New Directions in Secure Multi-party Computation: Techniques and Information Disclosure Analysis

Alessandro Baccarini, PhD

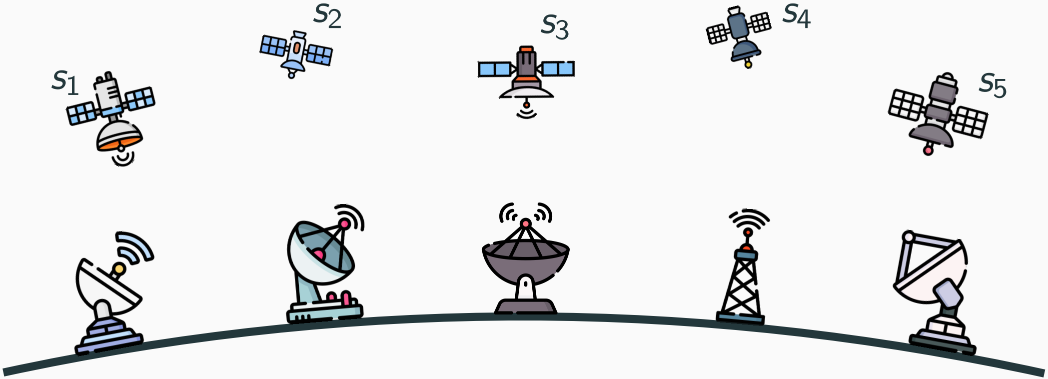
abaccarini@proton.me
abaccarini.github.io

February 5, 2025

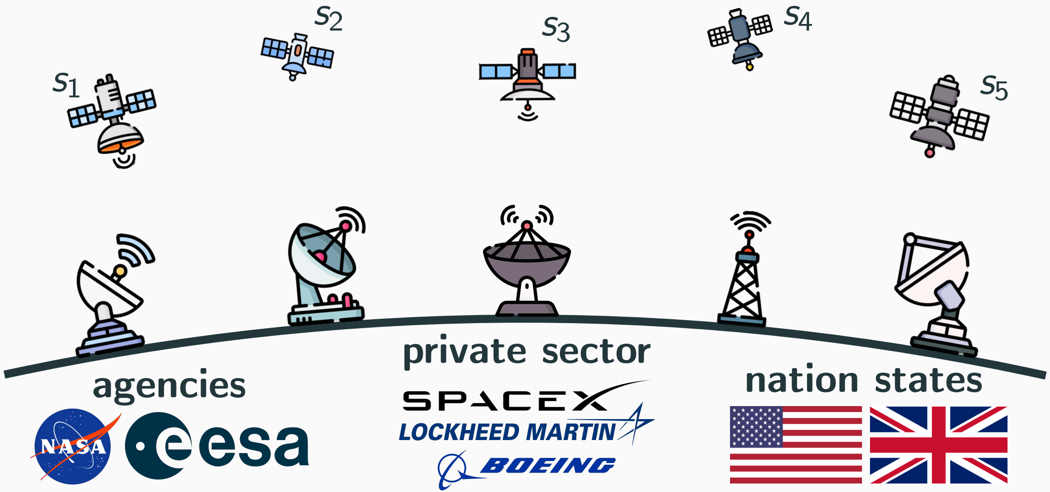
Outline

Motivation

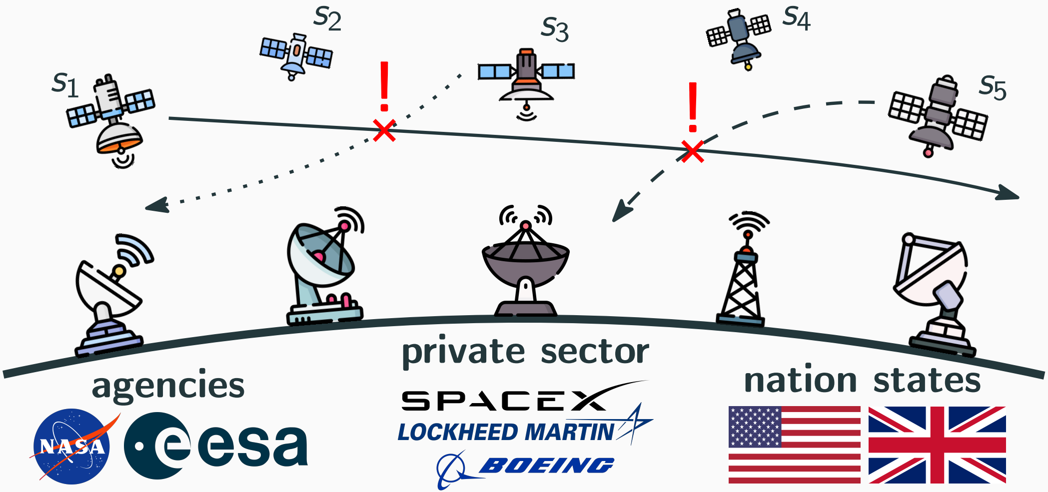
Motivational example: satellite mechanics



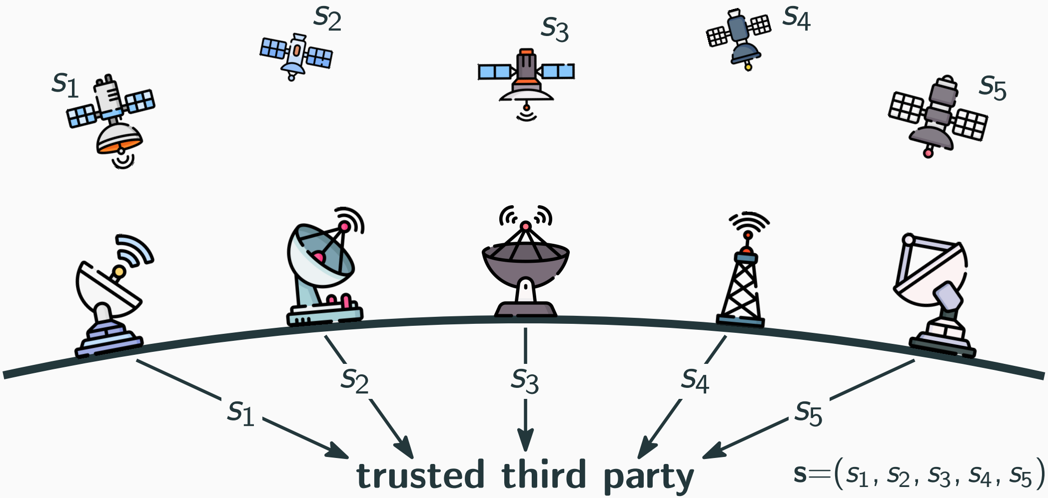
Motivational example: satellite mechanics



