Secure and Efficient Computing on Private Data

Eleftheria Makri | *for the CWI Student Seminar*

14/04/2023

Universiteit Leiden

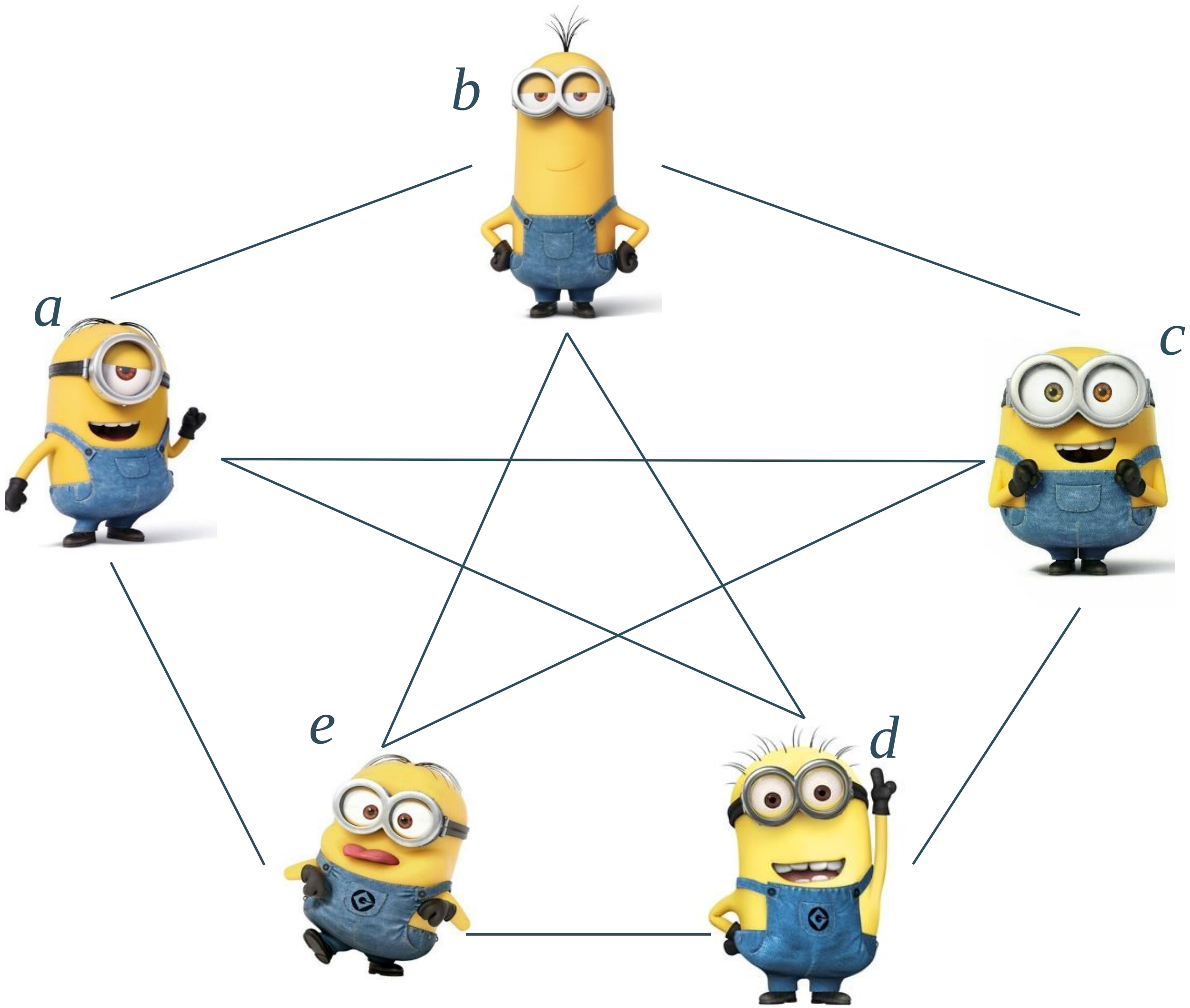
Media Player Classic – MPC

An Information-Driven Society


PRIVACY

“How to allow the collection and purposeful processing of private data, without compromising individual privacy?”

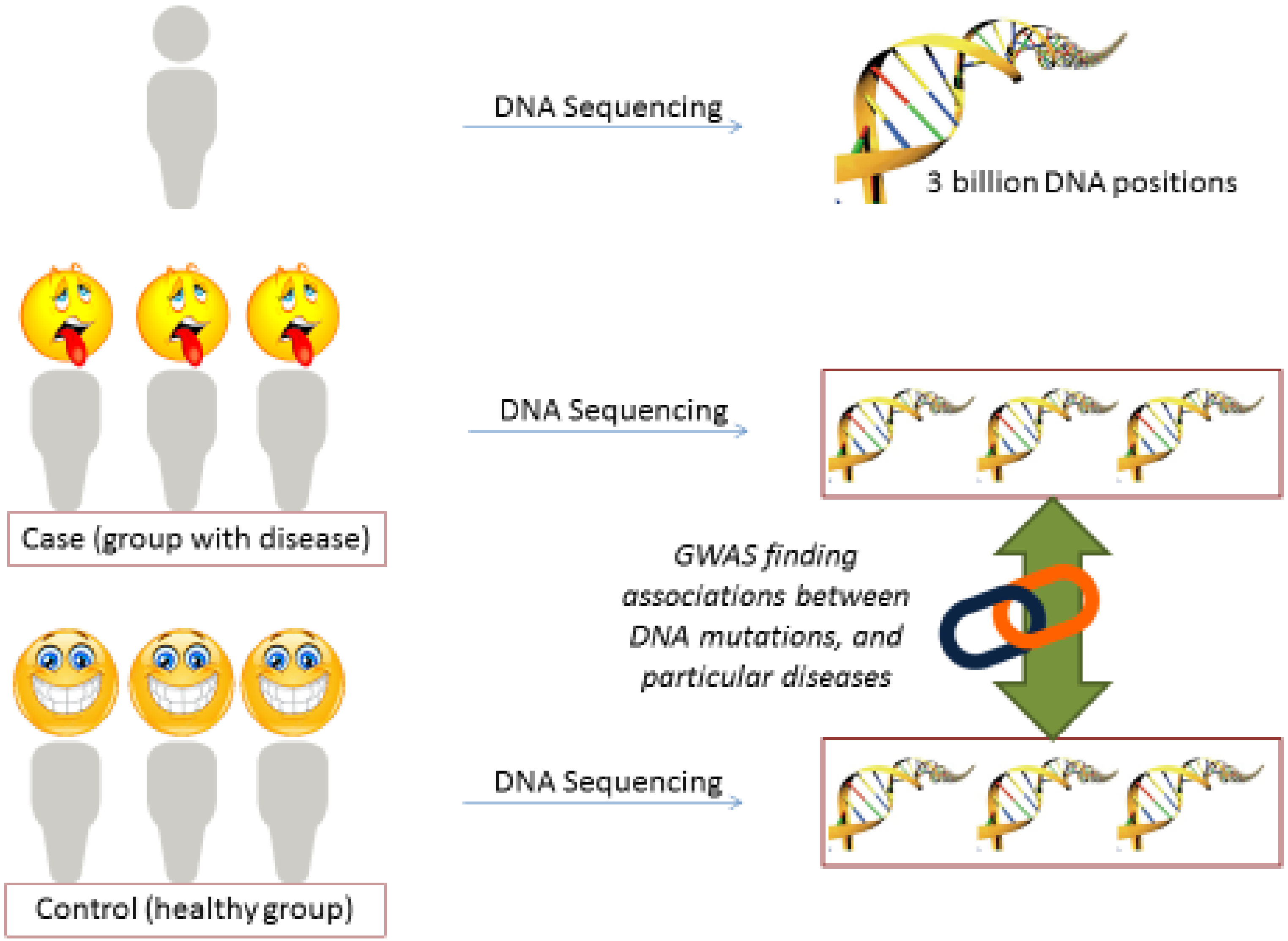
Multiparty Computation – MPC



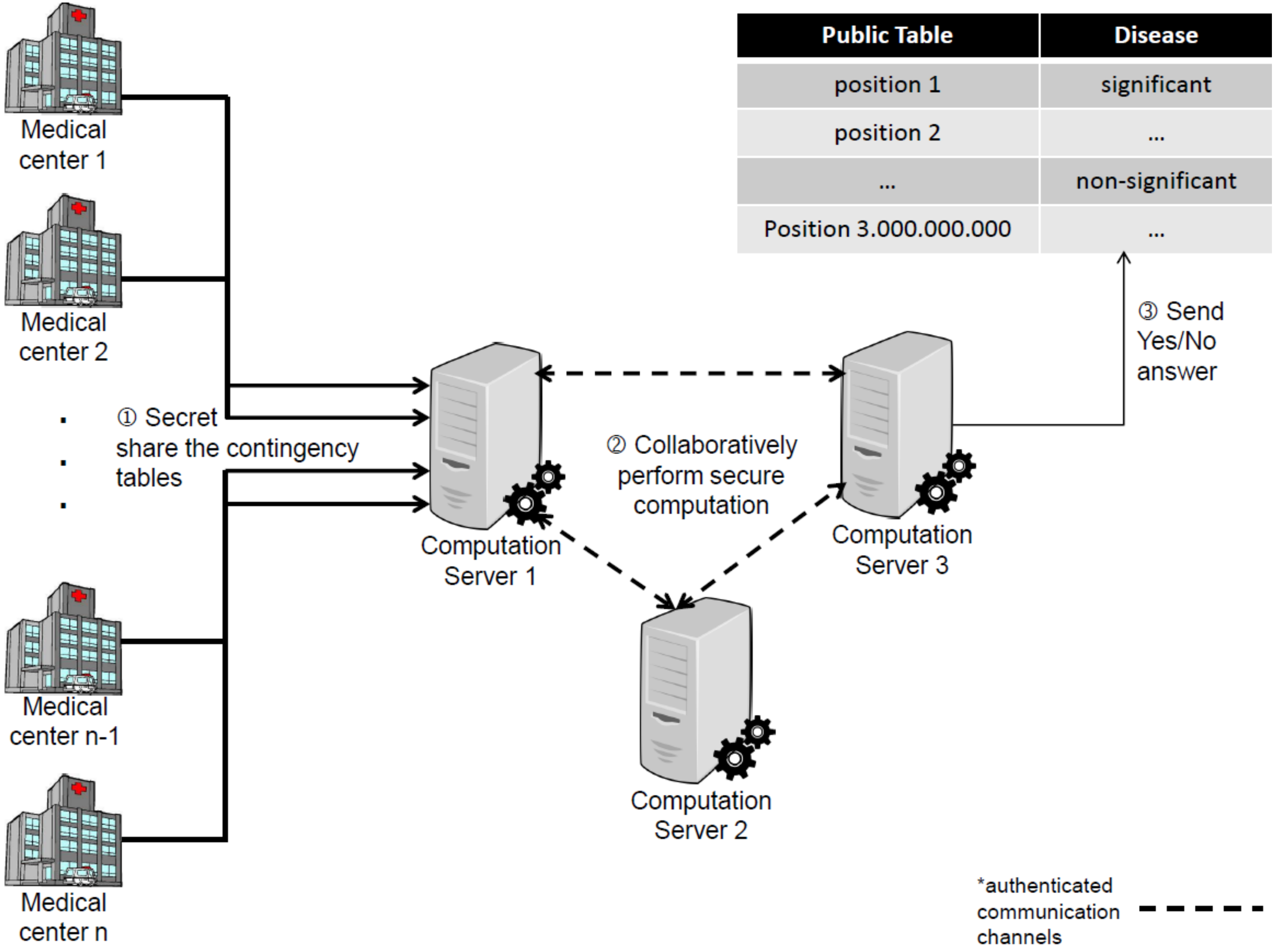
→ Securely compute $f(a, b, c, d, e)$.

