Privacy-Preserving Systems
(a.k.a., Private Systems)

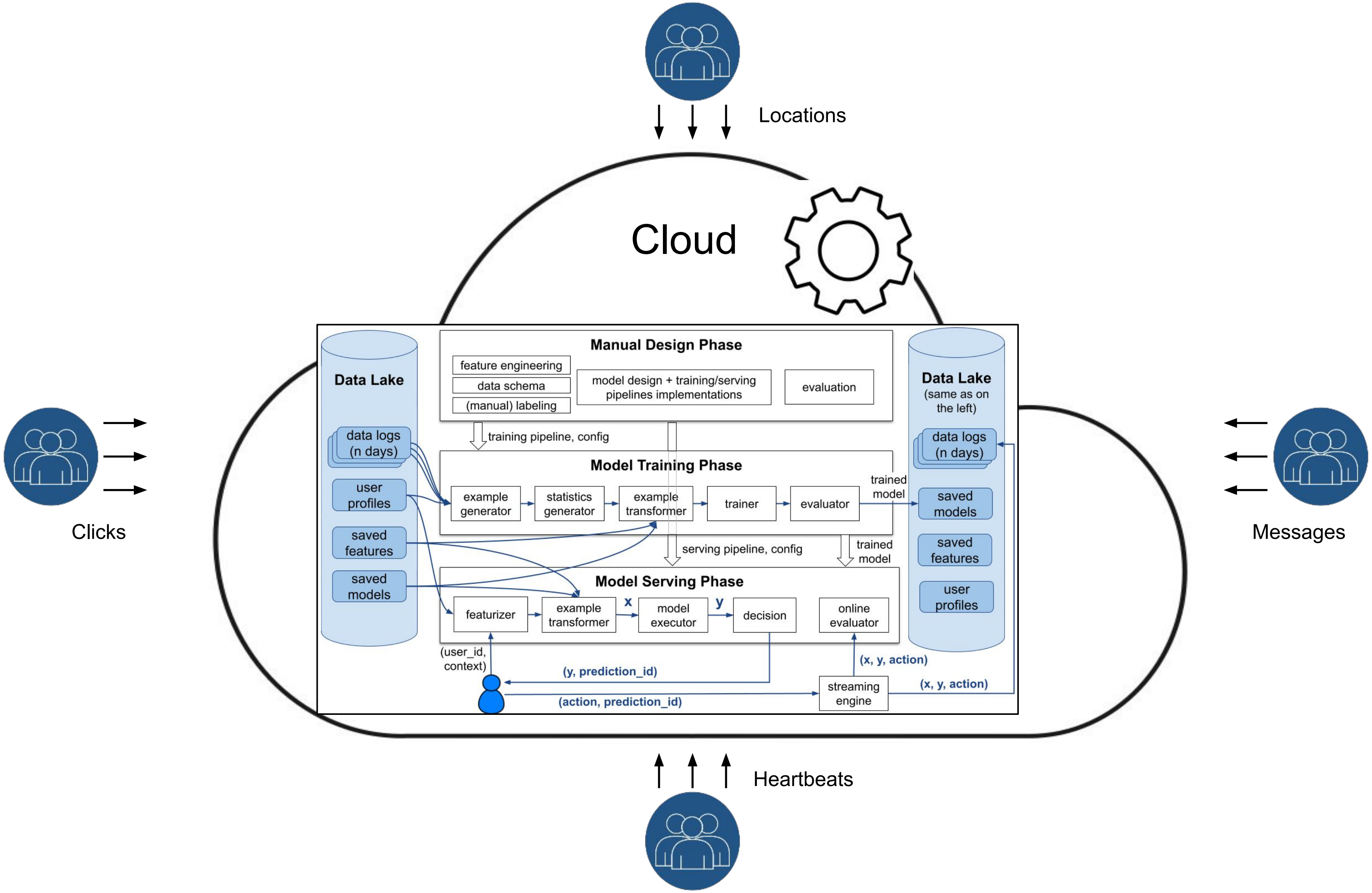
CU Graduate Seminar

Instructor: Roxana Geambasu

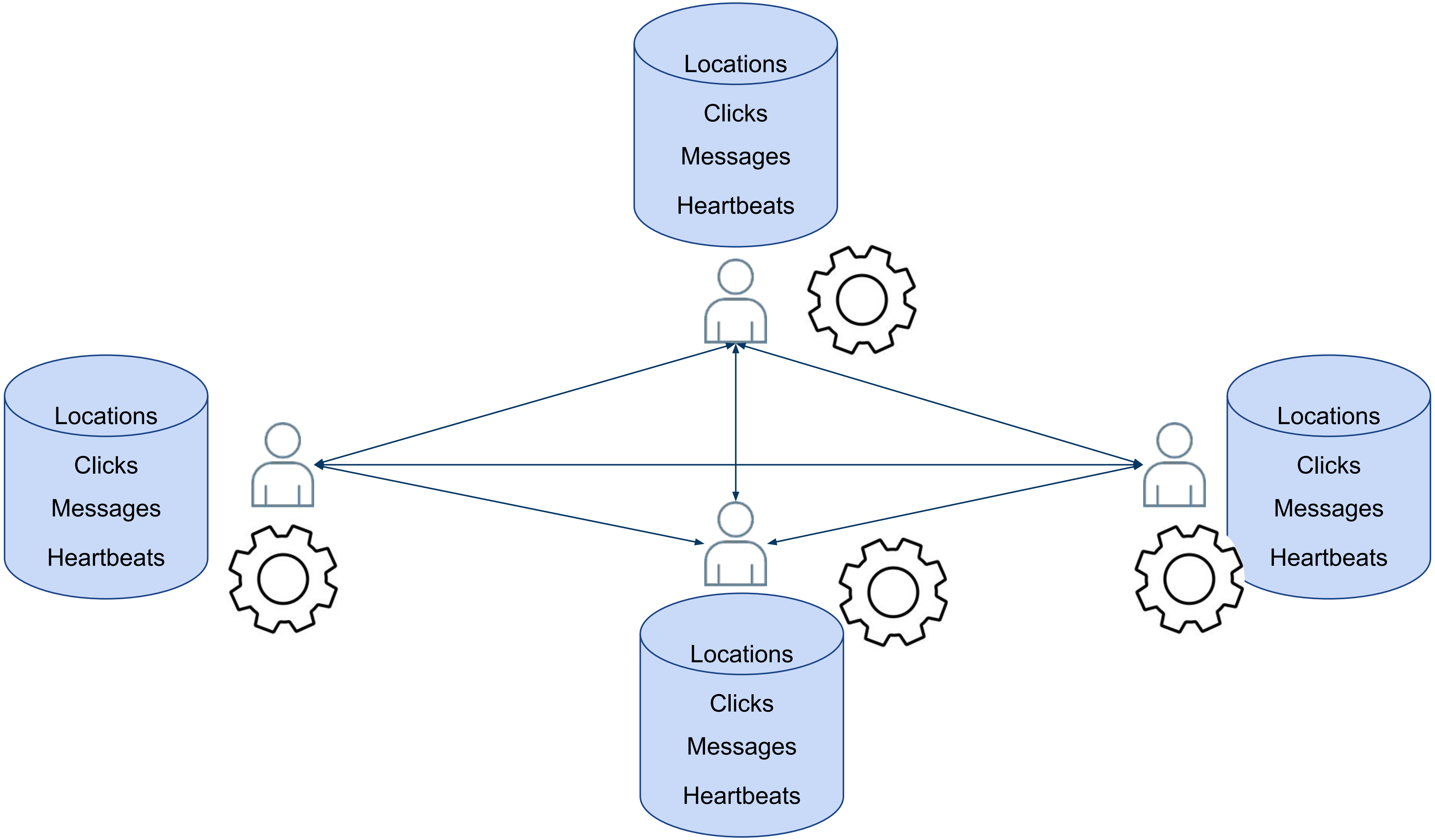
Secure Multiparty Computation

Course Assistant: Pierre Tholoniat

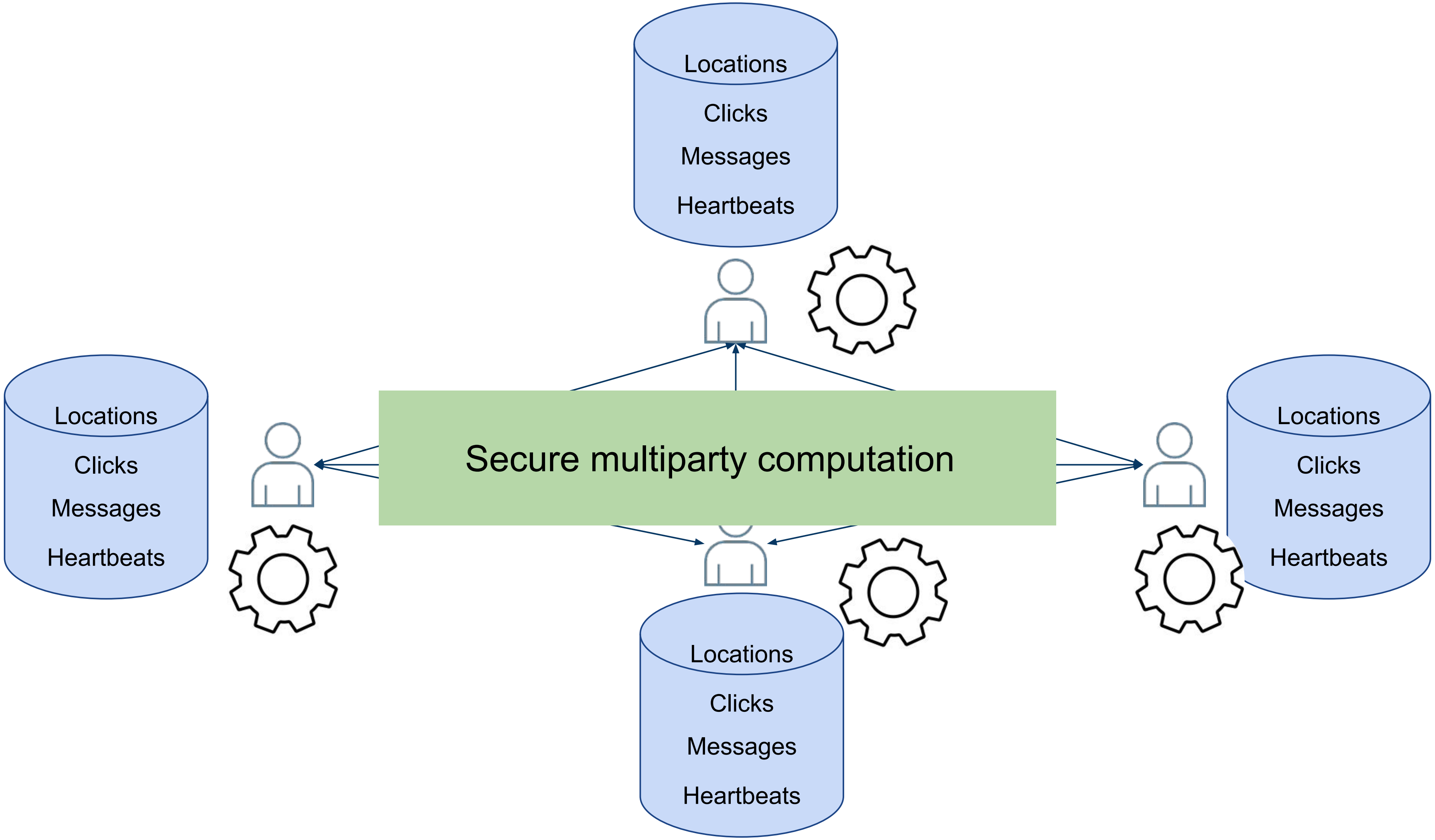
What If No Central Aggregation of Data?



What If No Central Aggregation of Data? (cont.)



What If No Central Aggregation of Data? (cont.)



Case 1: Money Laundering Detection

- Banks want to detect money laundering using machine learning.
- Criminals conceal illegal activities across many banks.
- Banks want to jointly compute a model on customer transaction data, but cannot share data.

Secure Multiparty Computation (MPC)

- Parties emulate a trusted third party via cryptography.
- No party learns any party's input beyond the final result (trained model).
- Performance depends on the number of parties, their computation power, the threat model and the complexity of the computation

(Yao, 1982)

Money Laundering Detection with MPC

- Parties: small number of powerful, interconnected, always-on servers (one for each bank)
- Computation: train a fraud detection model
- Practical today for few parties (say up to 10) and simple computations