Motivational example: satellite mechanics

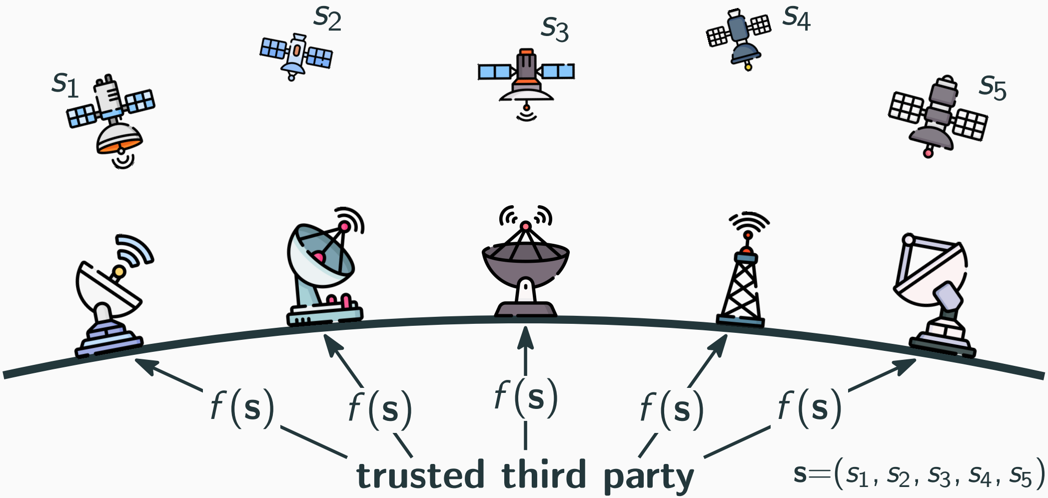


Motivational example: satellite mechanics

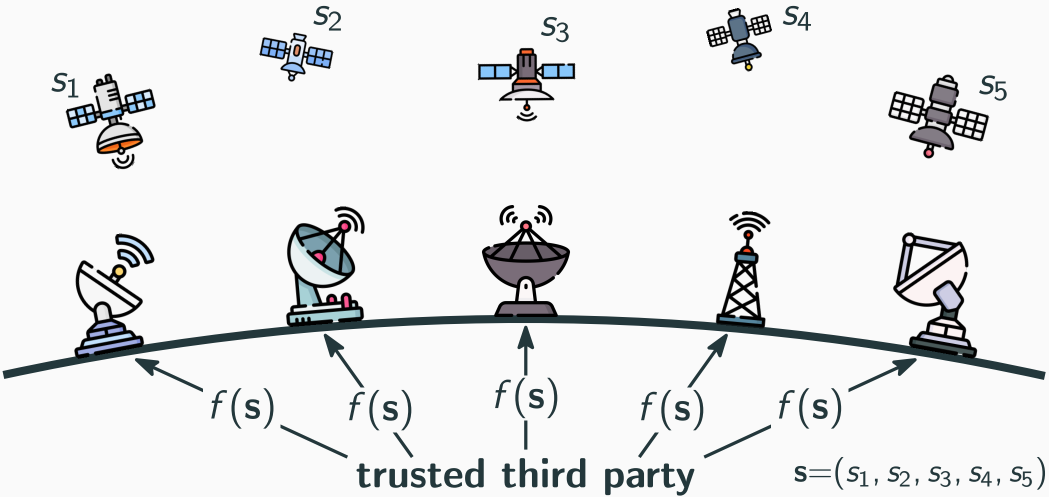


$\mathbf{s} = (s_1, s_2, s_3, s_4, s_5)$

Motivational example: satellite mechanics

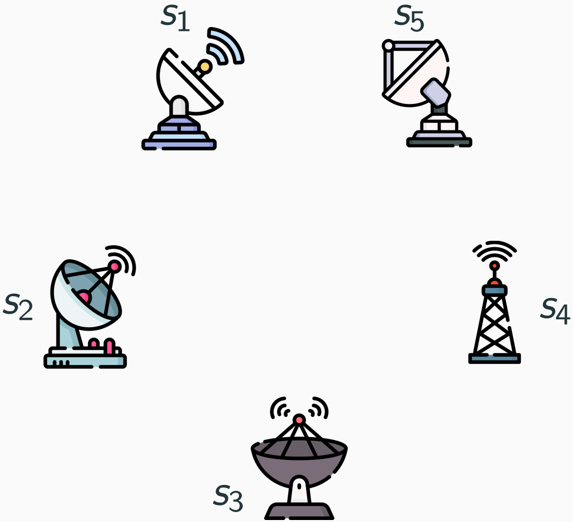


Motivational example: satellite mechanics



– How can we *privately* compute $f(\mathbf{s})$, **without a trusted third party**?

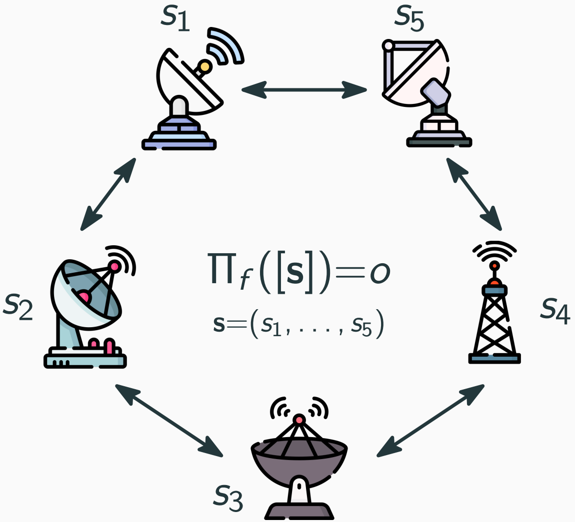
Enter (secure) multi-party computation



Multi-party computation (MPC)

Multiple participants **jointly** evaluating an **arbitrary** function on private inputs.

Enter (secure) multi-party computation

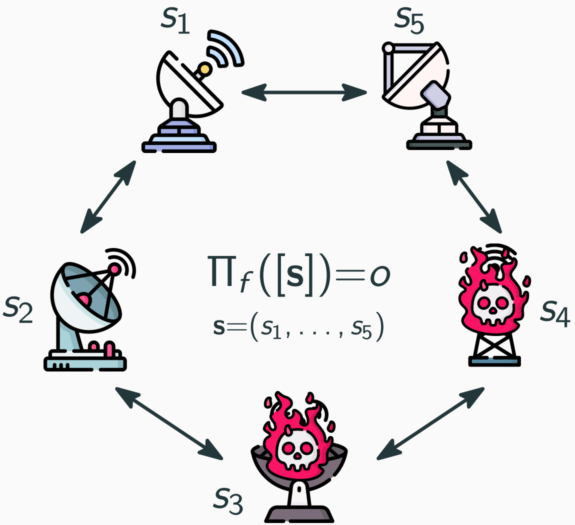


Multi-party computation (MPC)

Multiple participants **jointly** evaluating an **arbitrary** function on private inputs.

- FHE, garbled circuits, **secret sharing**

Enter (secure) multi-party computation



Multi-party computation (MPC)

Multiple participants **jointly** evaluating an **arbitrary** function on private inputs.

- FHE, garbled circuits, **secret sharing**
- (n, t)-threshold scheme
  - $\leq t$ **cannot** recover the secret
- **semi-honest (passive), honest majority**

Secret sharing (SS) techniques

Fields $\mathbb{F}_p$ (Shamir [Sha79])

Rings $\mathbb{Z}_{2^k}$ (Ito et al. [ISN87])

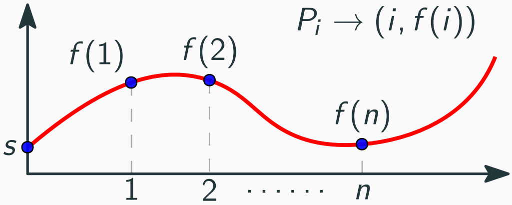
Secret sharing (SS) techniques

Fields $\mathbb{F}_p$ **(Shamir [Sha79])**

- Shares are points on a **polynomial**
- Reconstruction through **interpolation** (requires multiplicative inverses)
- Reliance on **large-number libraries**

$$f(x) = s + a_1 x + \cdots + a_t x^t \pmod{p}$$

$$P_i \rightarrow (i, f(i))$$



Rings $\mathbb{Z}_{2^k}$ **(Ito et al. [ISN87])**

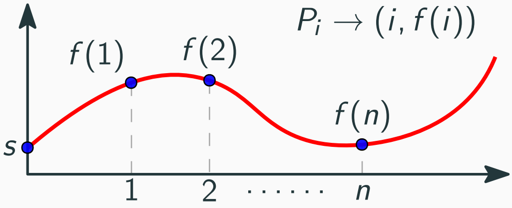
Secret sharing (SS) techniques

Fields $\mathbb{F}_p$ **(Shamir [Sha79])**

- Shares are points on a **polynomial**
- Reconstruction through **interpolation** (requires multiplicative inverses)
- Reliance on **large-number libraries**

$$f(x) = s + a_1 x + \cdots + a_t x^t \pmod{p}$$

$$P_i \to (i, f(i))$$



Rings $\mathbb{Z}_{2^k}$ **(Ito et al. [ISN87])**

- Each party maintains **replicated** shares
- Compatible with **native CPU instructions**
- **Limited to $n = 3, 4$ over integers**