Application Scenario 1:
Privacy-Preserving Genome-Wide Association Studies

Privacy-Preserving GWAS

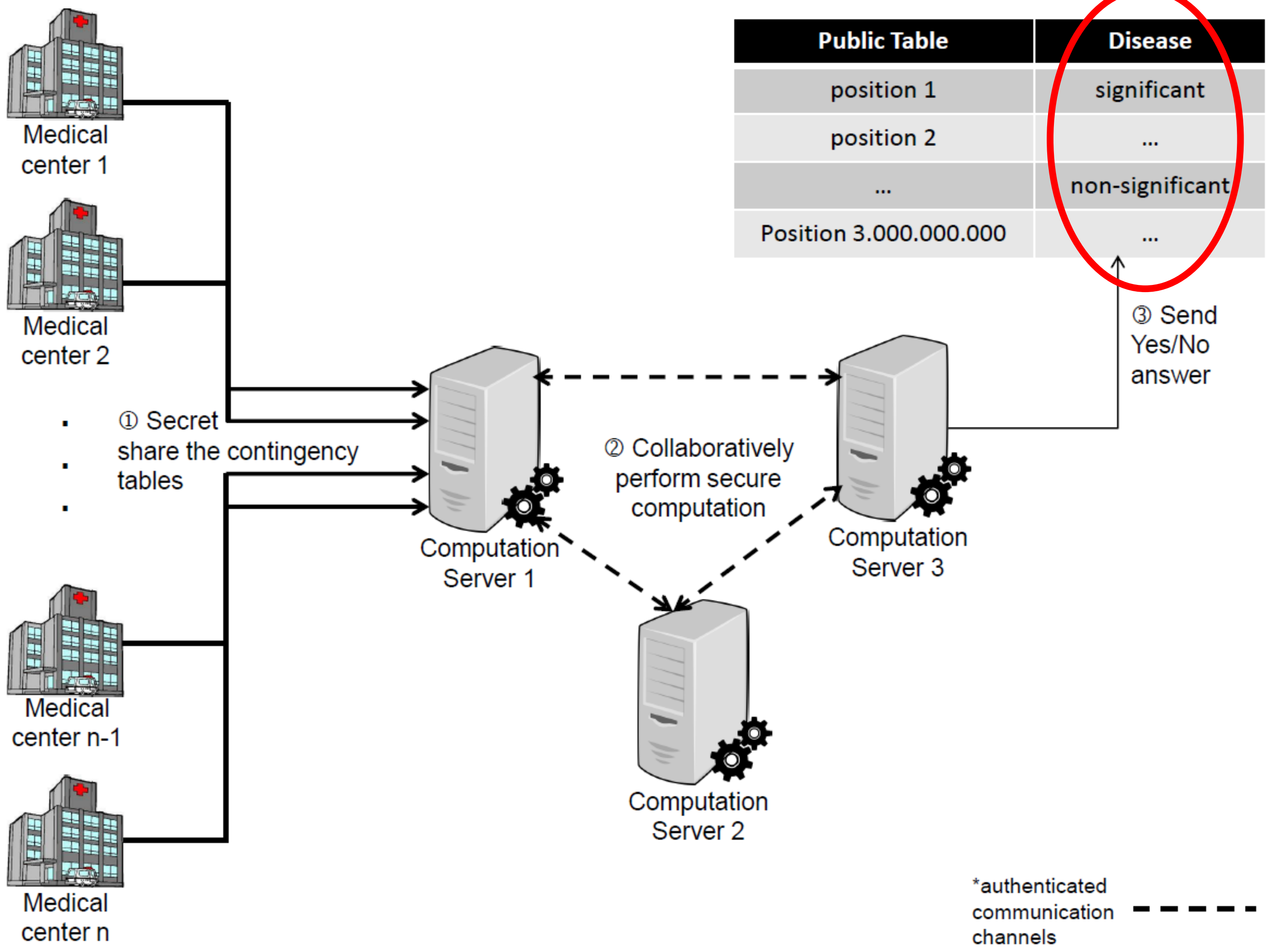


MPC-based Solution



Public Table	Disease
position 1	significant
position 2	...
...	non-significant
Position 3.000.000.000	...

MPC-based Solution



Public Table	Disease
position 1	significant
position 2	...
...	non-significant
Position 3.000.000.000	...

Problem of re-identification

Problem of re-identification

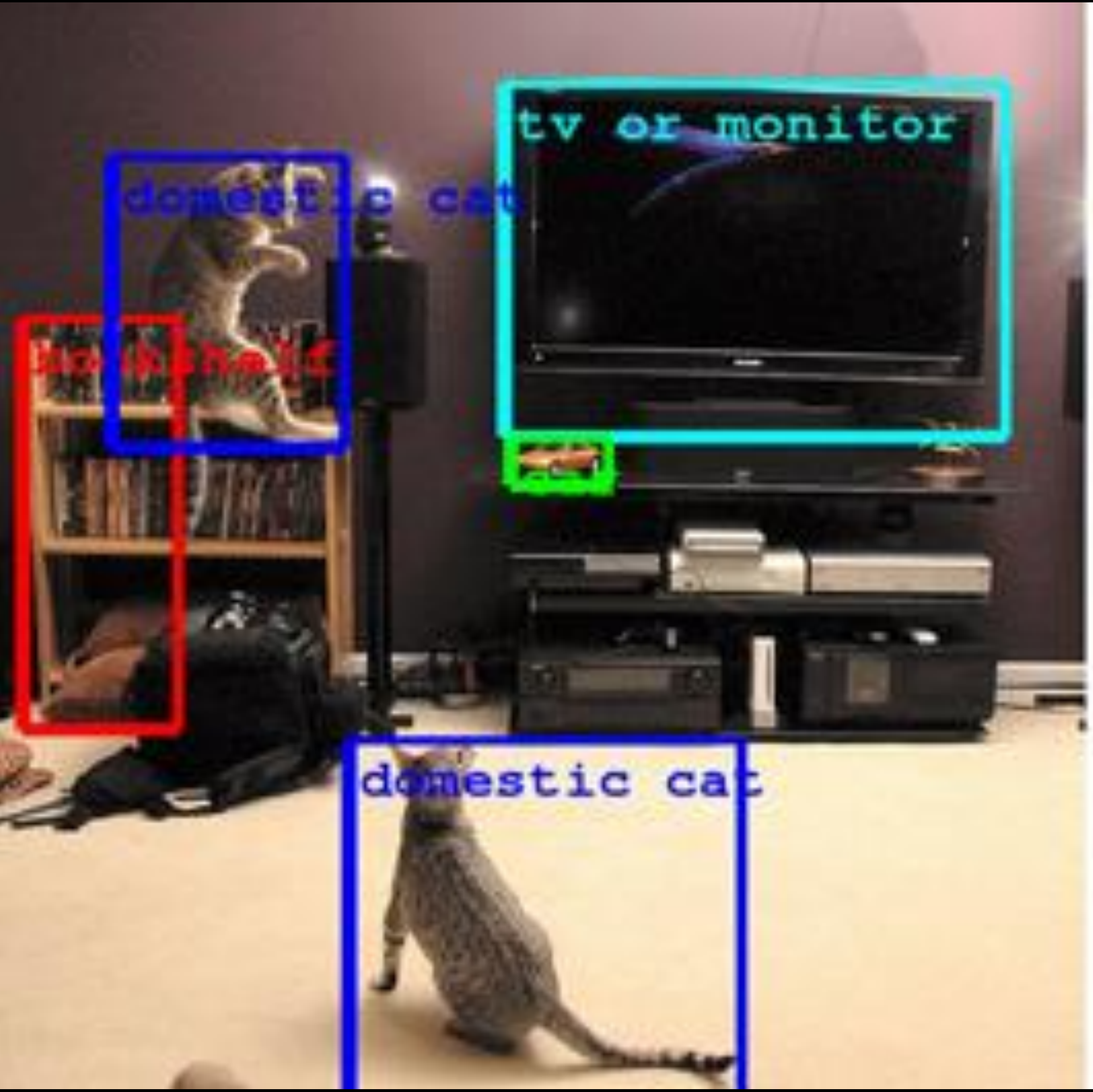
MPC-based GWAS: Performance

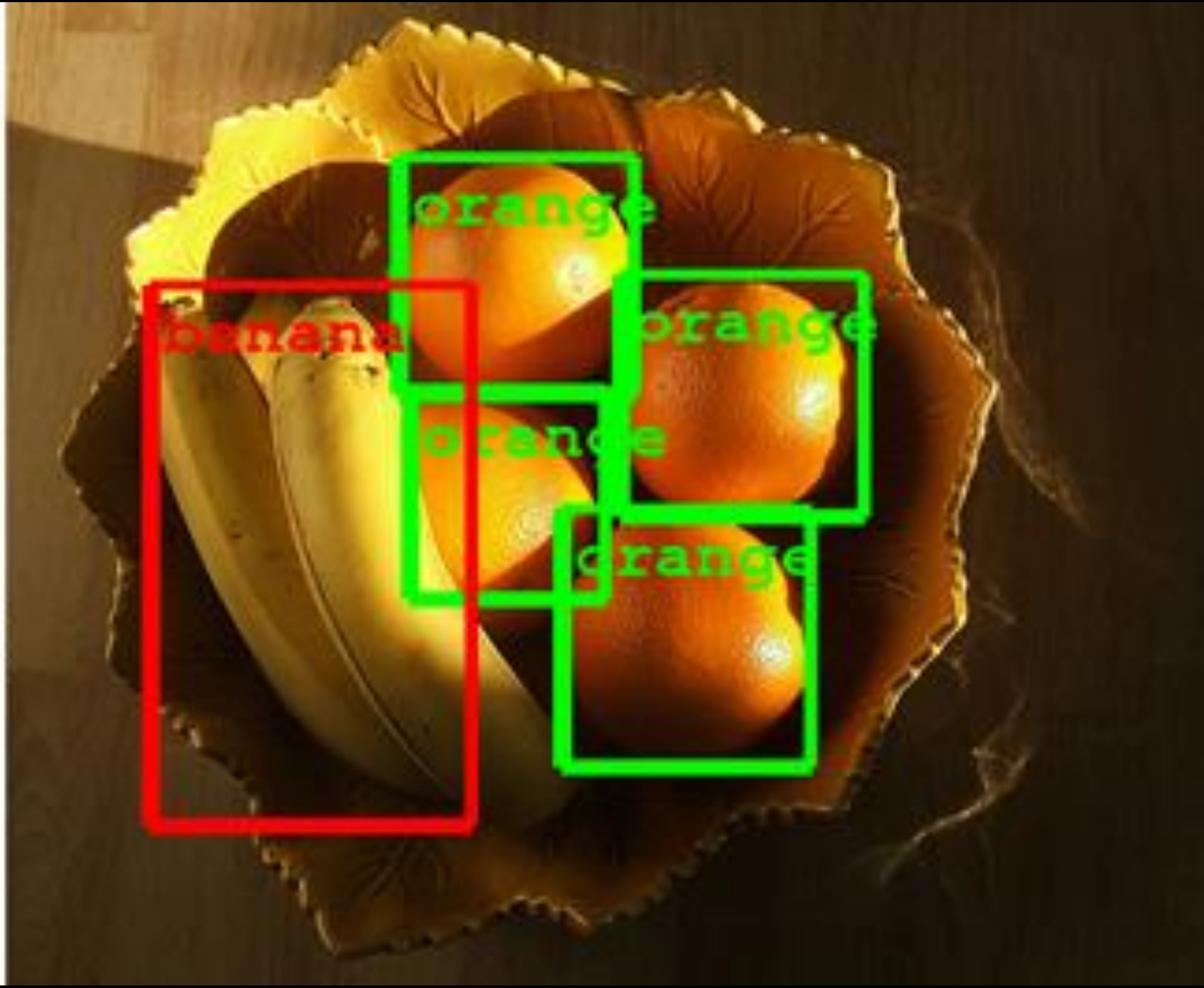
Medical Centers	Number of Patients	CPU Time (Server 1)	Data Sent (Server 1)	CPU Time (Server 2, and Server 3)
20	200000	2.2ms	12.7KB	1.9ms
40	400000	2.3ms	17.8KB	2.0ms
60	600000	2.3ms	23KB	2.0ms
80	800000	2.5ms	28.1KB	2.2ms
100	1000000	2.4ms	33.2KB	2.1ms

MPC-based GWAS vs. HE-based GWAS

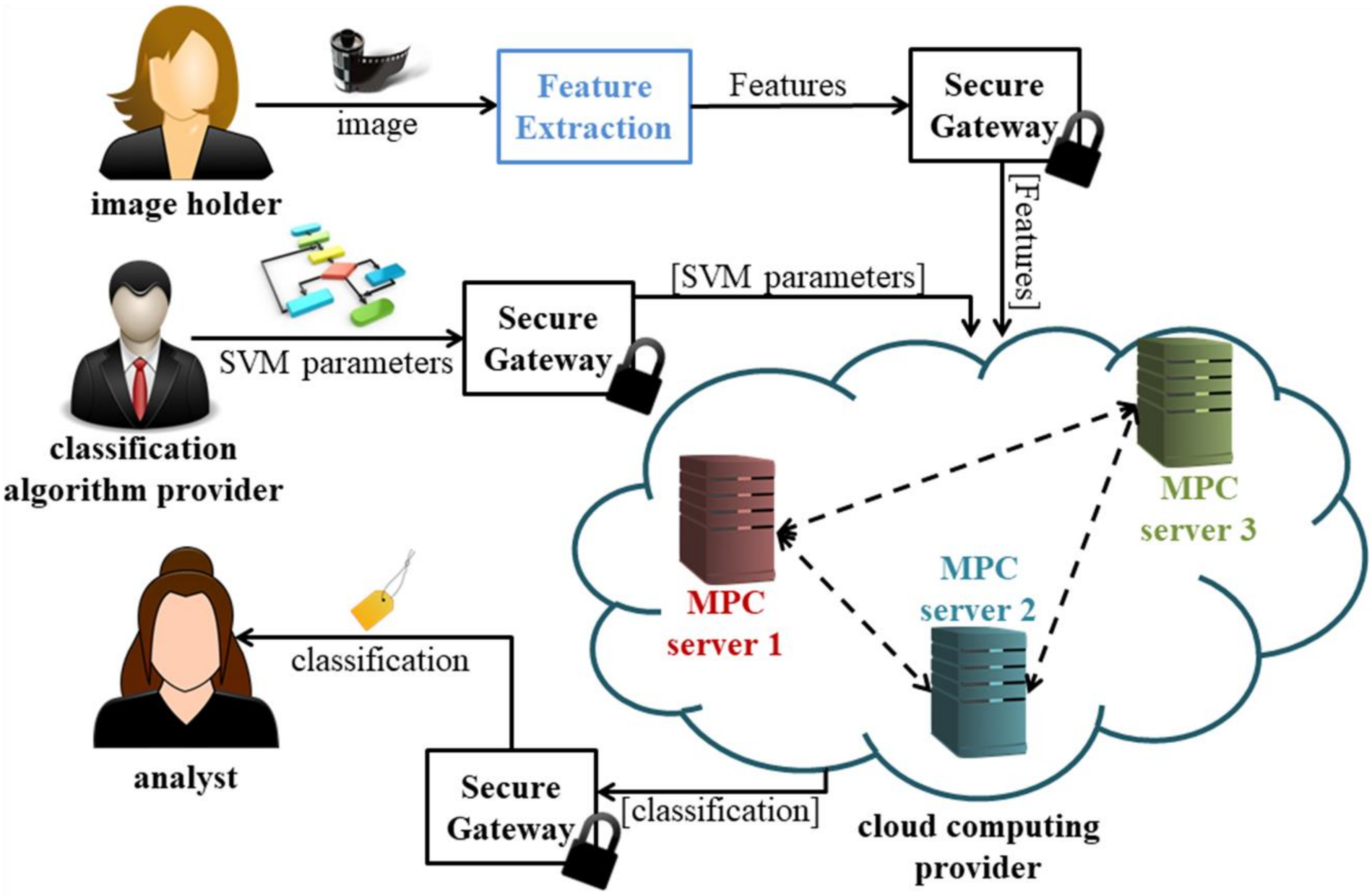
MPC-based GWAS	HE-based GWAS
CPU time in the range of milliseconds	CPU time in the range of seconds
Total communication cost in the range of KB	Total communication cost in the range of MB
Multiple parties (≥ 2) required to perform the computation	One party performs the computation
Active Security Guarantees	Semi-honest Decryptor assumption