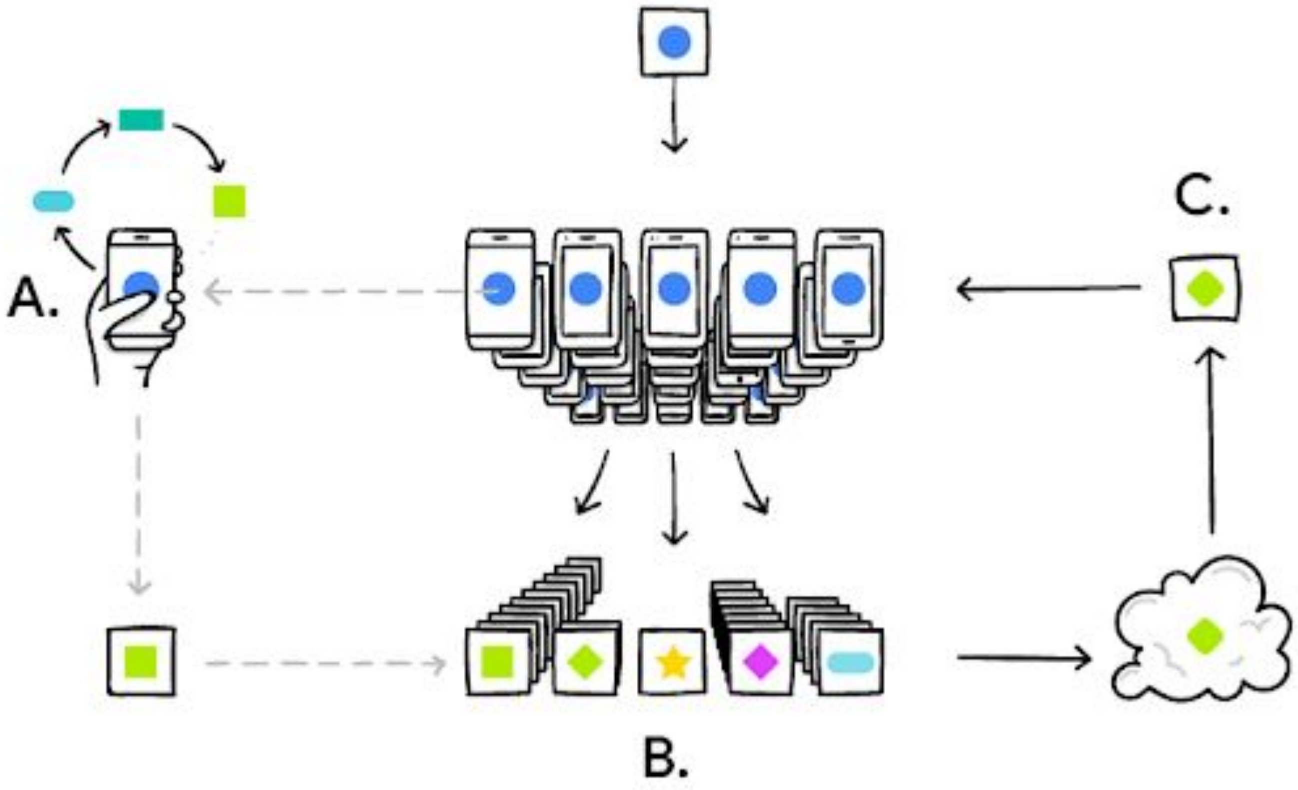
[1]

Case 2: Text Autocomplete

- Want to train a text autocomplete model on many users' data but don't want to collect users' data in a central location.
- Each user trains a local, partial model, and then the cloud combines these models into a global model, which it ships back to the clients.

Federated Learning

- Your phone personalizes the model locally, based on your usage (A)
- Many users' updates are aggregated (B) to form a consensus change (C) to the shared model
- The procedure is repeated as new data becomes available



Credit: [Google AI blog]

Federated Learning with MPC

- Federated learning is a broad term (Kairouz 2021).
  - Can be instantiated in different settings with various combinations of privacy technologies (MPC, differential privacy, secure enclaves)
  - Also involves machine learning and mobile computing considerations
- Secure multiparty computation (MPC) is usually a central building block for federated learning deployments, with specialized MPC protocols such as secure aggregation (Bell 2020)
  - Parties: one powerful central server (untrusted), and many weak clients (of which a certain fraction is untrusted)
  - Computation: aggregate model updates across devices (only sum, not an arbitrarily complex computation!)
  - Practical today. E.g., deployed on millions of Android devices (Xu 2023)

Today’s Plan

- We will consider the general MPC setting
  - Multiple parties with private inputs
  - For simplicity, assume parties are honest-but-curious (i.e. follow the protocol)
  - Compute a function on inputs without revealing anything else than the output
- We’ll sketch how some simple MPC protocols work
  - What is the intuition behind the math?
  - How practical are MPC protocols? What operations are expensive, how do they scale with the number of parties?
  - See the Pragmatic MPC textbook (Evans, 2018) and references for the details
- We’ll look at practical MPC systems and deployments

Outline

1. Shamir Secret Sharing
2. Evaluating Arithmetic Circuits with the BGW Protocol
3. Preprocessing for MPC with Beaver Triples
4. Examples of MPC systems

Outline

1. Shamir Secret Sharing
2. Evaluating Arithmetic Circuits with the BGW Protocol
3. Preprocessing for MPC with Beaver Triples
4. Examples of MPC systems

Shamir Secret Sharing (Shamir, 1979)

Setting:

- n parties, threshold $t \leq n$
- A global secret y $\in$ K := F_p is shared among parties
- Each party i has a share y_i
- Notation for a sharing of y: $[y] := (y_1, \ldots, y_n)$

Desired properties:

- Knowing $k \geq t$ shares is sufficient to reconstruct y
- Knowing $k < t$ shares doesn't reveal anything about y

How can secret-sharing be useful?