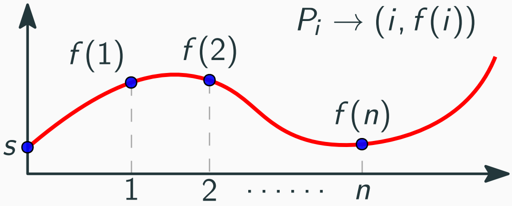
$$s = s_{\{1\}} + s_{\{2\}} + s_{\{3\}} \pmod{2^k}$$

Secret sharing (SS) techniques

Fields $\mathbb{F}_p$ **(Shamir [Sha79])**

- Shares are points on a **polynomial**
- Reconstruction through **interpolation** (requires multiplicative inverses)
- Reliance on **large-number libraries**

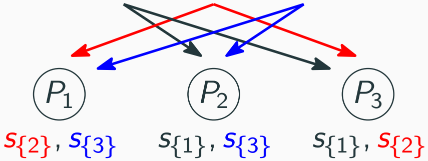
$$f(x) = s + a_1 x + \cdots + a_t x^t \pmod{p}$$



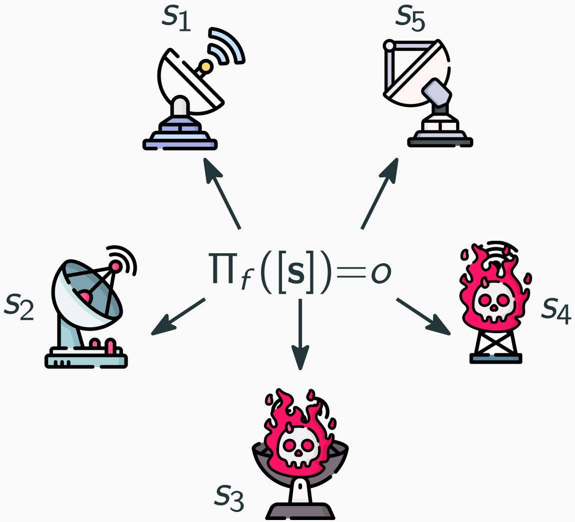
Rings $\mathbb{Z}_{2^k}$ **(Ito et al. [ISN87])**

- Each party maintains **replicated** shares
- Compatible with **native CPU instructions**
- **Limited to $n = 3, 4$ over integers**

$$s = s_{\{1\}} + s_{\{2\}} + s_{\{3\}} \pmod{2^k}$$

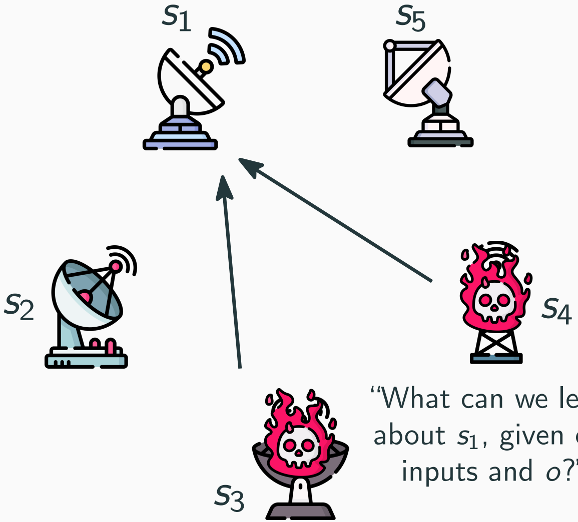


What do we *really* mean by “secure”?



- No information disclosed throughout computation, **other than the output**

What do we *really* mean by "secure"?



"What can we learn about s_1, given our inputs and o?"

- No information disclosed throughout computation, **other than the output**
- But does the **output itself** contain sensitive information?
- Can we **quantify** this disclosure in a meaningful way?

RSS framework for arbitrary *n*

- Develop a *comprehensive* suite of RSS protocols for any n to enable general-purpose computation on integers, and floating-point values
- Implement protocol constructions in an MPC compiler (PICCO) to enhance accessibility and usability

Information disclosure analysis

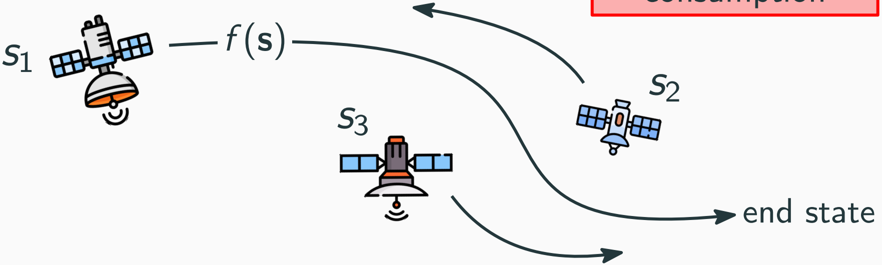
- Develop an information-theoretic approach to measure disclosure
- Apply technique to a practically significant function (the **average**)
- Extend analysis to complex statistical functions

General-purpose secure computation framework

Where to begin?

velocities, inertia, capabilities,...

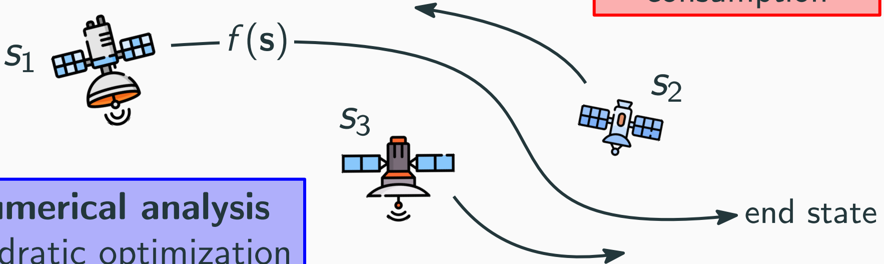
minimize resource consumption



Where to begin?

velocities, inertia, capabilities,. . .

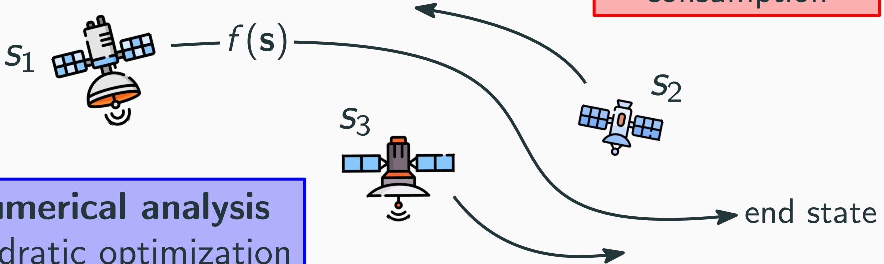
minimize resource consumption



numerical analysis
quadratic optimization
analytical functions
[Zap+18; Mun11; Vir+18]

velocities, inertia, capabilities, . . .

minimize resource consumption



numerical analysis
quadratic optimization
analytical functions
[Zap+18; Mun11; Vir+18]

matrix multiplications $(\mathbf{X} \cdot \mathbf{Y})$
comparisons $(x \overset{?}{>} y)$
approximations $(\tilde{f} \approx f)$

Towards general-purpose secure computation

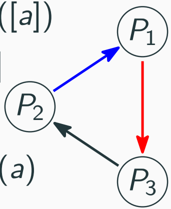
Building Blocks

reconstruction, mult.,
inputting private values

$\left.\begin{array}{c} c\cdot[a] \\ [a]+[b] \end{array}\right\}$ local, “free”

Open($[a]$)

$[a]\cdot[b]$



Input(a)

1 round,
$\mathcal{O}(t)$ comm.