Application Scenario 2:
Private Image Classification

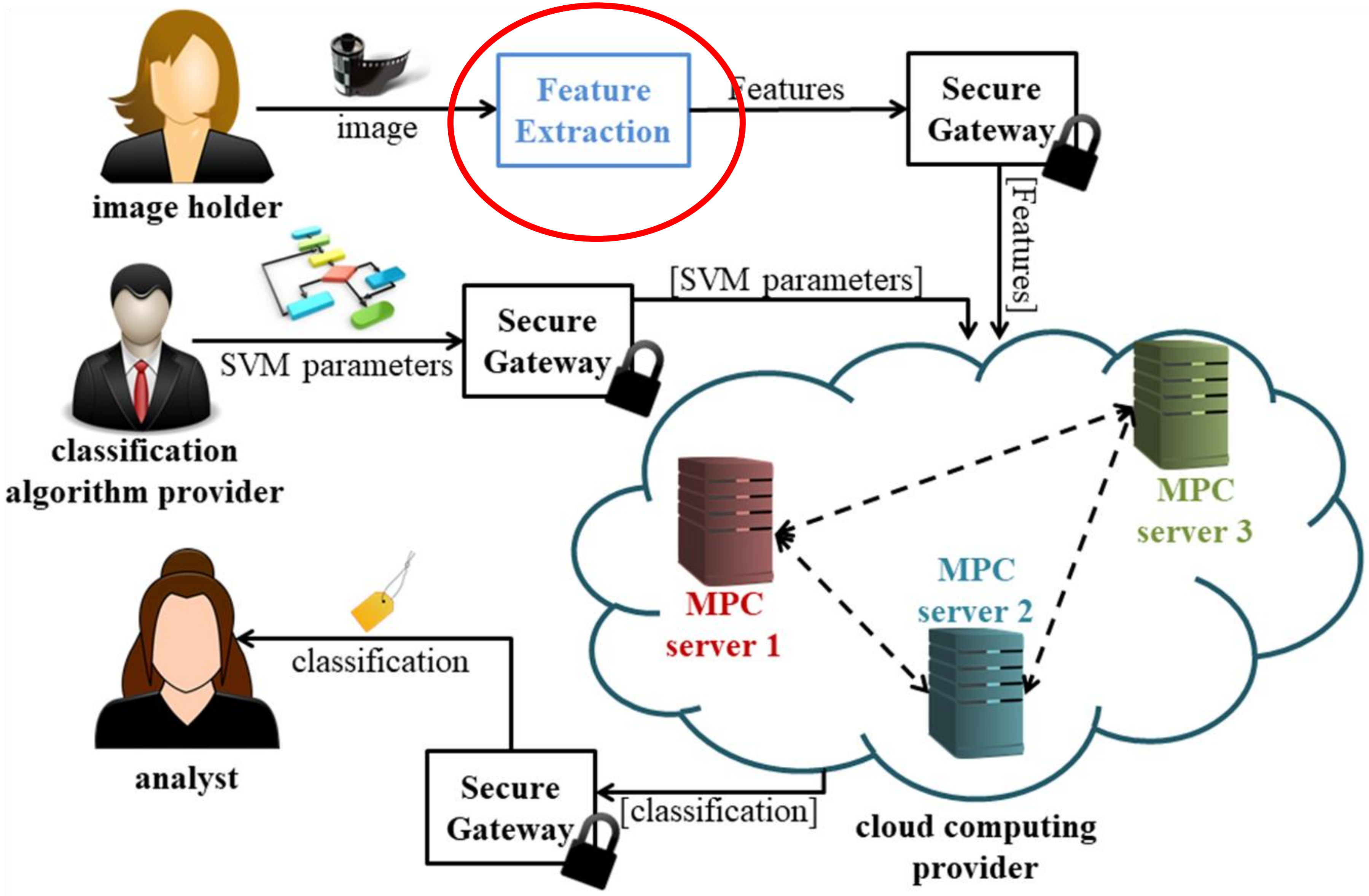

tv or monitor
domestic cat
domestic cat


orange
orange
banana
orange
orange

EPIC: Efficient Private Image Classification

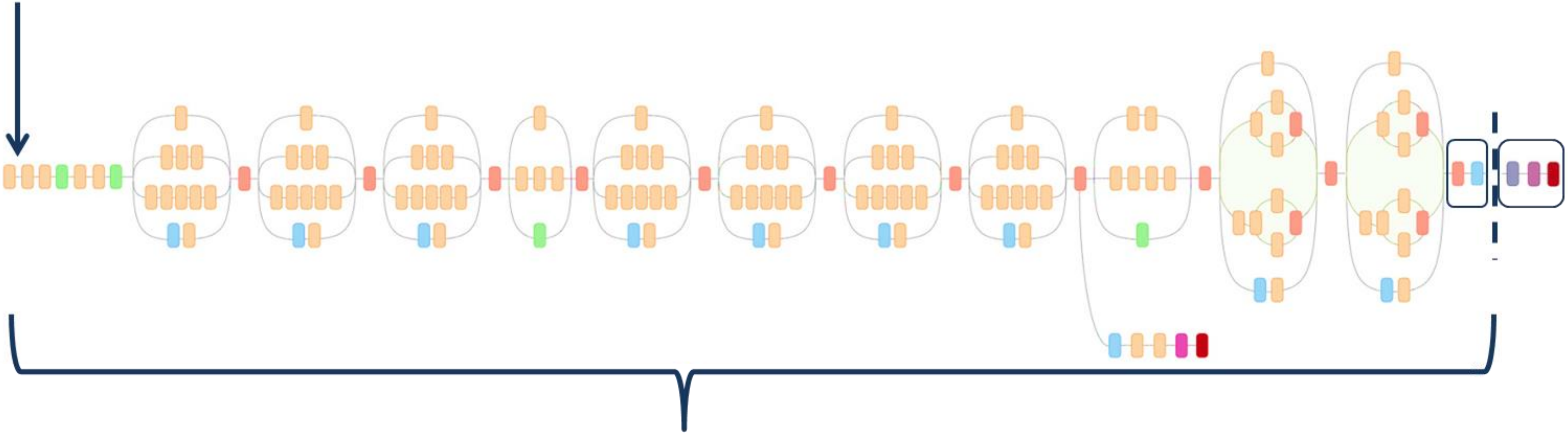


EPIC: Efficient Private Image Classification



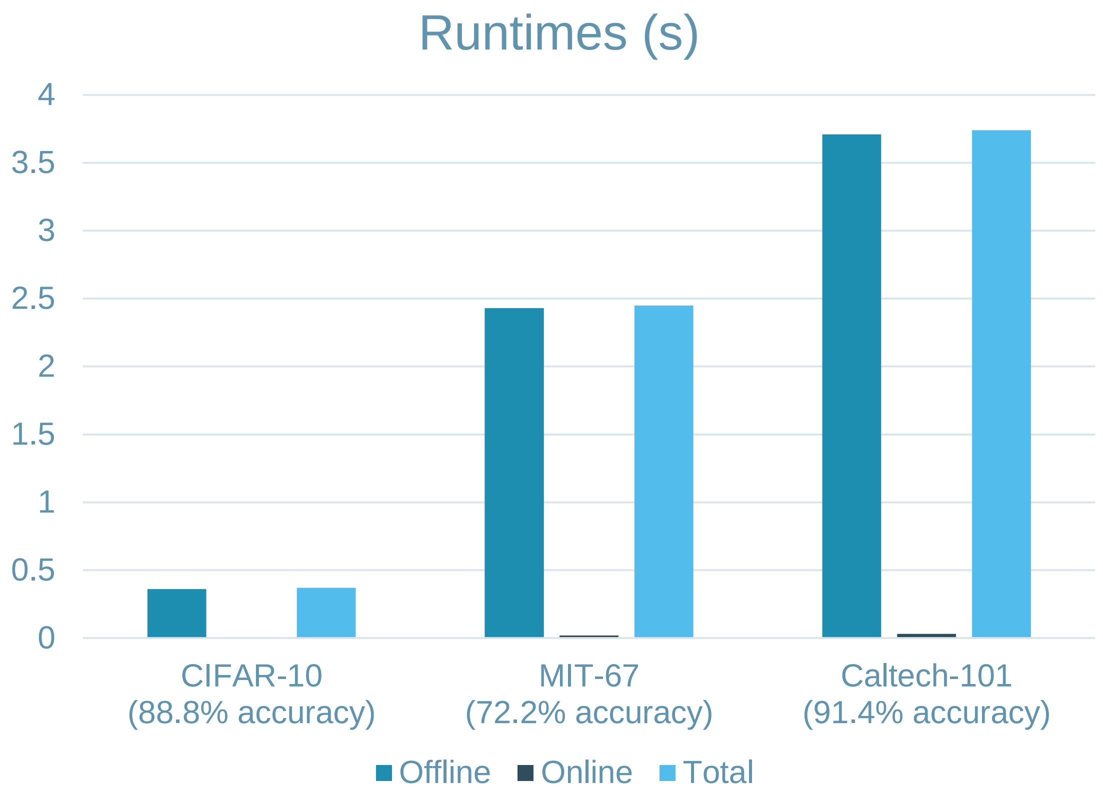
Transfer Learning Feature Extraction (or: Learning from the Masters)