Example: secret key recovery

- Split your wallet key into n=5 backups servers
- Reconstruct the key from t servers when needed
  - If t=1, a single corrupted server can steal your key
  - If t=5, a single faulty backup prevents you from recovering your key
  - If t=3, resilient against 2 corrupted colluding servers and 2 failures

We can also use secret-sharing for arbitrary MPC

Construction with polynomials

Lagrange interpolation:

- Fact: the only polynomial of degree ≤ *t-1* with *t* roots or more is zero
- Consequence: any polynomial $P \in K_{t-1}[X]$ is uniquely characterized by the list of coordinate pairs $(P(x_1), \ldots, P(x_t))$ for $(x_1, \ldots, x_t)$ distinct field elements
- Lagrange coefficients:

$$P(X) = \sum_{i=1}^{t} P(x_i) \prod_{j \neq i} \frac{X - x_i}{x_j - x_i}$$

Construction with polynomials

Protocol:

- We (the secret owner/dealer) sample a random polynomial in $K_{t-1}[X]$ such that $P(0) = y$
- Fix public non-zero interpolation points $x_1, \ldots, x_n$
- Distribute $y_i := P(x_i)$ to party $i \in \{1, \ldots, n\}$
- Any group of *t* parties can reconstruct y:

$$y = P(0) = \sum_{i=1}^{t} P(x_i) \prod_{j \neq i} \frac{0 - x_i}{x_j - x_i} = \sum_{i=1}^{t} y_i \lambda_i$$

- The Lagrange coefficients λ_i can be computed in advance, we just need a linear combination of the shares to reconstruct the secret

Outline

1. Shamir Secret Sharing
2. Evaluating Arithmetic Circuits with the BGW Protocol
3. Preprocessing for MPC with Beaver Triples
4. Examples of MPC systems

The BGW Protocol (Ben-Or, 1988)

Can we perform operations on a secret-shared input?

- Example application: split a private key into *n* shares, and sign a document without ever reconstructing the private key locally
- Any computation in F_p can be represented as an arithmetic circuit (why?)
- We just need to have secret-shared version of the + and x gates

Using multiple inputs:

- In the Shamir setting we had a trusted dealer that splits a secret into shares
- The dealer can be a (semi-honest) party that shares its own input with other parties
- We run multiple Shamir sharings in parallel and combine them with gates

Additions are Free

- Two inputs shared with Shamir's scheme:
  - Secret p, polynomial P such that $p = P(0)$, shares $P(x_1), \ldots, P(x_n)$
  - Secret q, polynomial Q such that $q = Q(0)$, shares $Q(x_1), \ldots, Q(x_n)$
- Output:
  - Desired output: $r := p + q = P(0) + Q(0)$
  - $R := P + Q$ is a valid Shamir polynomial (degree $\leq t-1$ and $R(0) = r$)
  - Party i's share is $R(x_i) = P(x_i) + Q(x_i)$
- Parties can construct their share of the output locally, without any interaction!

Problem with Multiplications

- Two inputs shared with Shamir's scheme:
  - Secret p, polynomial P such that $p = P(0)$, shares $P(x_1), \dots, P(x_n)$
  - Secret q, polynomial Q such that $q = Q(0)$, shares $Q(x_1), \dots, Q(x_n)$
  - Desired output: $r := p * q = P(0) * Q(0)$
- Problem:
  - $R := P * Q$ satisfies $R(0) = r$ but has degree $\leq 2(t-1)$, not a valid sharing
  - Since R doesn't work, can we find another polynomial R' with $R'(0) = r$ and degree $\leq t$?

Degree Reduction Trick

Goal: find a polynomial R' with R'(0) = r and degree ≤ t-1

- Observation: with Lagrange's formula, we have $R(0) = \sum_{i=1}^{2t-1} \lambda_i R(x_i)$
- Each party i can create a new Shamir sharing of $R(x_i)$
  - Choose a fresh degree t-1 polynomial R_i such that $R_i(0) = R(x_i)$
  - Distribute $R_i(x_j)$ to party j
- Summing up R_i with public Lagrange coefficients gives us $R' := \sum_{i=1}^{2t-1} \lambda_i R_i$
- R' meets our goal:

$$R(0) = \sum_{i=1}^{2t-1} \lambda_i \left(\sum_{j=1}^{t} \mu_j R_i(x_j) \right) = \sum_{j=1}^{t} \mu_j \left(\sum_{i=1}^{2t-1} \lambda_i R_i \right) (x_j)$$