Towards general-purpose secure computation

Building Blocks

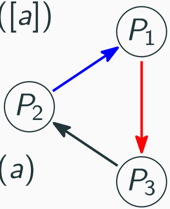
reconstruction, mult., inputting private values

$$\left.\begin{array}{r} c\cdot[a] \\ [a]+[b] \end{array}\right\}\ \text{local, “free”}$$

$\mathrm{Open}([a])$

$[a]\cdot[b]$

$\mathrm{Input}(a)$



Composite Operations

share conversion, shared randomness generation, comparisons, shifts, division

$$\mathbb{Z}_2 \longrightarrow \mathbb{Z}_{2^k} \quad [r_b] \in \{0, 1\}$$

$$([r] \in \mathbb{Z}_{2^k}, [r_i]_1 \in \mathbb{Z}_2)$$

$$[a] \overset{?}{<} [b] \quad [a] \overset{?}{=} [b]$$

$$[a/2^m], [a\cdot 2^m]$$

$$[a]/[b]$$

complexity

1 round, $\mathcal{O}(t)$ comm. $\longrightarrow$ Poly(log) (k, t) rounds/comm.

Towards general-purpose secure computation

Building Blocks

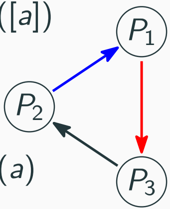
reconstruction, mult., inputting private values

$$\left.\begin{array}{r} c\cdot[a] \\ [a]+[b] \end{array}\right\}\ \text{local, “free”}$$

$\mathrm{Open}([a])$

$[a]\cdot[b]$

$\mathrm{Input}(a)$



Composite Operations

share conversion, shared randomness generation, comparisons, shifts, division

$\mathbb{Z}_2 \longrightarrow \mathbb{Z}_{2^k}\quad [r_b] \in \{0, 1\}$

$([r] \in \mathbb{Z}_{2^k}, [r_i]_1 \in \mathbb{Z}_2)$

$[a] \overset{?}{<} [b] \quad [a] \overset{?}{=} [b]$

$[a/2^m], [a\cdot 2^m]$

$[a]/[b]$

complexity

1 round, $\mathcal{O}(t)$ comm.	$\longrightarrow$	Poly(log) (k, t) rounds/comm.

integers (and fixed-point)

Towards general-purpose secure computation

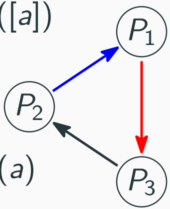
Building Blocks
reconstruction, mult., inputting private values

$\left.\begin{array}{r} c\cdot[a] \\ [a]+[b] \end{array}\right\}$ local, "free"

$\mathsf{Open}([a])$

$[a]\cdot[b]$

$\mathsf{Input}(a)$



1 round, $\mathcal{O}(t)$ comm. → Poly(log) (k, t) rounds/comm.

Composite Operations
share conversion, shared randomness generation, comparisons, shifts, division

$\mathbb{Z}_2 \longrightarrow \mathbb{Z}_{2^k} \quad [r_b] \in \{0,1\}$

$([r] \in \mathbb{Z}_{2^k}, [r_i]_1 \in \mathbb{Z}_2)$

$[a] \stackrel{?}{<} [b] \quad [a] \stackrel{?}{=} [b]$

$[a/2^m], [a{\cdot}2^m]$

$[a]/[b]$

complexity

Poly(log) (k, t) rounds/comm.

integers (and fixed-point)

Floating-point Computation
floating-point arithmetic, function approximation

?

reals

Towards general-purpose secure computation

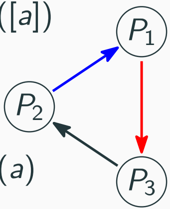
Building Blocks
reconstruction, mult., inputting private values

$\left.\begin{array}{l} c\cdot[a] \\ [a]+[b] \end{array}\right\}$ local, “free”

$\mathrm{Open}([a])$

$[a]\cdot[b]$

$\mathrm{Input}(a)$



Composite Operations
share conversion, shared randomness generation, comparisons, shifts, division

$\mathbb{Z}_2 \longrightarrow \mathbb{Z}_{2^k} \quad [r_b] \in \{0,1\}$

$([r] \in \mathbb{Z}_{2^k}, [r_i]_1 \in \mathbb{Z}_2)$

$[a] \overset{?}{<} [b] \quad [a] \overset{?}{=} [b]$

$[a/2^m], [a\cdot 2^m]$

$[a]/[b]$

complexity

Floating-point Computation
floating-point arithmetic, function approximation

$[\tilde{a}]<[\tilde{b}]$

$[\tilde{a}]\cdot[\tilde{b}] \quad [\tilde{a}]/[\tilde{b}]$

$[\tilde{a}]+[\tilde{b}]$

$f(x) \approx \begin{cases} \text{piecewise,} \\ \text{polynomial,} \\ \text{series, LUT,} \\ \dots \end{cases}$

1 round, $\mathcal{O}(t)$ comm. → Poly(log) (k, t) rounds/comm. → *many* rounds, expensive comm.

integers (and fixed-point) | **reals**

Towards general-purpose secure computation

Building Blocks

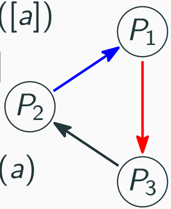
reconstruction, mult., inputting private values

$$\left.\begin{array}{r} c\cdot[a] \\ [a]+[b] \end{array}\right\}\text{local, “free”}$$

$\mathrm{Open}([a])$

$[a]\cdot[b]$

$\mathrm{Input}(a)$



1 round, $\mathcal{O}(t)$ comm.

Composite Operations

share conversion, shared randomness generation, comparisons, shifts, division

$\mathbb{Z}_2 \longrightarrow \mathbb{Z}_{2^k}$ $[r_b] \in \{0, 1\}$

$([r] \in \mathbb{Z}_{2^k}, [r_i]_1 \in \mathbb{Z}_2)$

$[a] \overset{?}{<} [b]$ $[a] \overset{?}{=} [b]$

$[a/2^m], [a\cdot 2^m]$

$[a]/[b]$

Poly(log) (k, t) rounds/comm.

complexity

Floating-point Computation

floating-point arithmetic, function approximation

$[\tilde{a}]<[\tilde{b}]$

$[\tilde{a}]\cdot[\tilde{b}]$ $[\tilde{a}]/[\tilde{b}]$

$[\tilde{a}]+[\tilde{b}]$

$$f(x) \approx \begin{cases} \text{piecewise,} \\ \text{polynomial,} \\ \text{series, LUT,} \\ \dots \end{cases}$$

many rounds, expensive comm.

true general-purpose computation

Lowering the barrier to entry

– How can we make MPC more **accessible** for end users?