Plaintext (non-sensitive) images

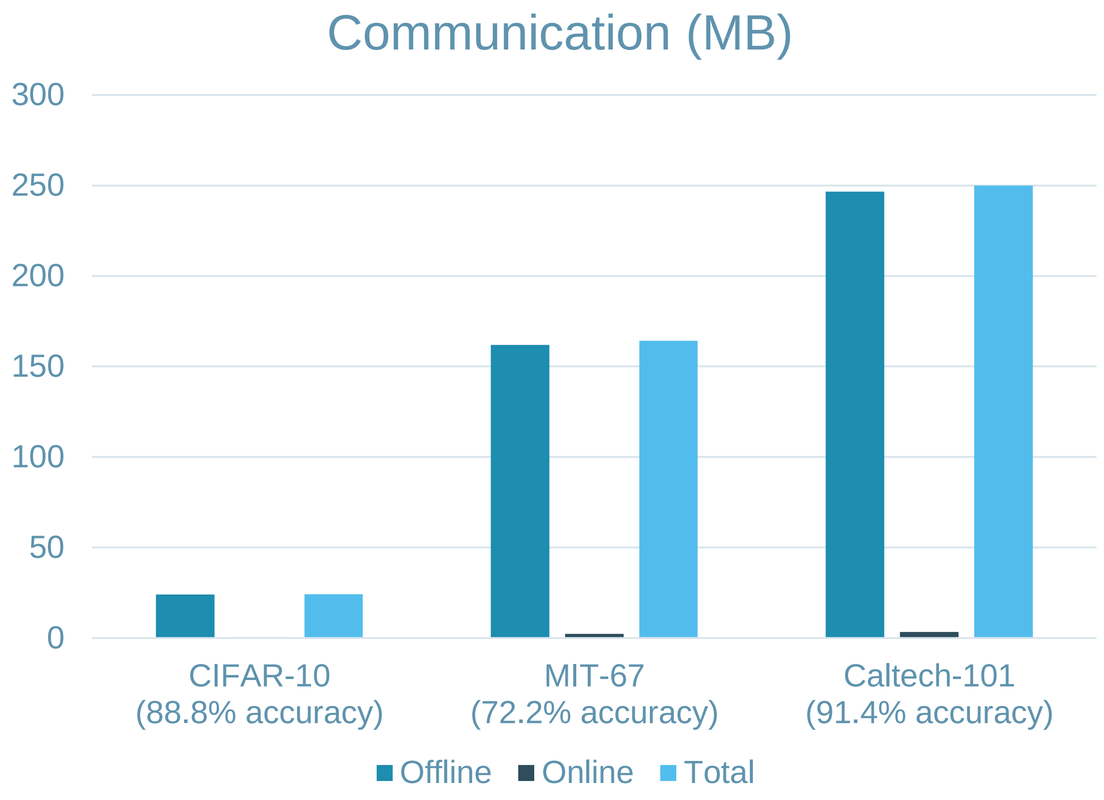


EPIC Performance – Simple Variant

Computation Cost

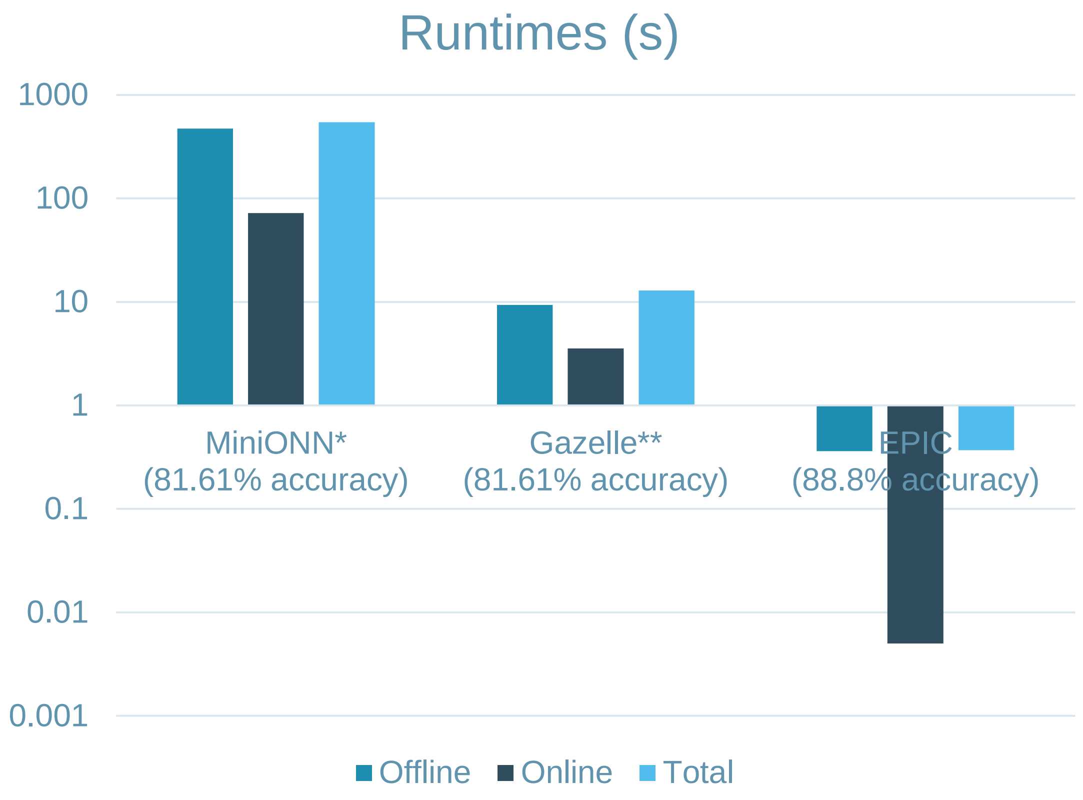


Communication Cost

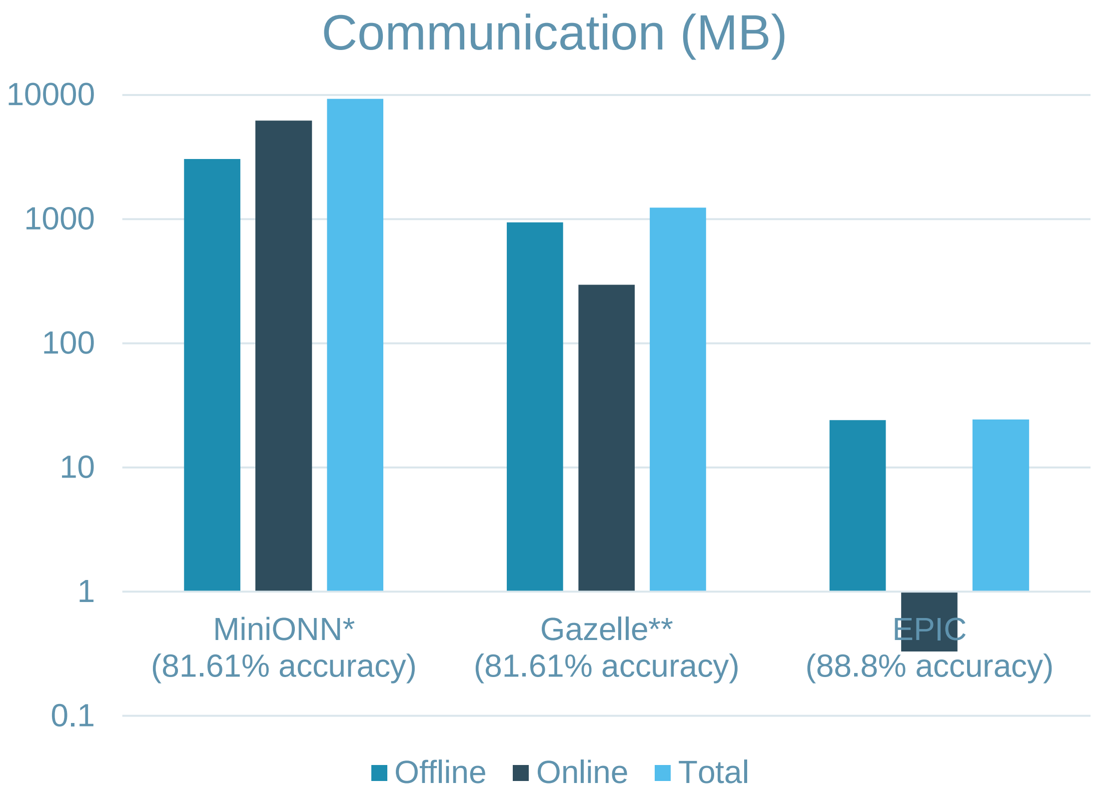


Performance of the state-of-the-art private image classification

Computation Cost



Communication Cost



* Jian Liu, Mika Juuti, Yao Lu, N. Asokan. **Oblivious Neural Network Predictions via MiniONN Transformations**. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 619-631). ACM.*

** Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. **GAZELLE: A low latency framework for secure neural network inference**. *In 27th USENIX Security Symposium (USENIX Security '18), Baltimore, MD, 2018. USENIX Association*.

Application Scenario 3:
Secure RSA Modulus Generation

RSA Modulus

- A biprime N, with two secret prime factors, p and q.

- Heart of the first public key cryptosystem; security based on factoring hardness assumption.

Why RSA Moduli?

- Signatures and Encryption
  - [RSA-77], [Paillier-99].
- Cryptographic accumulators
  - [Benaloh-deMare-93], [Camenisch-Lysyanskaya-02], [Li-Li-Xue-07], [Boneh-Bünz-Fisch-19],
- VDF and Timelock puzzles
  - [Rivest-Shamir-Wagner-99], Boneh-Bonneau-Bünz-Fisch-18], [Wesolowski-19], [Pietrzak-19], [Ephraim-Freitag-Komargodski-Pass-19].
- Efficient zk-SNARKs
  - [Bünz-Fisch-Szepieniec-19], [Lai-Malavolta-19]
- And others...

Why *distributed* RSA Moduli?

- Threshold Cryptography

Call 2021a for Feedback on Criteria for Threshold Schemes

NIST Multi-party Threshold Cryptography

2021-July-02: https://csrc.nist.gov/projects/threshold-cryptography

Please send comments to threshold-MP-call-2021a@nist.gov by September 13, 2021.

1. Scope of proposals. The future call for proposals will be intended to gather expert submissions of concrete threshold schemes for primitives that are *interchangeable* (in the sense of IR 8214A, Section 2.4) with[2] ECDSA, EdDSA, RSA signing/decryption, RSA keygen, AES, and ECC-based key agreement.[3] After an evaluation period, and possibly various stages for tweaks,

Why *distributed* RSA Moduli?

- *Companies or foundations*

LIGERO

Secure and Private Collaboration for Blockchains and Beyond

VDF Alliance

Connections with related work

CCD+20

OT based

Can we make it work over arithmetic circuits?

CHI+20

GMW based

OMG! They've solved it!
With a central coordinator and special SHE packing

Connections with related work

CCD+20
OT based

Can we make it work over arithmetic circuits?