Cost of Multiplications

- Re-sharing requires all-to-all communication
- We still have security against t-1 corrupt parties. But we also need $2t-1 \leq n$ to reconstruct R(0), so secure under **honest majority**.
  - ($h := n - (t-1) \geq n+1 - (n+1)/2$, i.e. $h > n/2$)
- Corrupt parties are still semi-honest here (a malicious party that re-shares garbage coefficients could completely destroy the output)

Outline

1. Shamir Secret Sharing
2. Evaluating Arithmetic Circuits with the BGW Protocol
3. Preprocessing for MPC with Beaver Triples
4. Examples of MPC systems

MPC with Preprocessing

- BGW multiplications are costly (in terms of interactions)
- We can save time by computing some things in advance
- MPC with preprocessing:
  - Offline phase: a trusted dealer generates input-independent cryptographic material
  - Online phase: parties use the material to save some time (less communication) when evaluating the circuit
- Beaver triples are secret-shared tuples for multiplication

Beaver Triples (Beaver, 1991)

Generation:

1. Take a random tuple (a,b,c) in F_p such that c = a*b
2. Split it and distribute shares to the parties: [a], [b], [c]

Multiplication: we have [x], [y] and want [xy]

1. Each party reveals [x] - [a]. d := x - a is now public
2. Each party reveals [y] - [b]. e: y - b is now public
3. Each party computes locally [xy] = de + d[b] + e[a] + [c]

Beaver Triples

Security:

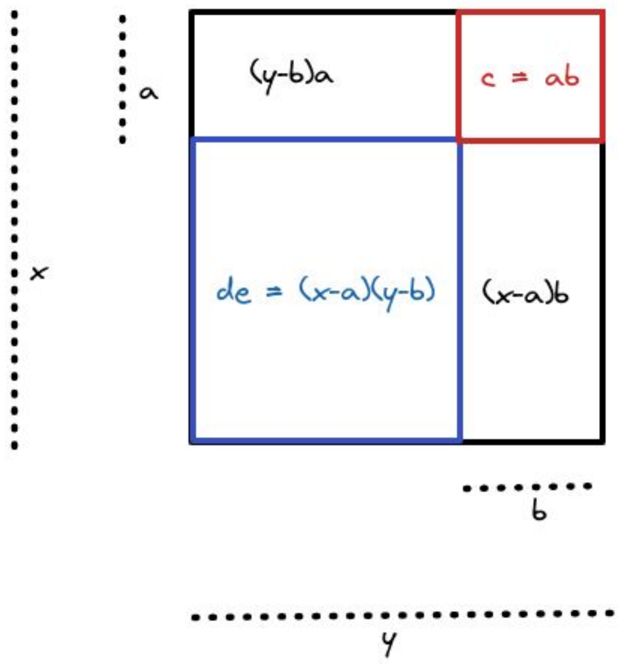
- x - a and y - b are one-time pad encryptions of x and y

Correctness:

$$\sum (de + d[b] + e[a] + [c])$$

$$= (x\text{-}a)(y\text{-}b) + (x\text{-}a)b + (y\text{-}b)a + c$$

$$= xy$$



Beaver Triples in a Circuit

Computational and communication cost:

- Each party just needs to broadcast 2 values ([x] - [a] and [y] - [b])
- In BGW, each party generates a polynomial and sends n values (one for each other party)
- Triples don't depend on the input, and can't be reused, so we need to prepare enough to evaluate the whole circuit
- There are techniques to generate triples in batches

Applicability of Beaver Triples

- Beaver triples work with other types of secret sharing, not just Shamir and BGW
- Information-theoretic security: no computational assumptions
- The trusted dealer can be emulated by the parties themselves, e.g. with HE (Smart, 2019)

Outline

1. Shamir Secret Sharing
2. Evaluating Arithmetic Circuits with the BGW Protocol
3. Preprocessing for MPC with Beaver Triples
4. Examples of MPC systems