Lowering the barrier to entry

- How can we make MPC more **accessible** for end users?
- MPC compilers: MP-SPDZ [Kel20], **PICCO** [ZSB13]
  - Extensive feature set (parallelization, pointers to private data, dynamic memory, ...)

```
public int main() {
    private int A, B, C;
    smcinput(A);
    smcinput(B);
    C = A * B;
    smcoutput(C);
}
```

User program (extended C)

Lowering the barrier to entry

- How can we make MPC more **accessible** for end users?
- MPC compilers: MP-SPDZ [Kel20], **PICCO** [ZSB13]
  - Extensive feature set (parallelization, pointers to private data, dynamic memory, ...)

```
public int main() {
    private int A, B, C;
    smcinput(A);
    smcinput(B);
    C = A * B;
    smcoutput(C);
}
```

User program (extended C)

compiles to $\longrightarrow$

```
int __original_main() {
    priv_int A, B, C;
    __s->smc_input(A);
    __s->smc_input(B);
    __s->smc_mult(A, B, C);
    __s->smc_output(C);
}
```

Translated program (C++), calls technique-specific protocols

- Uses **Shamir's secret sharing**

Lowering the barrier to entry

- How can we make MPC more **accessible** for end users?
- MPC compilers: MP-SPDZ [Kel20], **PICCO** [ZSB13]
  - Extensive feature set (parallelization, pointers to private data, dynamic memory, . . .)

```
public int main() {
    private int A, B, C;
    smcinput(A);
    smcinput(B);
    C = A * B;
    smcoutput(C);
}
```

User program (extended C)

compiles to →

```
int __original_main() {
    priv_int A, B, C;
    __s->smc_input(A);
    __s->smc_input(B);
    __s->smc_mult(A, B, C);
    __s->smc_output(C);
}
```

Translated program (C++), calls technique-specific protocols

- Uses **Shamir's secret sharing**
- **Integrated RSS protocols into PICCO**

Ongoing work

- Our n-party RSS framework serves as the **foundation** for a number of research directions

Protocols for nonlinear functions

[Ali+13; Rat+21; Rat+22]

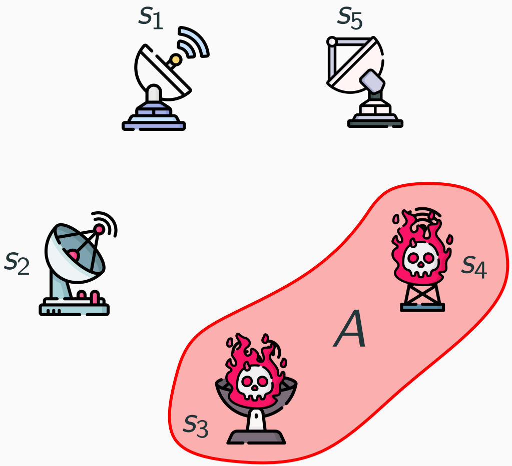
- $\log[\tilde{a}]$
- $\sqrt{[\tilde{a}]}$
- $2^{[\tilde{a}]}$
- $\exp([\tilde{a}])$

Interesting, practically significant applications of MPC

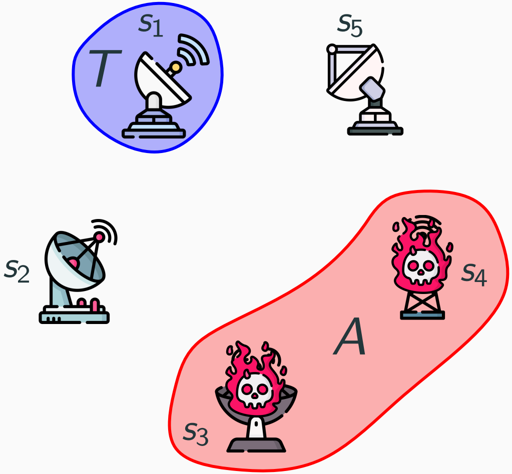
- Data streaming statistics, quantile queries
  - Hybrid RSS/DPF-based system [SVG24]

Information disclosure analysis

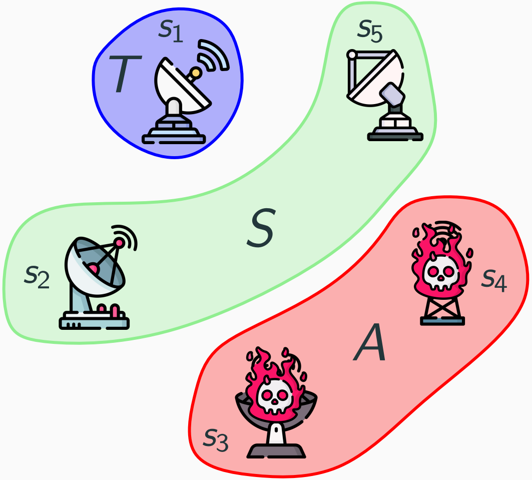
Setting and metric



- Partition into attackers A, targets T, and spectators S
- Model participants' inputs by **random variables** X_P

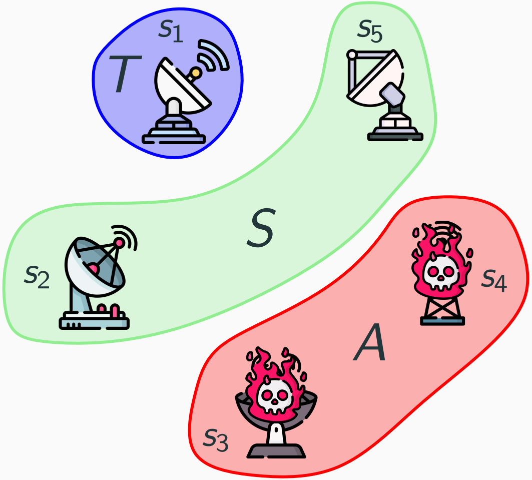


- Partition into attackers A, targets T, and spectators S
- Model participants' inputs by **random variables** X_P



- Partition into attackers A, targets T, and spectators S
- Model participants' inputs by **random variables** X_P
- How to **measure** the **information** disclosed by the output?

Setting and metric



- Partition into attackers A, targets T, and spectators S
- Model participants' inputs by **random variables** X_P
- How to **measure** the **information** disclosed by the output?

Entropy!

$$\underbrace{H(X)}_{\text{Shannon}} \qquad \underbrace{h(X)}_{\text{differential}}$$

Putting it together

- Attackers $\mathbf{X}_A$, targets $\mathbf{X}_T$, and spectators $\mathbf{X}_S$
- Treat the **output** as a random variable: $f(\mathbf{X}_A, \mathbf{X}_T, \mathbf{X}_S) = O$

Putting it together

- Attackers $\mathbf{X}_A$, targets $\mathbf{X}_T$, and spectators $\mathbf{X}_S$
- Treat the **output** as a random variable: $f(\mathbf{X}_A, \mathbf{X}_T, \mathbf{X}_S) = O$

Attackers' weighted average entropy **[AH17]**

$H(\mathbf{X}_T \mid \mathbf{X}_A = \mathbf{x}_A, O) \implies$ "how much information A learns about the target, given $\mathbf{x}_A$ and O"

Absolute entropy loss **[BBZ24a; BBZ24b]**

$H(\mathbf{X}_T) - H(\mathbf{X}_T \mid \mathbf{X}_A = \mathbf{x}_A, O) \implies$ "the total amount of information disclosed about the target, given $\mathbf{x}_A$ and O"

Case study: the average salary computation

- Analyzed the **average salary computation**, reduces to a **sum**:

$$f_\mu(\mathbf{x}) = \frac{1}{n}(x_1 + \cdots + x_n) \to x_1 + \cdots + x_n$$

Case study: the average salary computation

- Analyzed the **average salary computation**, reduces to a **sum**:

$$f_\mu(\mathbf{x}) = \frac{1}{n}(x_1 + \cdots + x_n) \to x_1 + \cdots + x_n$$

- Poisson, uniform, **Gaussian**, log-normal
- For a **single evaluation**, disclosure is **independent** of:
  - the attacker's input

$$H(\mathbf{X}_T \mid \mathbf{X}_A = \mathbf{x}_A, O) = H(\mathbf{X}_T \mid O)$$

  - the **distribution** and its parameters
- Much more analysis in the paper
  - ≥ 2 evaluations, min-entropy, mixed distribution parameters ...