CHI+20
GMW based

OMG! They've solved it!
With a central coordinator and special SHE packing

BonehFranklin97
Protocol Blueprint

MalkinWuBoneh99
Distributed sieving

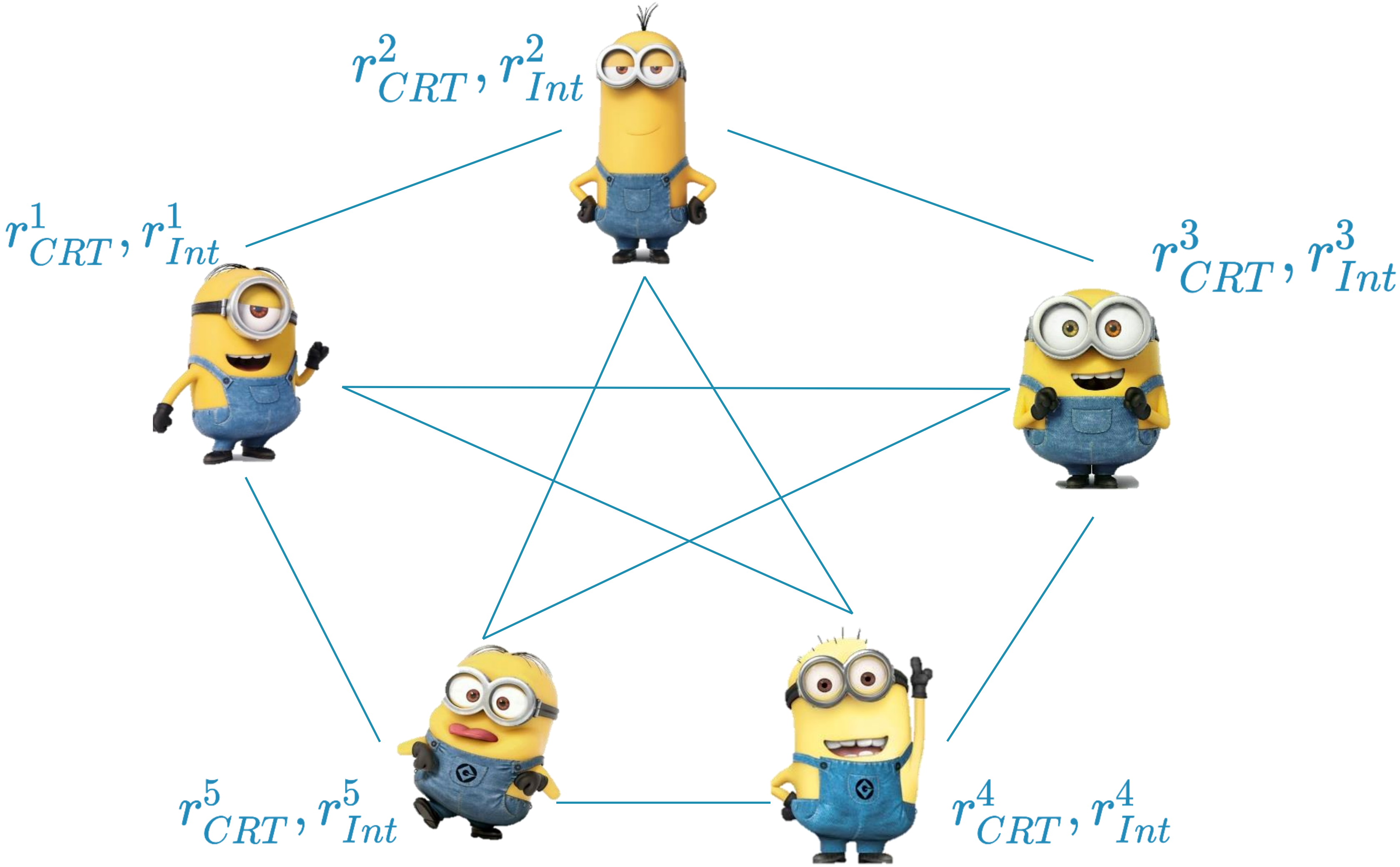
DamgårdMikkelsen2010
Integer sharing

GRR+16
Malicious exponentiation

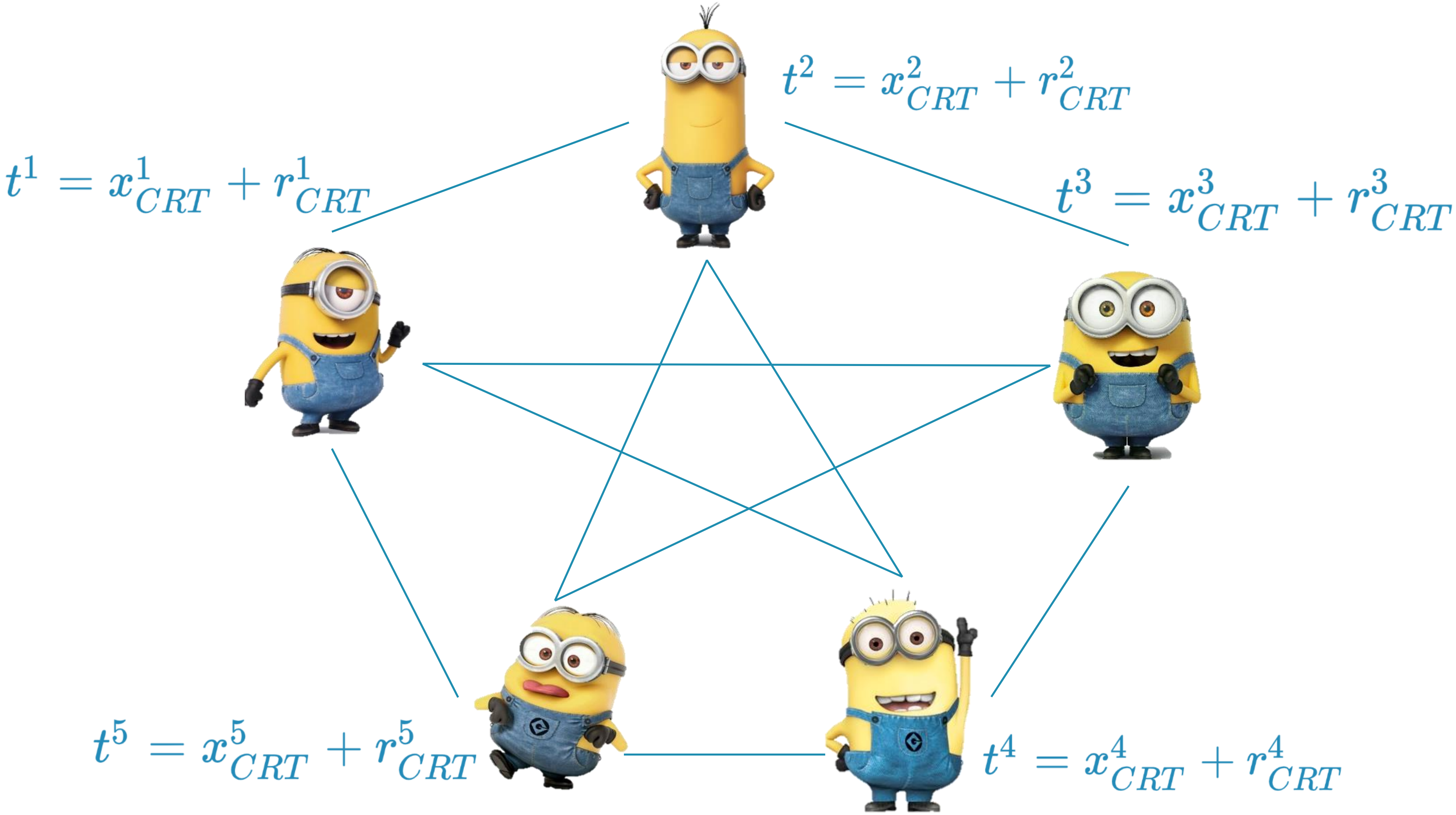
RST+19
Core of malicious check

Ours
Generic MPC

ConvInt Protocol

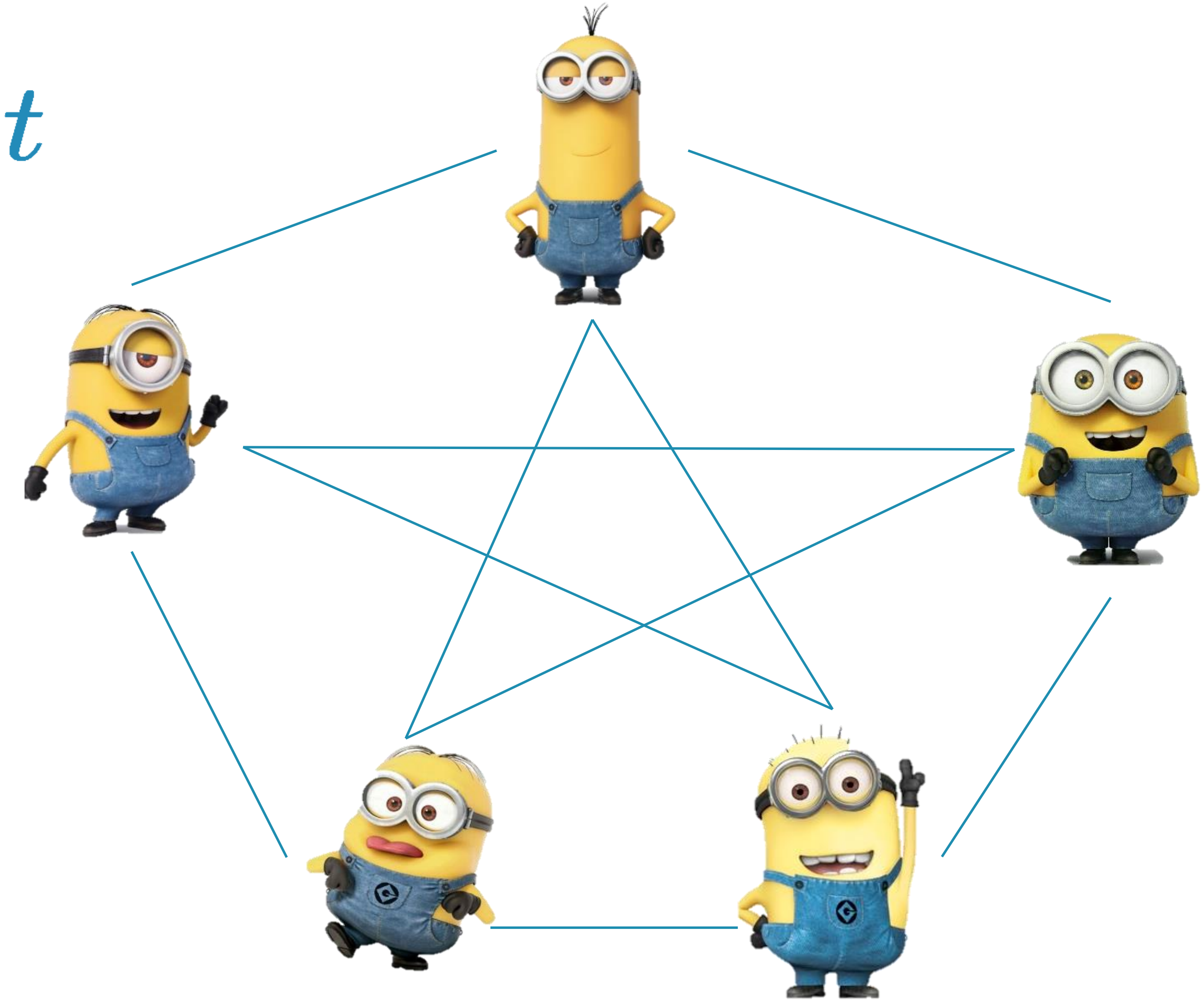


ConvInt Protocol

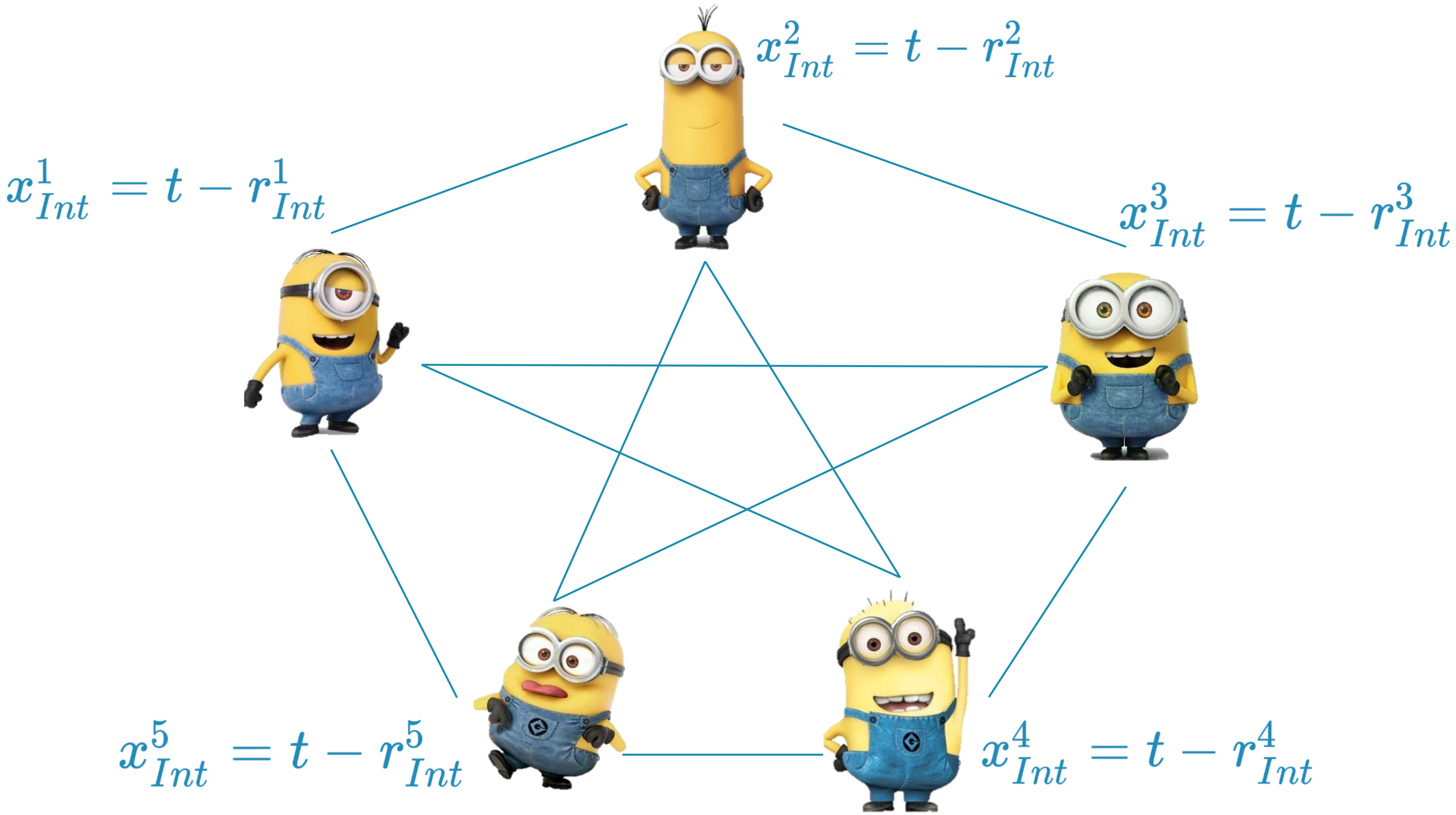


ConvInt Protocol

Open t



ConvInt Protocol

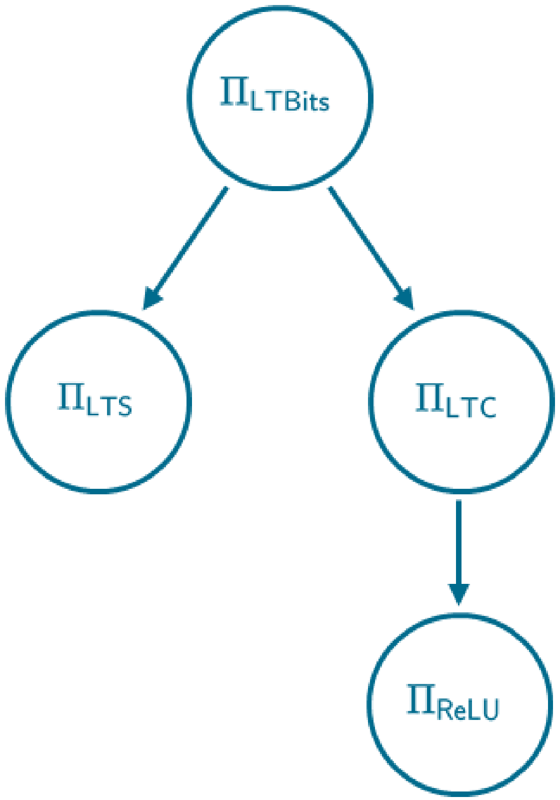


Contributions in Secure RSA modulus Generation

- RSA modulus generation protocol with generic MPC.
- Exploit *Distributed Sieving techniques* and *public knowledge* to perform parts of the protocol semi-honestly without degrading security.
- Convert to Integer protocol, of independent interest.
- Up to 37x better communication cost compared to CCD+20.

