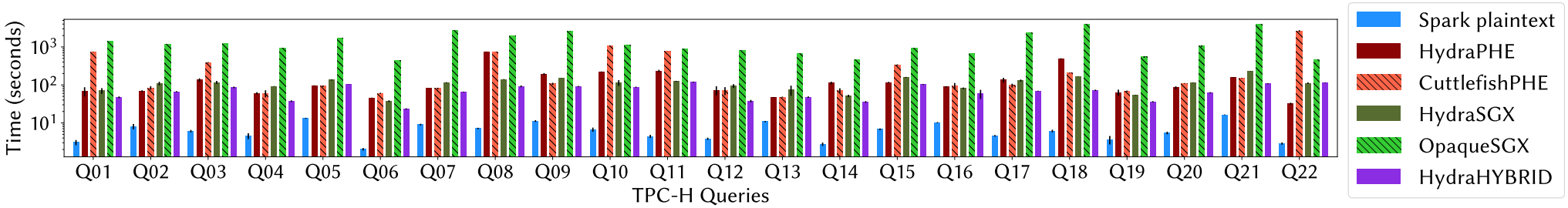
- Built on Apache Spark
  - Used by data analysts through original APIs (SparkSQL)
- Leveraging *Catalyst* extensible query optimizer (query analyzer, optimizer, execution planner)
  - Input data encrypted in analysis phase (3.2 kLoC Scala)
  - Logical optimization in analysis phase (3.4 kLoC Scala)
  - Physical opt. in planning phase (6.2 kLoC Scala and 100 LoC Java)
  - Code generation step (29 kLoC C++, 2.5 kLoC C, 568 LoC Scala)
- SGX used via JNI (mini-interpreter implemented in C)
- Simple HYDRA Hybrid PHE&SGX heuristic: PHE until hitting limit of PHE or operation faster in SGX



Evaluation

Execution time

- TPC-H benchmarks, average of 5 runs, labeling from [Savvides et al.'17]



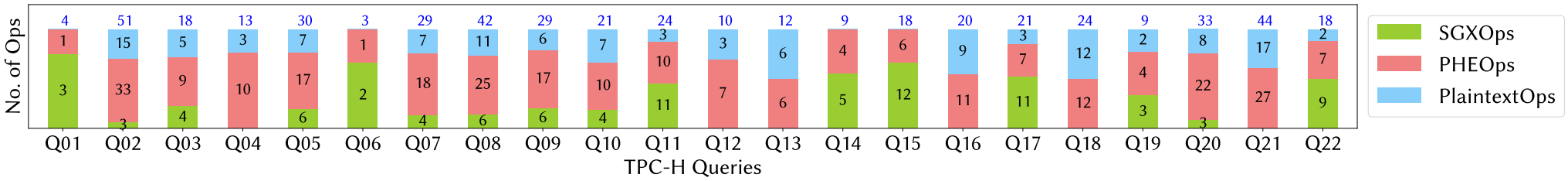
- HYDRA (PHE) on average 1.6x faster than Cuttlefish
- HYDRA (SGX) on average 11.3x faster than Opaque [Zheng et al.'17] (ORAM disabled)
- HYDRA (Hybrid) 1.7x and 1.6x times faster than HYDRA PHE and HYDRA SGX respectively
  - 2.7x and 17.9x faster than Cuttlefish and Opaque respectively

[Zheng et al.'17] W. Zheng, A. Dave, J. Beekman, R. A. Popa, J. E. Gonzalez, and I. Stoica: Opaque: An Oblivious and Encrypted Distributed Analytics Platform. NSDI'17.

Evaluation

Transparency

- What if a programmer wanted to use/combine mechanisms manually?



- Proxy for effort: #ops and distribution for HYDRA Hybrid in TPC-H
  - Large number of operators gives idea of all possible combinations!
- 16 of 22 queries use mix of PHE and SGX
- Note: 18 of 22 queries use plaintext operations

Extensibility

1. Adding/changing data relations (schema changes): (re-)label following $\mathscr{L}$ (data manager w/ security expert)
2. Modifying security policy $\mathbb{S}$
   A. To assign mechanisms differently to levels: change only $\mathbb{S}(l)$ (security expert)
   B. To add/remove levels: change $\langle \mathscr{L}, \preccurlyeq \sqcup , \sqcap \rangle$ and $\mathbb{S}(l)$ (security expert)
      i. Possibly revisit data labeling (data manager w/ security expert, cf. 1.)
3. Adding whole new mechanism to $\mathscr{D}$ or $\mathscr{S}$

- Some glue code
- Augment heuristic (security expert w/ performance engineer)
- Anybody wanting to use either 2.A. or 2.B.

Ongoing/Future Work and Open Challenges
HYDRA

- Placement
- Heuristic framework
  - More advanced heuristics
  - Improved verification of properties
  - Tracing back compilation/execution errors
- Security
  - Stronger guarantees, e.g., intermediate results, relation size leakage
  - Precision of guarantees, cf. universal composability [Canetti'00]
  - Beyond HbC: integrity

[Canetti'00] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. https://eprint.iacr.org/2000/067.

Ongoing/Future Work and Open Challenges
Beyond HYDRA

- Multi-party computing (MPC)
- Differential privacy (DP) [Dwork et al.'17]
- Continuous processing
  - Spark allows incremental computing, micro-batching
  - Main efforts in general still around data stores, e.g., *data clean rooms*
  - Vs MPC and/or DP [Pettai&Laud'15][Cao et al.'19]

[Dwork et al.'17] C. Dwork, F. McSherry, K. Nissim, and A. Smith: Calibrating Noise to Sensitivity in Private Data Analysis. Journal of Privacy and Confidentiality 7(3) 2017.

[Pettai&Laud'15] M. Pettai and P. Laud: Combining Differential Privacy and Secure Multiparty Computation. ACSAC 2015.

[Cao et al.'19] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong: Quantifying Differential Privacy in Continuous Data Release Under Temporal Correlations. IEEE TKDE 31(7) 2019.

Corporate and Open Community Initiatives

- Fortanix: specific products and generic platform, PHE & SGX ($130M raised)
- Opaque Systems: *data clean rooms* based on Opaque ($32M raised)
- Decentrique: *data clean rooms*, SGX ($26M raised)
- Decision3: chain(?) and SGX & AWS ($100K raised)
- *<Your company here>*
- Veracruz: backed by Linux Foundation's Confidential Compute Consortium (C3)
- Apache Teaclave: generic secure computing with SGX & Trustzone
- …
- General lack: continuous processing support, security guarantees that are verified ("securified")

[Crunchbase]

Thank You