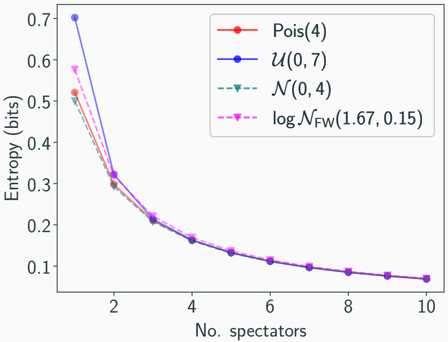


Figure 1: Absolute entropy loss (lower is better)

Next step: advanced statistical measures

- Prior analysis exploited properties of **sums of RVs**, leveraged **closed-form expressions** (of the entropy)

Next step: advanced statistical measures

- Prior analysis exploited properties of **sums of RVs**, leveraged **closed-form expressions** (of the entropy)
- What about **complex functions**?
  - Order statistics (max/min, median)
  - Variability measures (variance)
  - **Multidimensional outputs**
- Output could be **discrete**, while the inputs are **continuous**

Next step: advanced statistical measures

- Prior analysis exploited properties of **sums of RVs**, leveraged **closed-form expressions** (of the entropy)
- What about **complex functions**?
  - Order statistics (max/min, median)
  - Variability measures (variance)
  - **Multidimensional outputs**
- Output could be **discrete**, while the inputs are **continuous**
- **Data-driven techniques** [Gao+17] to **estimate** the entropy

Estimating Mutual Information for Discrete-Continuous Mixtures

Weihao Gao
Department of ECE
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
wgao9@illinois.edu

Sreeram Kannan
Department of Electrical Engineering
University of Washington
ksreeram@uw.edu

Sewoong Oh
Department of IESE
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
swoh@illinois.edu

Pramod Viswanath
Department of ECE
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
pramodv@illinois.edu

mutual information $\Leftrightarrow$ absolute loss

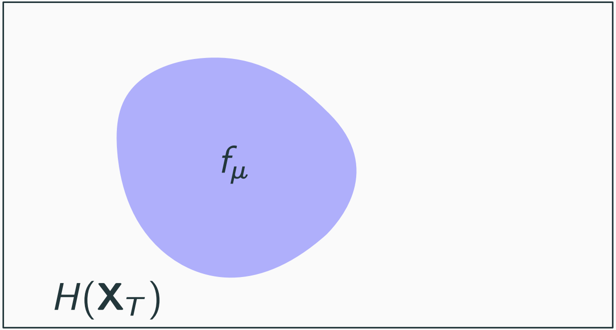
Interesting observations: simultaneous release

Variance and mean release

The total disclosure from **individual** function outputs f_μ and f_{σ^2} is **at least** the amount of information disclosed from a **joint release** $f_{(\mu,\sigma^2)}$?

$$f_{\sigma^2}(\mathbf{x}) = \frac{1}{n}\sum_i (x_i - f_\mu(\mathbf{x}))^2$$

$$\implies f_{(\mu,\sigma^2)}(\mathbf{x}) = (f_{\sigma^2}(\mathbf{x}), f_\mu(\mathbf{x}))$$

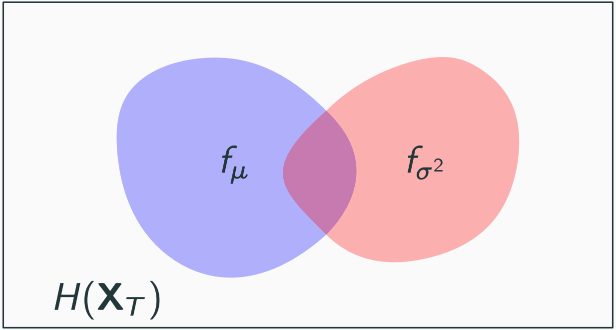


Variance and mean release

The total disclosure from **individual** function outputs f_μ and f_{σ^2} is **at least** the amount of information disclosed from a **joint release** $f_{(\mu,\sigma^2)}$?

$$f_{\sigma^2}(\mathbf{x}) = \frac{1}{n}\sum_i (x_i - f_\mu(\mathbf{x}))^2$$

$$\implies f_{(\mu,\sigma^2)}(\mathbf{x}) = (f_{\sigma^2}(\mathbf{x}), f_\mu(\mathbf{x}))$$



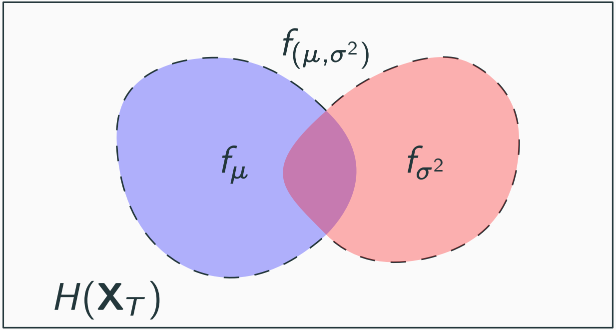
Interesting observations: simultaneous release

Variance and mean release

The total disclosure from **individual** function outputs f_μ and f_{σ^2} is **at least** the amount of information disclosed from a **joint release** $f_{(\mu,\sigma^2)}$?

$$f_{\sigma^2}(\mathbf{x}) = \frac{1}{n}\sum_i (x_i - f_\mu(\mathbf{x}))^2$$

$$\implies f_{(\mu,\sigma^2)}(\mathbf{x}) = (f_{\sigma^2}(\mathbf{x}), f_\mu(\mathbf{x}))$$



Variance and mean release

The total disclosure from **individual** function outputs f_μ and f_{σ^2} is **at least** the amount of information disclosed from a **joint release** $f_{(\mu,\sigma^2)}$?

$$f_{\sigma^2}(\mathbf{x}) = \frac{1}{n}\sum_i (x_i - f_\mu(\mathbf{x}))^2$$

$$\implies f_{(\mu,\sigma^2)}(\mathbf{x}) = (f_{\sigma^2}(\mathbf{x}), f_\mu(\mathbf{x}))$$

- **Gap** between the curves suggests A can learn **more** information about the target

$H_{f_\mu} + H_{f_{\sigma^2}}$

$H_{f_{(\mu,\sigma^2)}}$

$|S| = 2$

$|S| = 5$

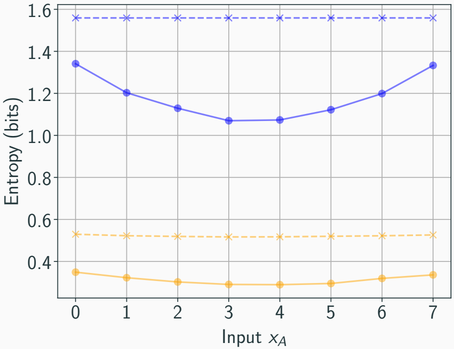


Figure 2: Abs. entropy loss, $\mathcal{U}(0,7)$ (lower is better)

Variance and mean release

The total disclosure from **individual** function outputs f_μ and f_{σ^2} is **at least** the amount of information disclosed from a **joint release** $f_{(\mu,\sigma^2)}$?