Improving MPC Primitives:
The Case of Comparisons

Rabbit: Comparison Protocols Collection



Comparison of a public constant with a bit decomposed secret
Input : Public constant R and Boolean sharings $[x_0]_2, \cdots, [x_{m-1}]_2$

Comparison of a secret with another secret
Input : Arithmetic sharings $[x]_M$ and $[y]_M$

Comparison of a secret with a constant
Input : Arithmetic sharing $[x]_M$ and public constant R

Specialized comparison for rings with a specific constant
Input : Π_{LTC} when $M = 2^k$ and $R = 2^{k-1}$

Rabbit Intuition: Observation 1

GOAL: ***Detect when a sum over a particular modulus wraps around and correct for it.***

- Given a function: $\mathsf{LT}(\cdot,\cdot) : \mathbb{Z} \times \mathbb{Z} \to \{0,1\} \subseteq \mathbb{Z} \ : \ \begin{cases} \mathsf{LT}(x,y) = 1 & \text{if } (x < y); \\ \mathsf{LT}(x,y) = 0 & \text{otherwise}, \end{cases}$

- We can compute a modular sum by performing computations over the integers:

$$x + y \bmod M = x + y - M \cdot \mathsf{LT}(x + y \bmod M, x) = x + y - M \cdot \mathsf{LT}(x + y \bmod M, y)$$

Rabbit Intuition: Observation 2

Note that we are after: $x \overset{?}{<} M - B$

- Exploit the commutativity of addition:

$$\begin{array}{ccc} [x]_M & \xrightarrow{+r} & a = [x+r]_M \\ \downarrow{\scriptstyle +B} & & \downarrow{\scriptstyle +B} \\ c = [x+B]_M & \xrightarrow{+r} & b = [x+r+B]_M \end{array}$$

Rabbit Intuition: Observation 2

Note that we are after: $x \overset{?}{<} M - B$

- Exploit the commutativity of addition:

$$
\begin{tikzcd}
{[x]_M} \arrow[r, "+r"] \arrow[d, "+B"] & a = [x+r]_M \arrow[d, "+B"] \\
c = [x+B]_M \arrow[r, "+r"] & b = [x+r+B]_M
\end{tikzcd}
$$

- Combine with Observation 1 (performing modular sums over the integers):

Rabbit Intuition: Observation 2

Note that we are after: $x \overset{?}{<} M - B$

- Exploit the commutativity of addition:

$$\begin{tikzcd}
{[x]_M} \arrow[r, "+r"] \arrow[d, "+B"] & a = [x+r]_M \arrow[d, "+B"] \\
c = [x+B]_M \arrow[r, "+r"] & b = [x+r+B]_M
\end{tikzcd}$$

- Combine with Observation 1 (performing modular sums over the integers):

$$\begin{aligned} b = [a+B] &= a + B - M \cdot \mathsf{LT}(b, B) \\ &= x + r - M \cdot \mathsf{LT}(a, r) + B - M \cdot \mathsf{LT}(b, B) \end{aligned}$$

Rabbit Intuition: Observation 2

Note that we are after: $x \overset{?}{<} M - B$

- Exploit the commutativity of addition:

$$\begin{tikzcd}
{[x]_M} \arrow[r, "+r"] \arrow[d, "+B"] & a = [x+r]_M \arrow[d, "+B"] \\
c = [x+B]_M \arrow[r, "+r"] & b = [x+r+B]_M
\end{tikzcd}$$

- Combine with Observation 1 (performing modular sums over the integers):

$$\begin{aligned} b = [a+B] &= a + B - M \cdot \mathsf{LT}(b,B) \\ &= x + r - M \cdot \mathsf{LT}(a,r) + B - M \cdot \mathsf{LT}(b,B) \end{aligned}$$

$$\begin{aligned} b = [c+r] &= c + r - M \cdot \mathsf{LT}(b,r) \\ &= x + B - M \cdot \mathsf{LT}(c,B) + r - M \cdot \mathsf{LT}(b,r) \end{aligned}$$

Rabbit Intuition: Observation 2

Note that we are after: $x \overset{?}{<} M - B$

- Exploit the commutativity of addition:

$$\begin{tikzcd}
{[x]_M} \arrow[r, "+r"] \arrow[d, "+B"] & a = [x+r]_M \arrow[d, "+B"] \\
c = [x+B]_M \arrow[r, "+r"] & b = [x+r+B]_M
\end{tikzcd}$$

- Combine with Observation 1 (performing modular sums over the integers):

$$\begin{aligned} b = [a+B] &= a + B - M \cdot \mathsf{LT}(b,B) \\ &= x + r - M \cdot \mathsf{LT}(a,r) + B - M \cdot \mathsf{LT}(b,B) \end{aligned}$$

$$\begin{aligned} b = [c+r] &= c + r - M \cdot \mathsf{LT}(b,r) \\ &= x + B - M \cdot \mathsf{LT}(c,B) + r - M \cdot \mathsf{LT}(b,r) \end{aligned}$$

$$\mathsf{LT}(a,r) + \mathsf{LT}(b,B) = \mathsf{LT}(c,B) + \mathsf{LT}(b,r)$$

Rabbit Intuition: Observation 2

Note that we are after: $x \overset{?}{<} M - B$

- Exploit the commutativity of addition:

$$\begin{tikzcd}
[x]_M \arrow[r, "+r"] \arrow[d, "+B"] & a = [x+r]_M \arrow[d, "+B"] \\
c = [x+B]_M \arrow[r, "+r"] & b = [x+r+B]_M
\end{tikzcd}$$

- Combine with Observation 1 (performing modular sums over the integers):

$$\begin{aligned} b = [a+B] &= a + B - M \cdot \mathsf{LT}(b,B) \\ &= x + r - M \cdot \mathsf{LT}(a,r) + B - M \cdot \mathsf{LT}(b,B) \end{aligned}$$

$$\begin{aligned} b = [c+r] &= c + r - M \cdot \mathsf{LT}(b,r) \\ &= x + B - M \cdot \mathsf{LT}(c,B) + r - M \cdot \mathsf{LT}(b,r) \end{aligned}$$

$$\mathsf{LT}(a,r) + \mathsf{LT}(b,B) = \mathsf{LT}(c,B) + \mathsf{LT}(b,r)$$

Rabbit Intuition: Observation 2

Note that we are after: $x \overset{?}{<} M - B$

- Exploit the commutativity of addition:

$$\begin{tikzcd} {[x]_M} \arrow[r, "+r"] \arrow[d, "+B"] & a = [x+r]_M \arrow[d, "+B"] \\ c = [x+B]_M \arrow[r, "+r"] & b = [x+r+B]_M \end{tikzcd}$$

- Combine with Observation 1 (performing modular sums over the integers):

$$b = [a+B] = a + B - M \cdot \mathsf{LT}(b,B)$$
$$= x + r - M \cdot \mathsf{LT}(a,r) + B - M \cdot \mathsf{LT}(b,B)$$

$$b = [c+r] = c + r - M \cdot \mathsf{LT}(b,r)$$
$$= x + B - M \cdot \mathsf{LT}(c,B) + r - M \cdot \mathsf{LT}(b,r)$$

$$\mathsf{LT}(a,r) + \boxed{\mathsf{LT}(b,B)} = \mathsf{LT}(c,B) + \mathsf{LT}(b,r)$$

Rabbit Intuition: Observation 2

Note that we are after: $x \stackrel{?}{<} M - B$

- Exploit the commutativity of addition:

$$
\begin{tikzcd}
{[x]_M} \arrow[rr, "+r"] \arrow[d, "+B"] & & a = [x + r]_M \arrow[d, "+B"] \\
c = [x + B]_M \arrow[rr, "+r"] & & b = [x + r + B]_M
\end{tikzcd}
$$

- Combine with Observation 1 (performing modular sums over the integers):

$$
\begin{aligned}
b = [a + B] &= a + B - M \cdot \mathsf{LT}(b, B) \\
&= x + r - M \cdot \mathsf{LT}(a, r) + B - M \cdot \mathsf{LT}(b, B)
\end{aligned}
$$

$$
\begin{aligned}
b = [c + r] &= c + r - M \cdot \mathsf{LT}(b, r) \\
&= x + B - M \cdot \mathsf{LT}(c, B) + r - M \cdot \mathsf{LT}(b, r)
\end{aligned}
$$

$$\boxed{\mathsf{LT}(a, r)} + \boxed{\mathsf{LT}(b, B)} = \mathsf{LT}(c, B) + \boxed{\mathsf{LT}(b, r)}$$

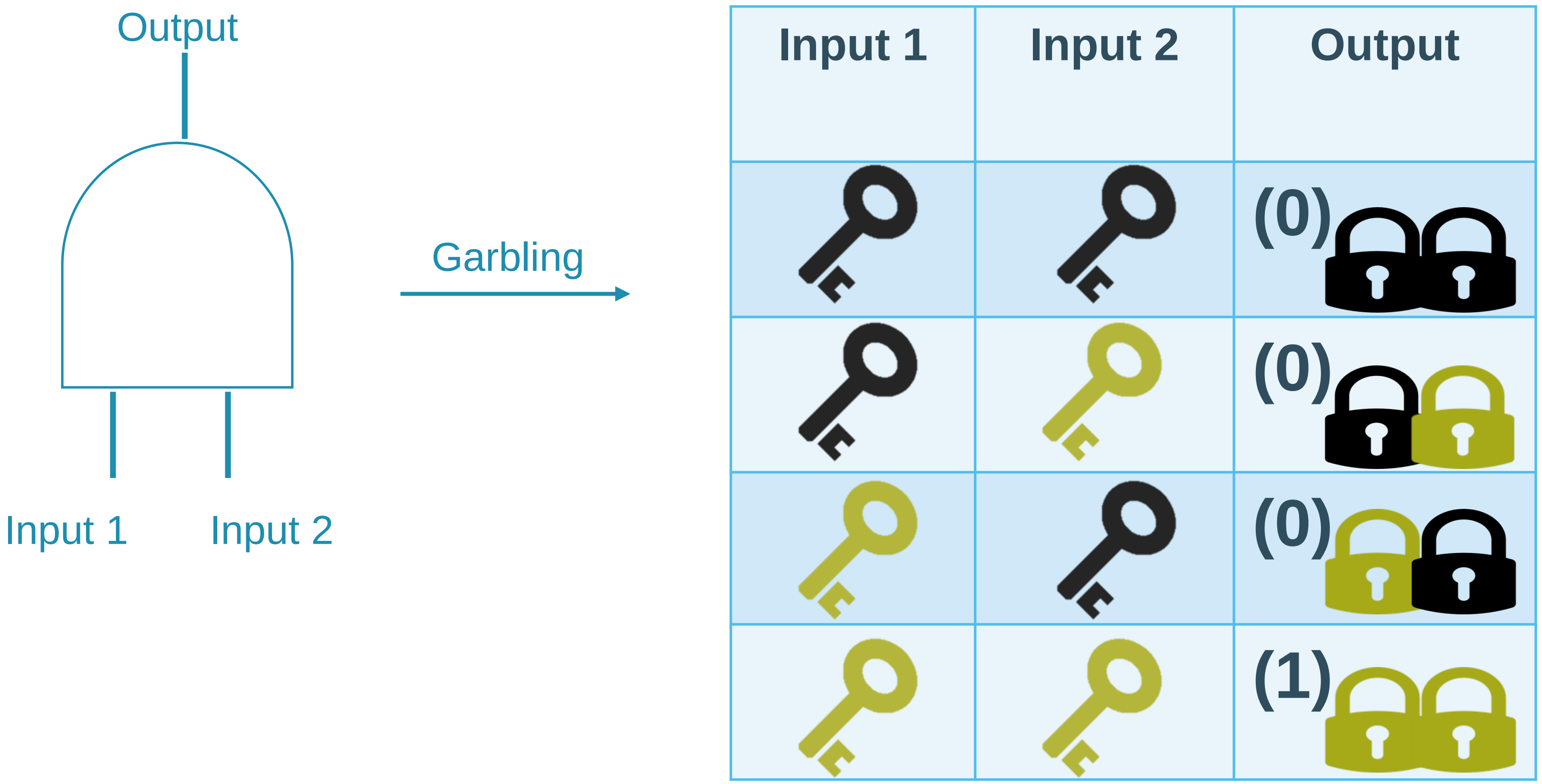
Rabbit's conclusions

- Rabbit comparisons are ***more efficient*** than edaBit[1] comparisons with:
  - ~1.5x better throughput in most adversarial settings
  - Over 2.3x better throughput in the passive, honest majority setting
  - Lower communication cost
  - Lower memory footprint for the HE-based preprocessing
- Rabbit eliminates the need for "slack"
- The core of the Rabbit comparison algorithms is unconditionally secure

1 Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. **Improved primitives for MPC over mixed arithmetic-binary circuits.** In *Annual International Cryptology Conference*, pp. 823-852. Springer, Cham, 2020.

Improving MPC Primitives:
The Case of Multiparty Arithmetic Garbling

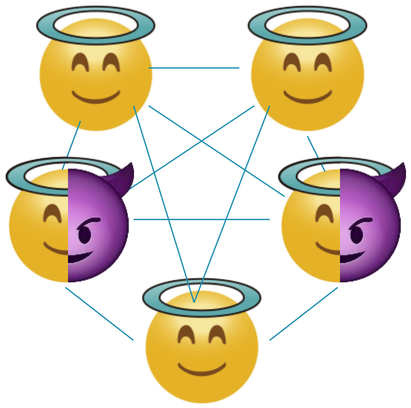
Full-Threshold Actively-Secure Multiparty Arithmetic Circuit Garbling



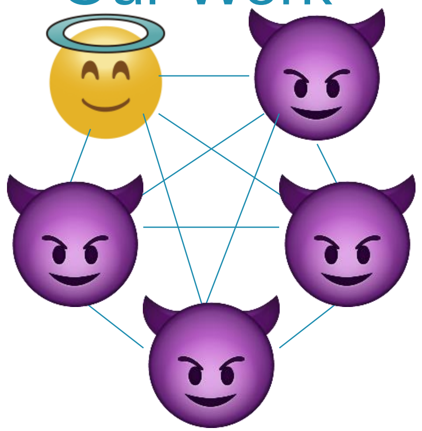
Full-Threshold Actively-Secure Multiparty Arithmetic Circuit Garbling: Contributions

Previous Work

i.