Existing Systems and Production Libraries

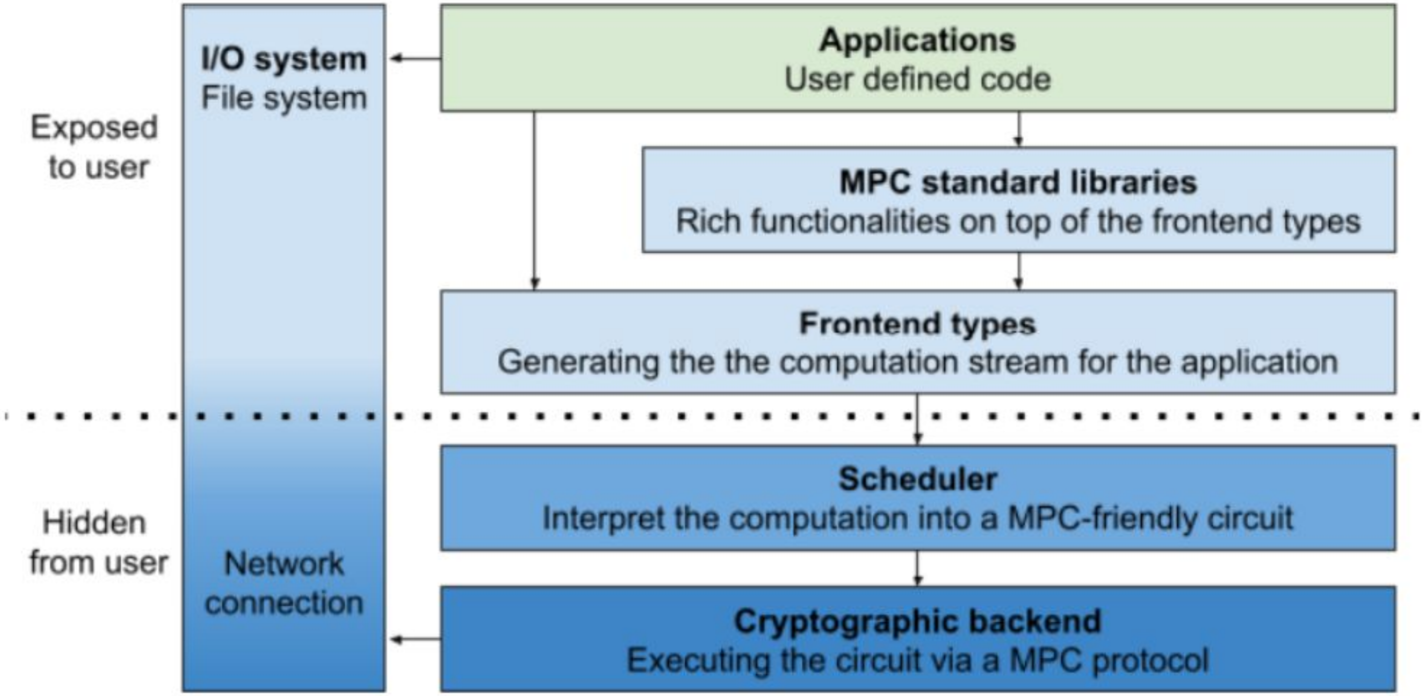
- Generic MPC:
  - Inpher's XOR Secret Computing
  - Meta's Private Computation Framework
- Federated learning:
  - Google's Tensorflow Federated
  - Flower framework: See demo from their docs
- Secure aggregation for simple statistics:
  - Libprio-rs (we'll discuss the Prio protocol next week)

Practical Deployments

- State-of-the-art MPC protocols can be practical:
  - Usually with 2 or 3 *active* parties (e.g., non-colluding cloud providers)
  - But can handle large numbers of *passive* parties (e.g., browsers) who share their input once and let the active parties compute the output
  - Primitives tailored for different use cases
- Examples:
  - AES evaluation on a secret-shared secret key (Damgård, 2010)
  - Distributed aggregation for telemetry or contact tracing (Corrigan-Gibbs, 2017)
  - Training ML models on secret-shared data (Mohassel, 2018)

Deep Dive: Meta's MPC Framework

- General purpose library to build MPC *systems*
- Open-source: https://github.com/facebookresearch/fbpcf
- Architecture from the whitepaper:



Cyptographic Backend and Scheduler

- Boolean circuits instead of arithmetic circuits
  - Inputs are secret-shared bits
  - AND and XOR instead of + and x
  - Easier to manipulate and compile programs
- Cryptographic primitives:
  - GMW secret sharing, a different scheme than BGW tailored for F_2 and resilient against up to n-1 corrupt parties (while BGW needs an honest majority)
  - Preprocessed Beaver triples to speed up AND gates
  - https://github.com/facebookresearch/fbpcf/blob/main/fbpcf/engine/SecretShareEngine.cpp
- Scheduler:
  - Keep track of intermediate results
  - Order gates and execute them
  - Supports multithreading

C++ Types and Operators

- Frontend types: special C++ types for Bit, Int, BitString
- Everything is reduced to bitwise operations (gates)
- Gates are passed to the scheduler
- Example: integer comparison. https://github.com/facebookresearch/fbpcf/blob/b38024cccc79dff74bbce3fbbf9836caf80a4ce7/fbpcf/frontend/Int_impl.h#L186