$$f_{\sigma^2}(\mathbf{x}) = \frac{1}{n}\sum_i (x_i - f_\mu(\mathbf{x}))^2$$

$$\implies f_{(\mu,\sigma^2)}(\mathbf{x}) = (f_{\sigma^2}(\mathbf{x}), f_\mu(\mathbf{x}))$$

– **Gap** between the curves suggests A can learn **more** information about the target

—●—	$H_{f_\mu} + H_{f_{\sigma^2}}$
--×--	$H_{f_{(\mu,\sigma^2)}}$

blue line	$\lvert S\rvert = 2$
orange line	$\lvert S\rvert = 5$

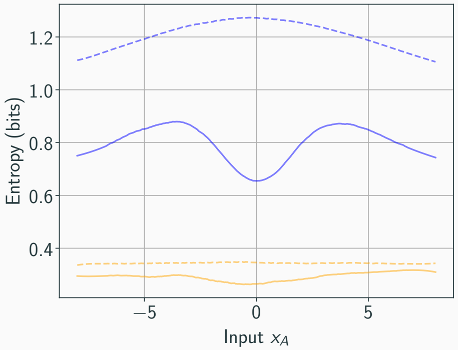


Figure 3: Abs. entropy loss, $\mathcal{N}(0,2)$ (lower is better)

Interesting observations: simultaneous release

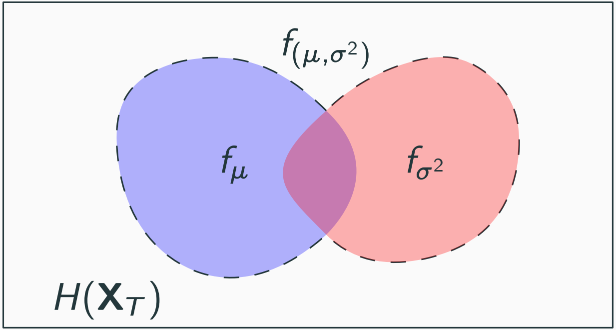
Variance and mean release

More information is revealed from the **joint release** $f_{(\mu,\sigma^2)}$ than from the **individual** function outputs f_μ and f_{σ^2}.

$$f_{\sigma^2}(\mathbf{x}) = \frac{1}{n}\sum_i (x_i - f_\mu(\mathbf{x}))^2$$

$$\implies f_{(\mu,\sigma^2)}(\mathbf{x}) = (f_{\sigma^2}(\mathbf{x}), f_\mu(\mathbf{x}))$$

– **Gap** between the curves suggests A can learn **more** information about the target



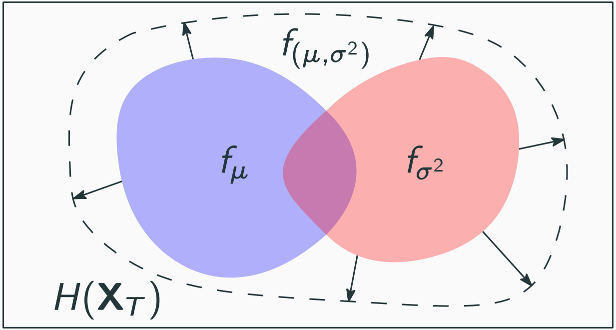
Variance and mean release

More information is revealed from the **joint release** $f_{(\mu,\sigma^2)}$ than from the **individual** function outputs f_μ and f_{σ^2}.

$$f_{\sigma^2}(\mathbf{x}) = \frac{1}{n}\sum_i (x_i - f_\mu(\mathbf{x}))^2$$

$$\implies f_{(\mu,\sigma^2)}(\mathbf{x}) = (f_{\sigma^2}(\mathbf{x}), f_\mu(\mathbf{x}))$$

– **Gap** between the curves suggests A can learn **more** information about the target



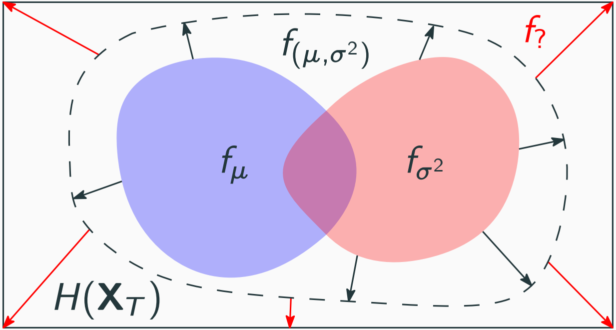
Variance and mean release

More information is revealed from the **joint release** $f_{(\mu,\sigma^2)}$ than from the **individual** function outputs f_μ and f_{σ^2}.

$$f_{\sigma^2}(\mathbf{x}) = \frac{1}{n}\sum_i (x_i - f_\mu(\mathbf{x}))^2$$

$$\implies f_{(\mu,\sigma^2)}(\mathbf{x}) = (f_{\sigma^2}(\mathbf{x}), f_\mu(\mathbf{x}))$$

– **Gap** between the curves suggests A can learn **more** information about the target



Ongoing work

- Theoretical basis from our comprehensive analysis of the average
- *Much* to learn for complex functions

Analytical and data-driven evaluation of complex functions

- Derive analytical expressions the entropy
- Estimators suffer from the "curse of dimensionality"
  - Can project high-dimensional data into lower-dimensional space

Mitigation strategies

- Synthetic inputs
- Modifying the function
- Adding noise (DP)

Alternate metrics

- (min-, g-, cross) entropies

Conclusions

Conclusions

- RSS for any number of parties
- Information disclosure analysis
- Number of interesting current/future research directions

Thank you!

Questions?

References

[AH17] P. Ah-Fat and M. Huth. "Secure Multi-party Computation: Information Flow of Outputs and Game Theory". In: *International Conference on Principles of Security and Trust*. 2017, pp. 71–92.

[Ali+13] M. Aliasgari, M. Blanton, Y. Zhang, and A. Steele. "Secure Computation on Floating Point Numbers". In: *Network and Distributed System Security Symposium (NDSS)*. 2013.

[Bac24] A. Baccarini. "New Directions in Secure Multi-Party Computation: Techniques and Information Disclosure Analysis". PhD Thesis. University at Buffalo, 2024.

[BBZ24a] A. Baccarini, M. Blanton, and S. Zou. "Understanding Information Disclosure from Secure Computation Output: A Comprehensive Study of Average Salary Computation". In: *ACM Transactions on Privacy and Security (TOPS)* 28.1 (2024), pp. 1–36.

[BBZ24b] A. Baccarini, M. Blanton, and S. Zou. "Understanding Information Disclosure from Secure Computation Output: A Study of Average Salary Computation". In: *ACM CODASPY*. 2024, pp. 187–198.

[BGY23] M. Blanton, M. T. Goodrich, and C. Yuan. "Secure and Accurate Summation of Many Floating-Point Numbers". In: *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2023.3 (2023), pp. 432–445.

[Dam+19] I. Damgård, D. Escudero, T. Frederiksen, M. Keller, P. Scholl, and N. Volgushev. "New Primitives for Actively-Secure MPC over Rings with Applications to Private Machine Learning". In: *IEEE Symposium on Security and Privacy (S&P)*. 2019, pp. 1102–1120.

[Gao+17] W. Gao, S. Kannan, S. Oh, and P. Viswanath. "Estimating mutual information for discrete-continuous mixtures". In: *Proceedings on Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017), pp. 5988–5999.

[ISN87] M. Ito, A. Saito, and T. Nishizeki. "Secret Sharing Schemes Realizing General Access Structures". In: *IEEE Global Telecommunication Conference (GLOBECOM)*. 1987, pp. 99–102.

References

[Kel20] M. Keller. "MP-SPDZ: A Versatile Framework for Multi-Party Computation". In: *ACM Conference on Computer and Communications Security (CCS)*. 2020, pp. 1575–1590.