Our Work

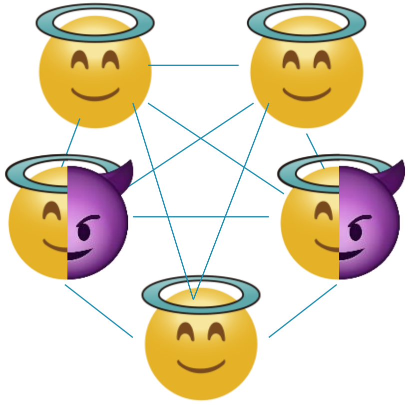


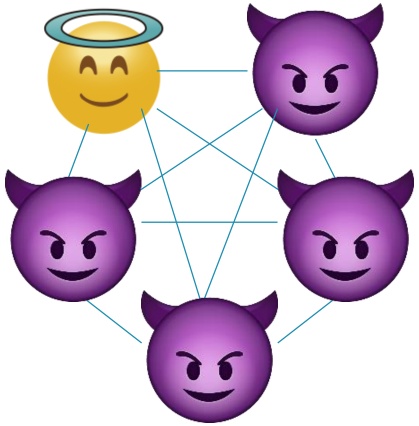
Full-Threshold Actively-Secure Multiparty Arithmetic Circuit Garbling: Contributions

Previous Work

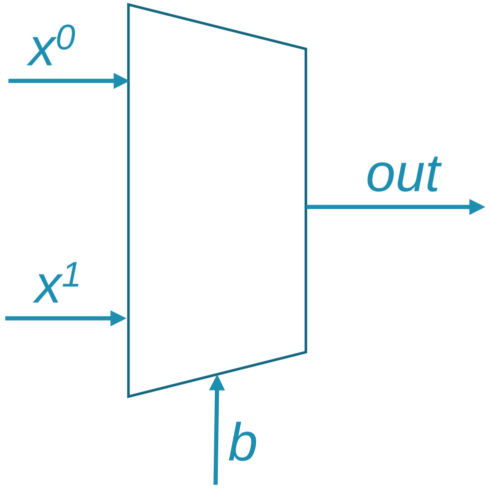
Our Work

i.

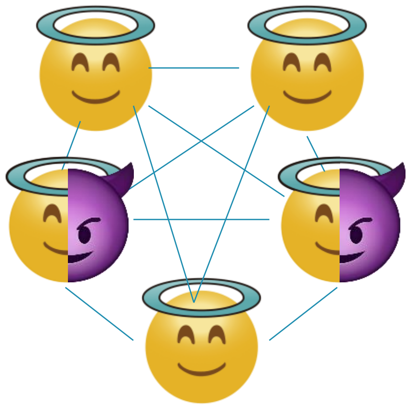
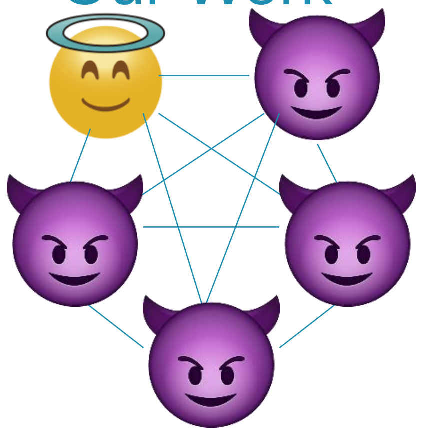




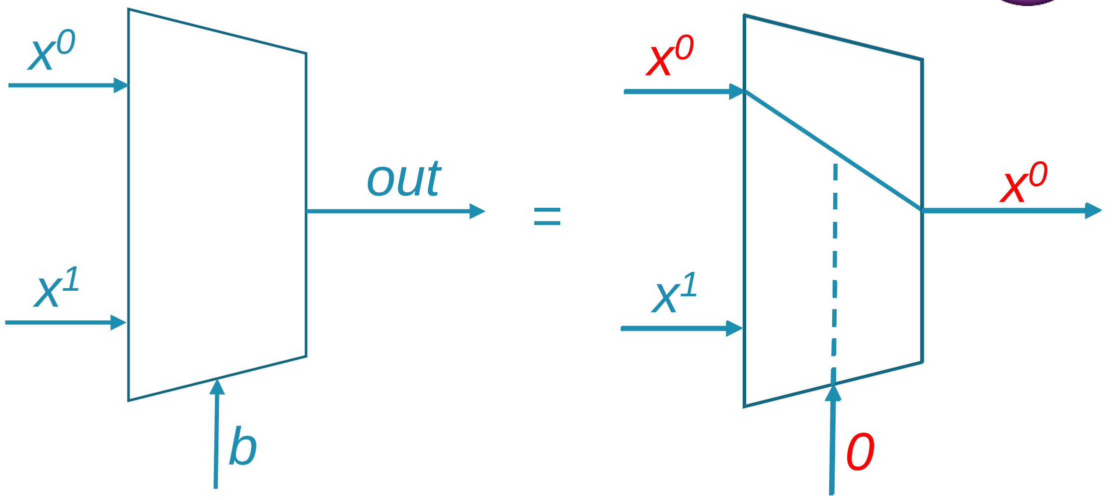
ii.



Full-Threshold Actively-Secure Multiparty Arithmetic Circuit Garbling: Contributions

	Previous Work	Our Work
i.		

ii.

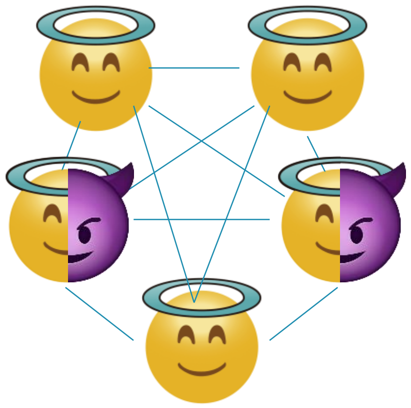


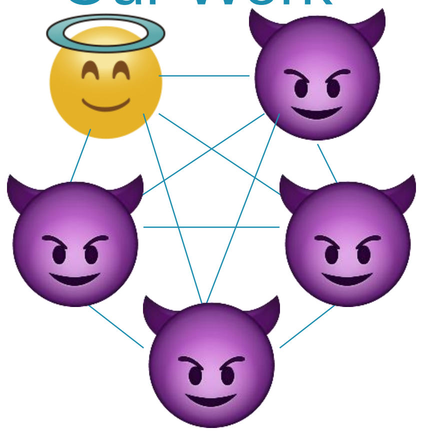
Full-Threshold Actively-Secure Multiparty Arithmetic Circuit Garbling: Contributions

Previous Work

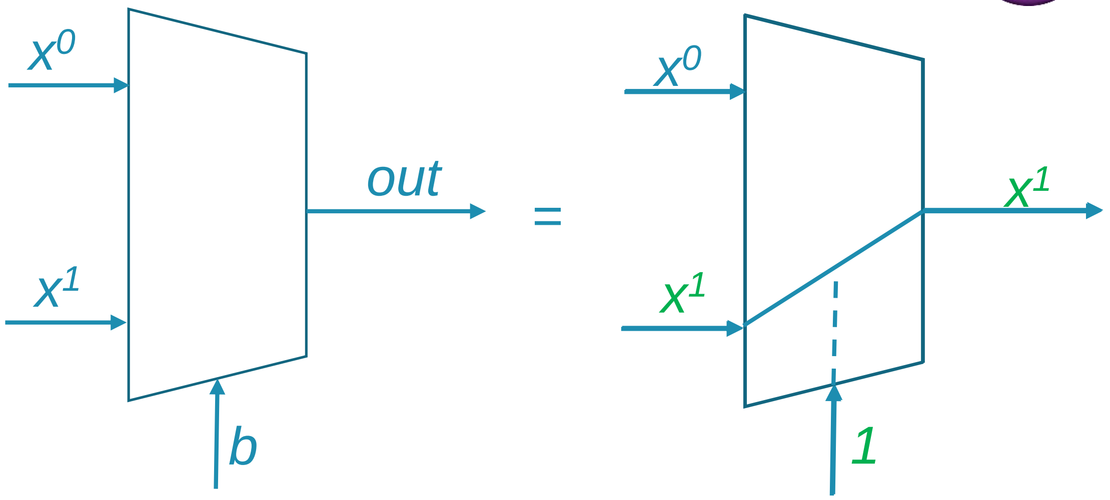
Our Work

i.





ii.

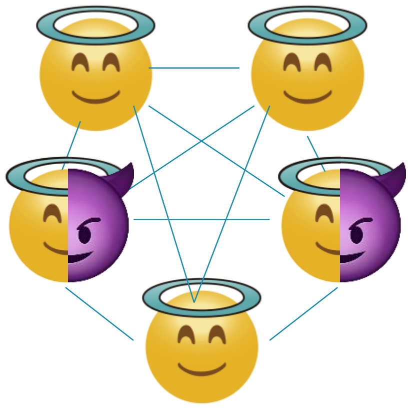


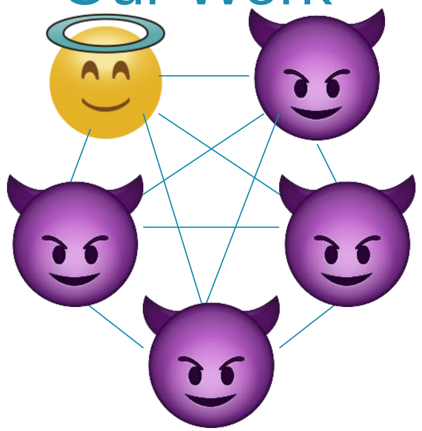
Full-Threshold Actively-Secure Multiparty Arithmetic Circuit Garbling: Contributions

Previous Work

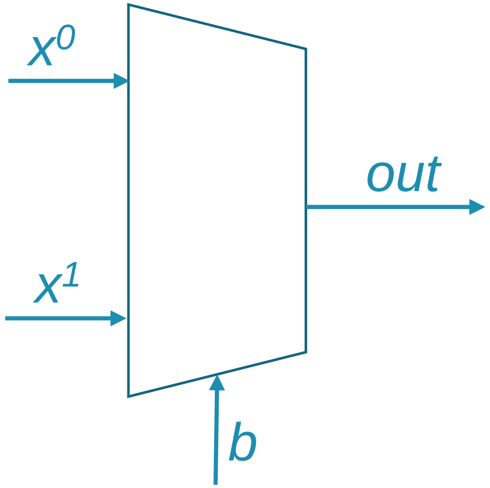
Our Work

i.

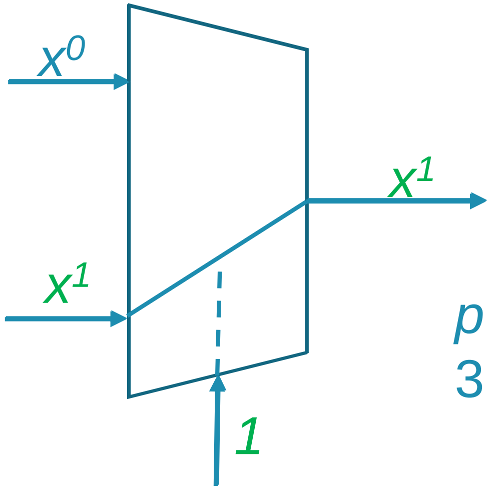




ii.