Example Application

- The millionaire game:
  - Alice and Bob each have one secret input (their wealth)
  - The output of the circuit is one single bit: who is the richest
  - Parties shouldn’t learn anything else than the output
- https://github.com/facebookresearch/fbpcf/blob/main/example/millionaire/MillionaireGame.h
- Deployment: TCP socket communication, parties can run in Docker

Conclusion

- Secure multiparty computation (MPC) allows parties to jointly compute an output without revealing their input or intermediary results
- We saw basic MPC techniques (secret sharing, circuit evaluation, preprocessing) in a simple setting (honest-but-curious adversary and information-theoretic security)
- Different computation/communication tradeoff than fully homomorphic encryption: local computations are lightweight, but parties need to communicate often.
- MPC is already practical and deployed for specific use cases today

Going Further

There are many other important concepts we didn’t cover. Some keywords:

- **Malicious security**: when parties can deviate from the protocol, instead of being simply honest-but-curious. We can adapt honest-but-curious protocols with MACs, ZK proofs and other techniques (e.g. see the SPDZ family of protocols and its modern implementations, Keller 2020).
- **Oblivious transfer (OT)**: a useful primitive where a receiver privately picks one of two secrets offered by a sender.
- **Garbled circuits**: evaluate circuits in constant number of rounds (BGW’s number of rounds is proportional to the depth of the circuit).
- **FHE and Homomorphic Secret Sharing**: other ways of achieving MPC.
- **Oblivious RAM (ORAM):** hide data access patterns efficiently.

References

Yao, Andrew C. "Protocols for secure computations." 23rd annual symposium on foundations of computer science (sfcs 1982). IEEE, 1982.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... & Zhao, S. (2021). Advances and open problems in federated learning. Foundations and trends® in machine learning, 14(1–2), 1-210.

Bell, J. H., Bonawitz, K. A., Gascón, A., Lepoint, T., & Raykova, M. (2020, October). Secure single-server aggregation with (poly) logarithmic overhead. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (pp. 1253-1269).

Xu, Z., Zhang, Y., Andrew, G., Choquette-Choo, C. A., Kairouz, P., McMahan, H. B., ... & Zhang, Y. (2023). Federated learning of gboard language models with differential privacy. arXiv preprint arXiv:2305.18465.

References

D. Evans, V. Kolesnikov, and M. Rosulek, “A Pragmatic Introduction to Secure Multi-Party Computation,” SEC, vol. 2, no. 2–3, pp. 70–246, Dec. 2018, doi: 10.1561/3300000019.

“Private Computation Framework 2.0 - Meta Research,” Meta Research. https://research.facebook.com/publications/private-computation-framework-2-0/ (accessed Mar. 08, 2023).

N. P. Smart and T. Tanguy, “TaaS: Commodity MPC via Triples-as-a-Service,” in Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop, New York, NY, USA, Nov. 2019, pp. 105–116. doi: 10.1145/3338466.3358918.

M. Keller, “MP-SPDZ: A Versatile Framework for Multi-Party Computation,” in Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, Nov. 2020, pp. 1575–1590. doi: 10.1145/3372297.3417872.

I. Damgård and M. Keller, “Secure Multiparty AES: (Short Paper),” in Financial Cryptography and Data Security, vol. 6052, R. Sion, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 367–374. doi: 10.1007/978-3-642-14577-3_31.

References

P. Mohassel and P. Rindal, “ABY3: A Mixed Protocol Framework for Machine Learning,” in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, Oct. 2018, pp. 35–52. doi: 10.1145/3243734.3243760.

H. Corrigan-Gibbs and D. Boneh, “Prio: Private, Robust, and Scalable Computation of Aggregate Statistics,” presented at the 14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17), 2017, pp. 259–282. Accessed: Dec. 15, 2020. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/corrigan-gibbs

A. Shamir, “How to share a secret,” Commun. ACM, vol. 22, no. 11, pp. 612–613, Nov. 1979, doi: 10.1145/359168.359176.

M. Ben-Or, S. Goldwasser, and A. Wigderson, “Completeness theorems for non-cryptographic fault-tolerant distributed computation,” in Proceedings of the twentieth annual ACM symposium on Theory of computing, New York, NY, USA, Jan. 1988, pp. 1–10. doi: 10.1145/62212.62213.

D. Beaver, “Efficient Multiparty Protocols Using Circuit Randomization,” in Advances in Cryptology — CRYPTO ’91, vol. 576, J. Feigenbaum, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 420–432. doi: 10.1007/3-540-46766-1_34.

Secure Multiparty Computation

The End