[Mun11] J. D. Munoz. "Rapid path-planning algorithms for autonomous proximity operations of satellites". PhD Thesis. University of Florida, 2011.

[Rat+21] D. Rathee, M. Rathee, R. K. K. Goli, D. Gupta, R. Sharma, N. Chandran, and A. Rastogi. "SiRnn: A math library for secure RNN inference". In: *IEEE Symposium on Security and Privacy (S&P)*. 2021, pp. 1003–1020.

[Rat+22] D. Rathee, A. Bhattacharya, R. Sharma, D. Gupta, N. Chandran, and A. Rastogi. "SecFloat: Accurate Floating-Point meets Secure 2-Party Computation". In: *IEEE Symposium on Security and Privacy (S&P)*. 2022, pp. 1553–1553.

[Sha79] A. Shamir. "How to Share a Secret". In: *Communications of the ACM* 22.11 (1979), pp. 612–613.

[SVG24] S. Sasy, A. Vadapalli, and I. Goldberg. "PRAC: Round-Efficient 3-Party MPC for Dynamic Data Structures". In: *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2024.3 (2024), pp. 692–714.

[Vir+18] J. Virgili-Llop, C. Zagaris, H. Park, R. Zappulla, and M. Romano. "Experimental evaluation of model predictive control and inverse dynamics control for spacecraft proximity and docking maneuvers". In: *CEAS Space Journal* 10 (2018), pp. 37–49.

[Zap+18] R. Zappulla, H. Park, J. Virgili-Llop, and M. Romano. "Real-time autonomous spacecraft proximity maneuvers and docking using an adaptive artificial potential field approach". In: *IEEE Transactions on Control Systems Technology* 27.6 (2018), pp. 2598–2605.

[ZSB13] Y. Zhang, A. Steele, and M. Blanton. "PICCO: A general-purpose compiler for private distributed computation". In: *ACM Conference on Computer and Communications Security (CCS)*. 2013, pp. 813–826.

Intuitive observations: maximum

Maximum

An adversary **maximizes** the information learned by **minimizing** their influence.

$$f_{\max}(\mathbf{x}) = \max_i x_i$$

– Inverse behavior for $f_{\min}(\mathbf{x})$

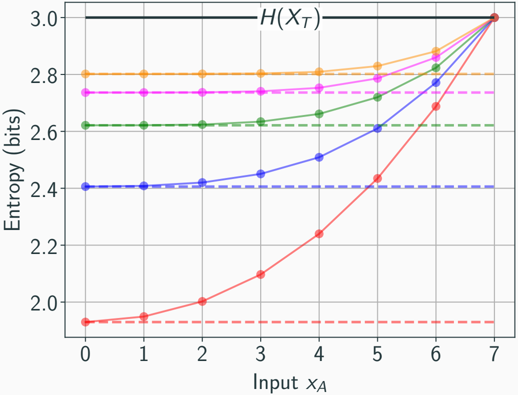


Figure 4: Uniform $\mathcal{U}(0, 7)$, $H(\mathbf{X}_T \mid \mathbf{X}_A = \mathrm{x}_A, O)$

Intuitive observations: maximum

Maximum

An adversary **maximizes** the information learned by **minimizing** their influence.

$$f_{\max}(\mathbf{x}) = \max_i x_i$$

- Inverse behavior for $f_{\min}(\mathbf{x})$

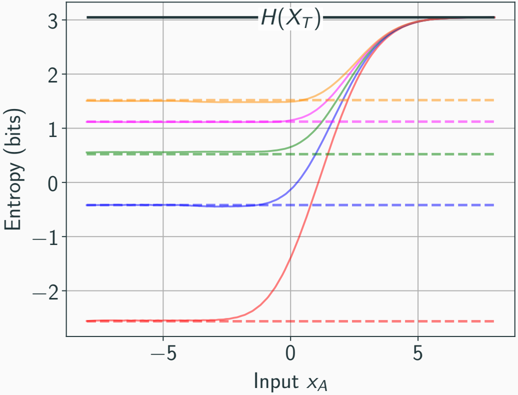


Figure 4: Normal $\mathcal{N}(0, 4.0)$, $H(\mathbf{X}_T \mid \mathbf{X}_A = \mathbf{x}_A, O)$

Binary-to-arithmetic conversion (B2A)

- Often operate on **individual bits** of secrets, requiring conversion from $\mathbb{Z}_2 \to \mathbb{Z}_{2^k}$
- Prior works use **RandBit** [Dam+19], requires temporary computation in $\mathbb{Z}_{2^{k+2}}$
  - E.g., $k = 8$ requires 16-bit integers, **doubling** the communication
- Blanton et al. [BGY23] **eliminated** this requirement for 3-party RSS

Generalization of [BGY23] to any n **[Bac24]**

1. t parties locally XOR a subset of their shares, enter result into computation
2. Remaining $t+1$ parties "locally reshare" last share (all but one share is nonzero)
3. Compute XOR (in $\mathbb{Z}_{2^k}$) of local XOR(s) and the last share as a tree

- Can use approach to generate shared random bits (RandBit) **without** $\mathbb{Z}_{2^{k+2}}$
- Up to 6.5× faster for 3 parties, 2× faster for 5 parties

Floating-point representation

$$\tilde{a} = \underbrace{\boxed{\text{sign } s}}_{1}\ \underbrace{\boxed{\text{exponent } e}}_{p}\ \underbrace{\boxed{\text{mantissa (significand) } m}}_{q(+\,1)}$$

$$\tilde{a} = (1 - z) \cdot (1 - 2s) \cdot 2^{e} \cdot m = (z, s, e, m)$$

$$z = \begin{cases} 1 & \text{if } \tilde{a} = 0 \\ 0 & \text{otherwise} \end{cases}$$

[Ali+13,Rat+22]

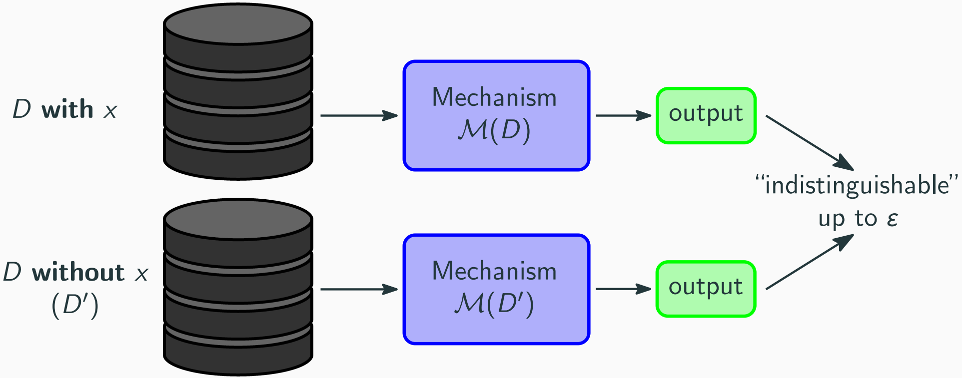
Most operations are conceptually similar to their integer equivalents...

- Comparisons $[\tilde{a}] \overset{?}{<} [\tilde{b}]$
- Multiplication $[\tilde{a}] \cdot [\tilde{b}]$
- Division $[\tilde{a}] / [\tilde{b}]$

... except for addition $[\tilde{a}] + [\tilde{b}]$

- Exponents, mantissas must be **obliviously aligned and normalized**
- Comparisons, left/right shifts, prefix ops, rounding, ...

Differential privacy



– Useful for large databases (think $n \geq 10{,}000$)...
– ... but **absolutely destroys** the utility of the result (up to 100% error!)
– Our goal: first **determine** if a function discloses too much information