=



$2p + 2$ ctxt.
2 decryptions

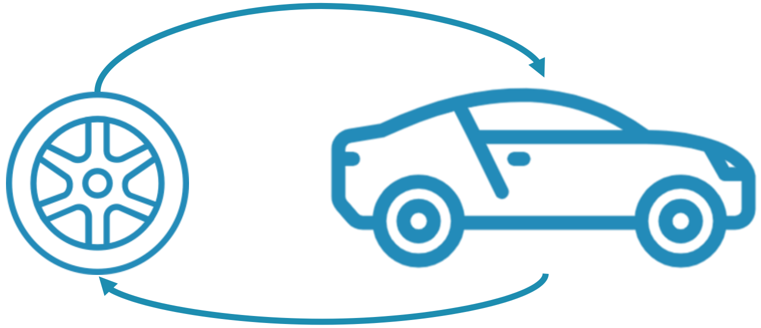
$p + 4$ ctxt.
3 decryptions

Closing Remarks

Conclusions

MPC is practical

Primitives and
Application Scenarios

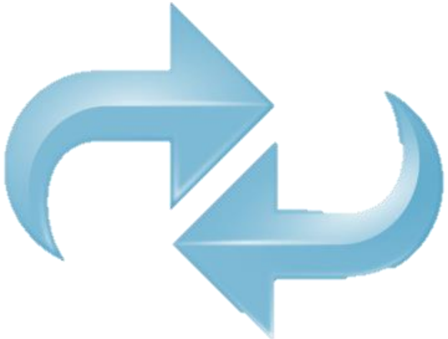


Exploit interdisciplinary
research

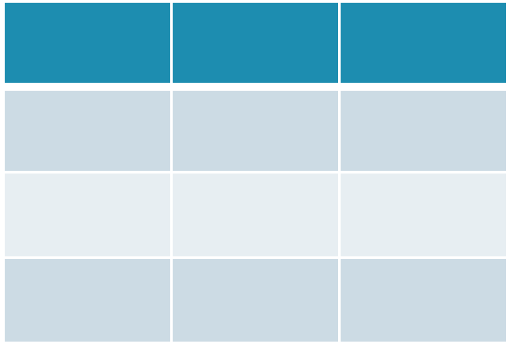
Use each data representation
for what is best

Future Work

Switching Protocols

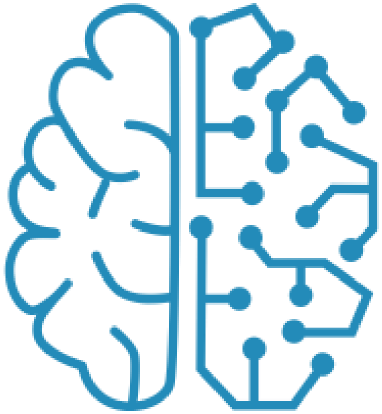


Special Preprocessing



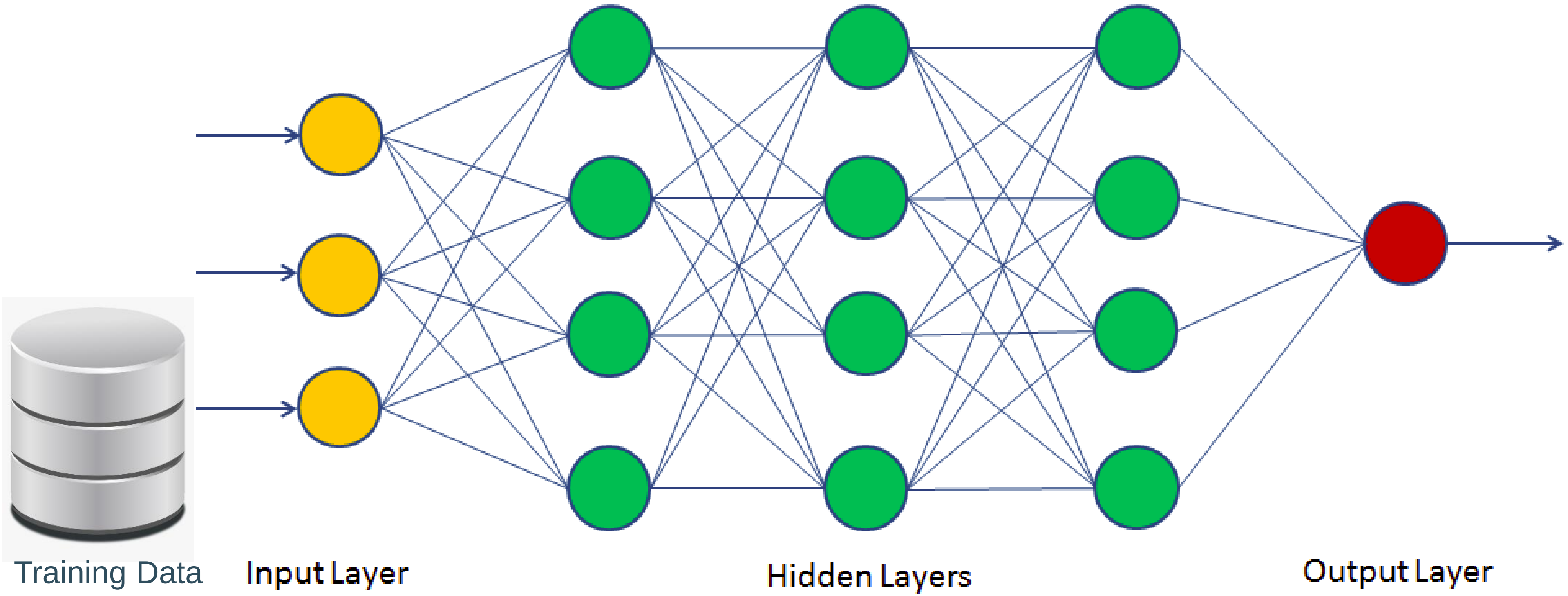
Generalized
Beaver Tuples

MPC for Machine
Learning

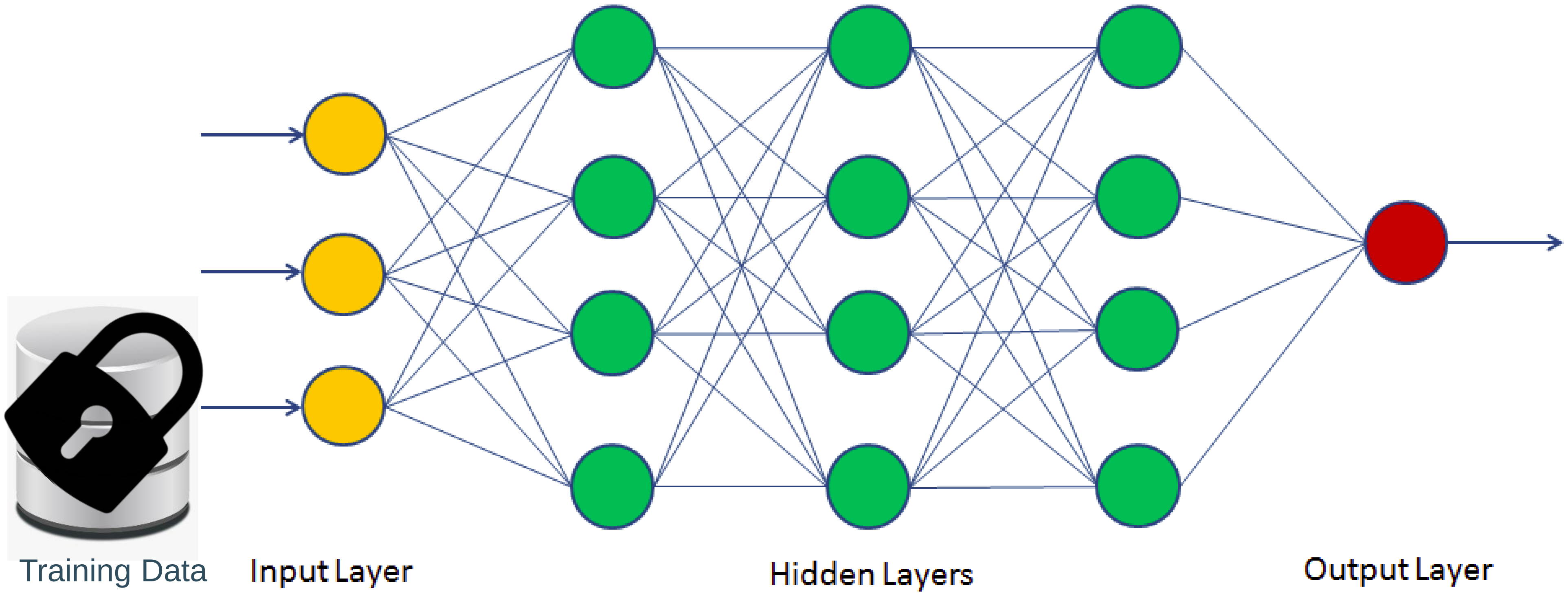


Future Research

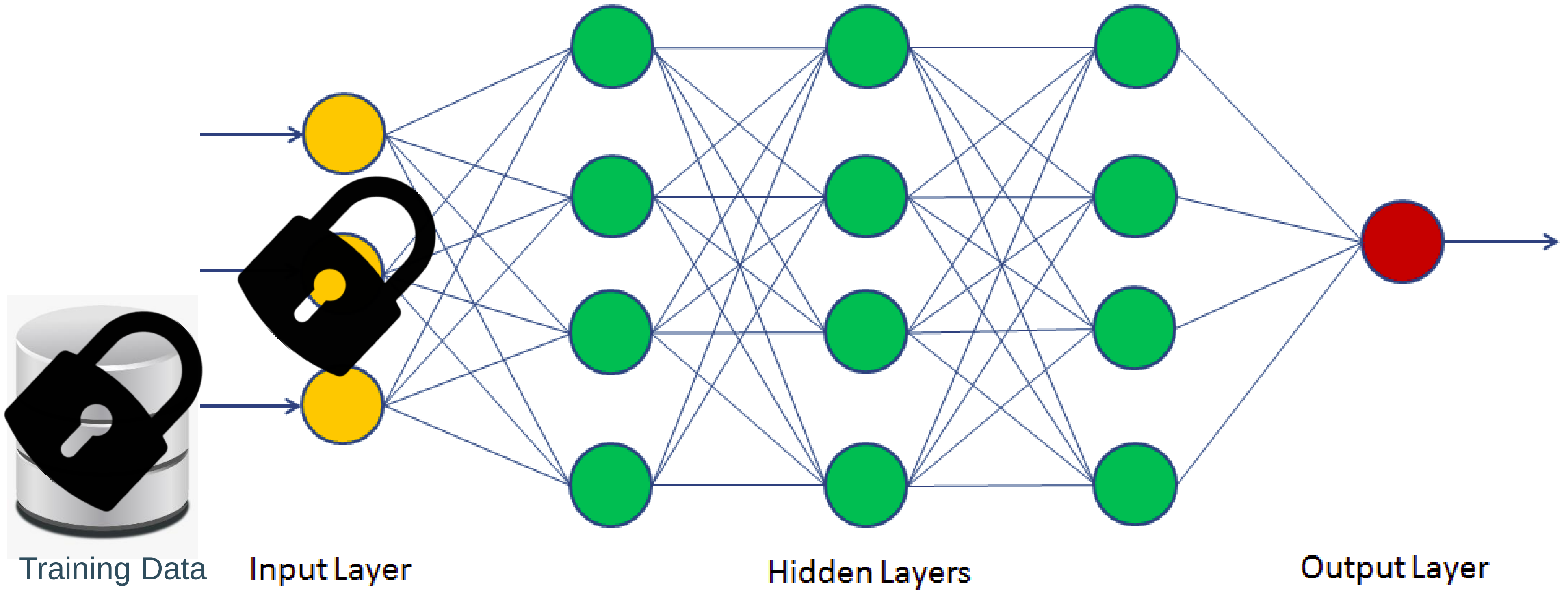
Privacy-Preserving Machine Learning



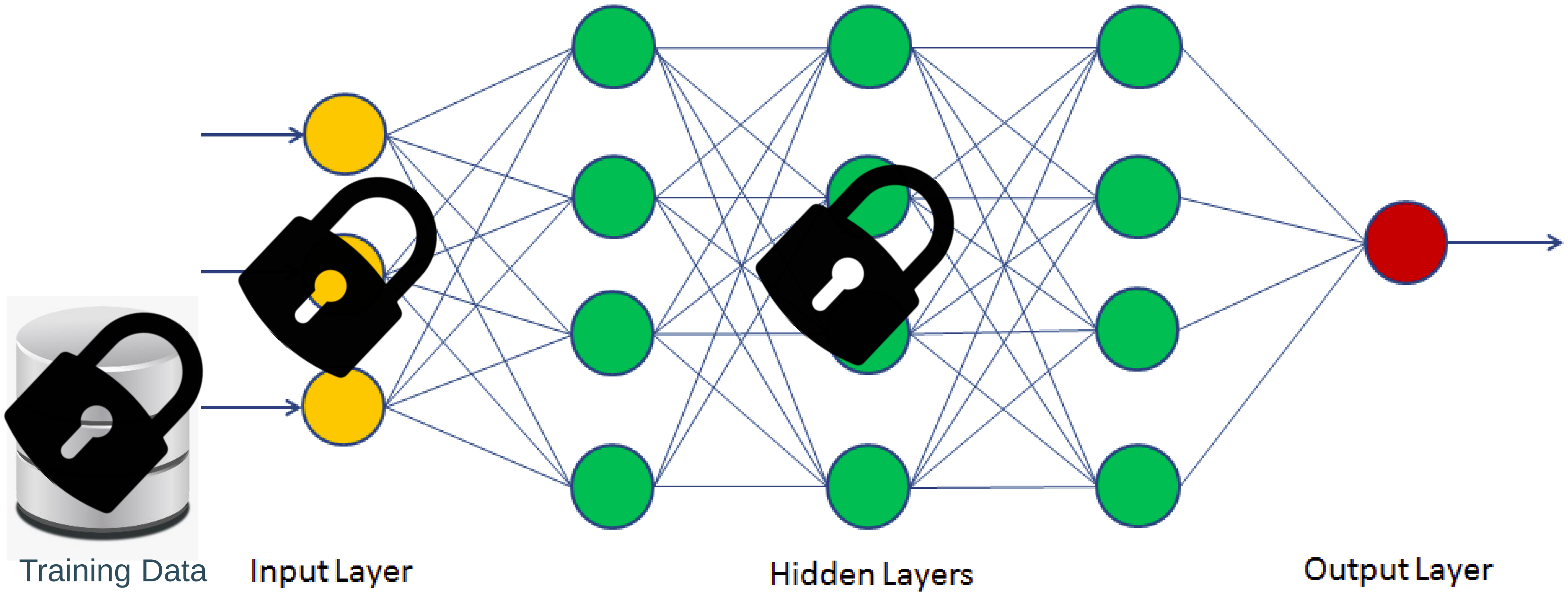
Privacy-Preserving Machine Learning



Privacy-Preserving Machine Learning



Privacy-Preserving Machine Learning



Privacy-Preserving Machine Learning