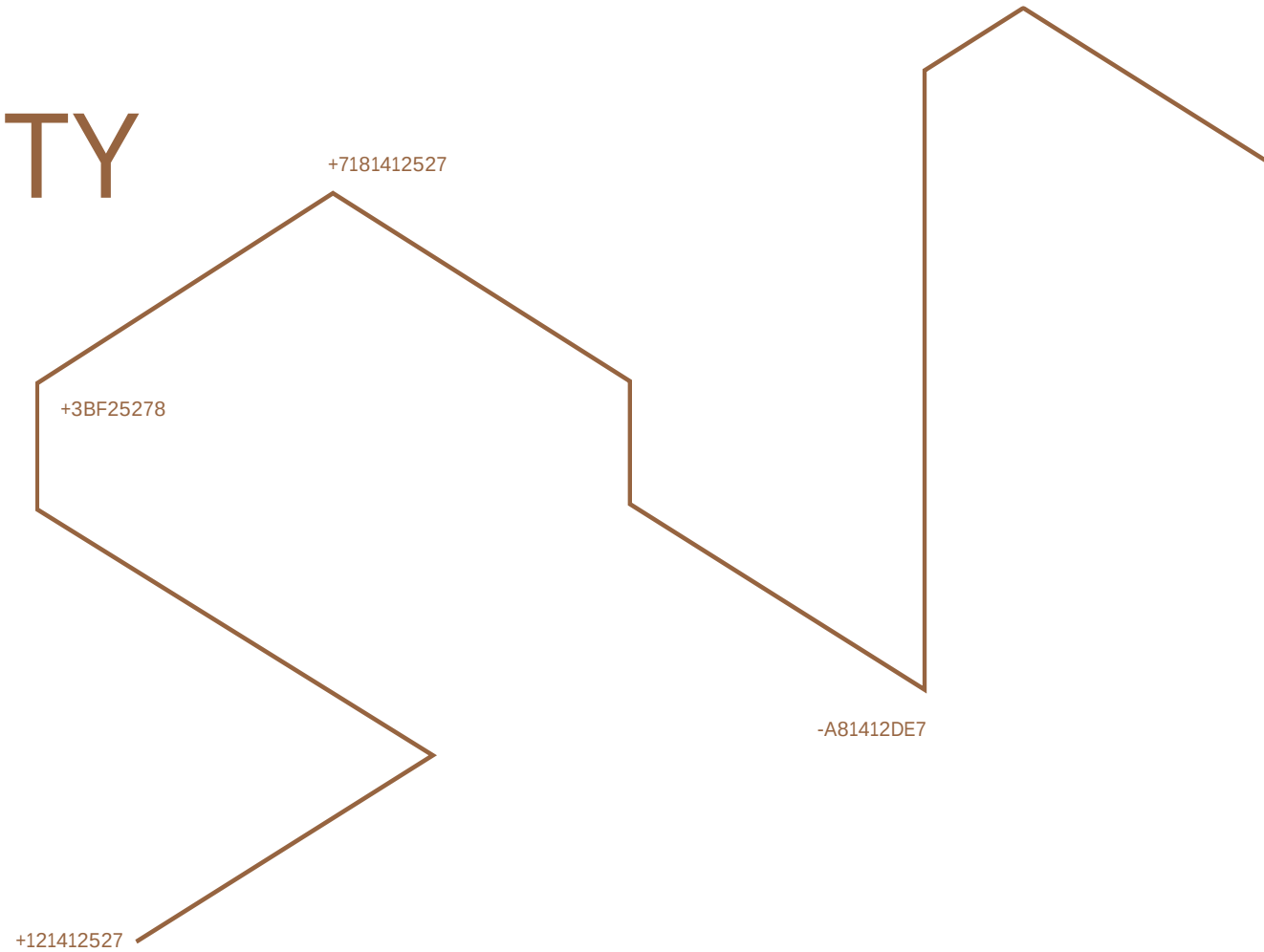
accu 2023

INTRODUCTION TO SECURE MULTI-PARTY COMPUTATION

AHTO TRUU

guardtime

INTRODUCTION TO SECURE MULTI-PARTY COMPUTATION

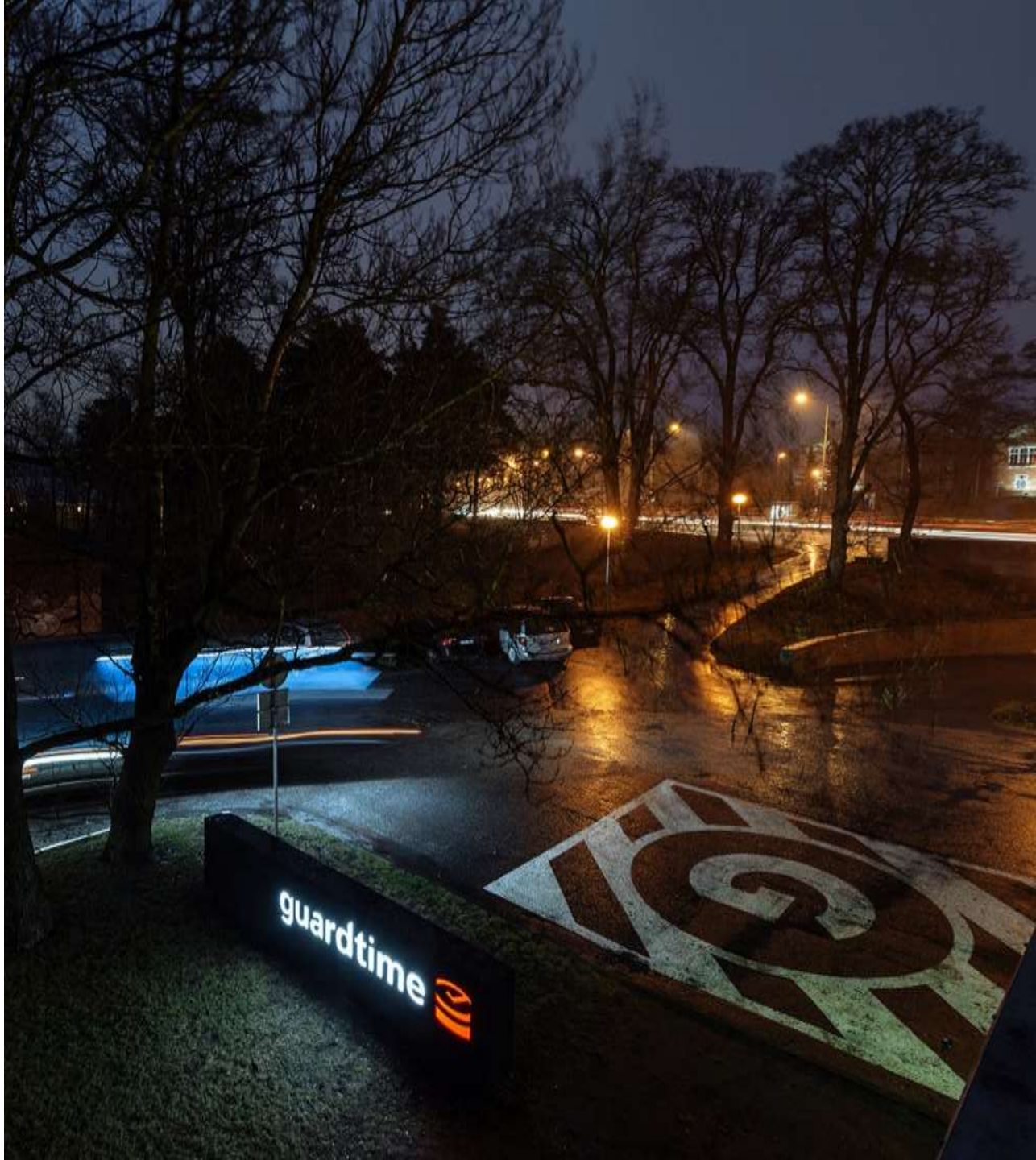


AHTO TRUU
SOFTWARE ARCHITECT, GUARDTIME

ACCU CONFERENCE, 21-APR-2023

ABOUT GUARDTIME

+ Systems engineering company focusing on data security solutions
+ Founded in 2007 in Tallinn, Estonia
+ Global HQ in Lausanne, Switzerland
+ Offices in EU, US, Middle East
+ 150 employees
+ 80% engineers and researchers
+ https://guardtime.com/



SECURE MULTI-PARTY COMPUTATION: WHAT?

- Multiple parties each have some confidential data
- Need to jointly compute a function on these as inputs
- Without revealing anything about the inputs
  - Anything beyond what can be inferred from the output

SECURE MULTI-PARTY COMPUTATION: WHY?

- Classical example (Yao, 1982):
  - Two millionaires want to find out who is richer, but without revealing their actual wealth
- A more practical problem:
  - Tracking the stock levels of medicines across multiple wholesalers in a country

DN DEBATT

Sjukvård.

”Patientsäkerheten hotas när allt fler läkemedel restnoteras”

Antalet restnoterade läkemedel har ökat sedan apotekets monopol upphörde 2009. Tiotusentals svenskar kan inte längre hämta ut förskriven medicin, med risk för allvarliga hälsoeffekter som följd. Kraven på läkemedelsföretagen bör skärpas för att säkerställa att läkemedel finns tillgängliga, skriver överläkarna **Jan Calissendorff** och **Mikael Lehtihet**.

DN. DEBATT 2015/09

En annan patient reste till Uppsala för att besöka apoteken där i sin jakt på läkemedel. En tredje åkte från Stockholm till Hudiksvall där apoteket uppgivit att det efterfrågade läkemedlet fanns.

Våra patienter ska kunna lita på att läkemedel fungerar bra och är säkra, och att de kan levereras i rimlig tid.

SECURE MULTI-PARTY COMPUTATION: HOW?

- Low-tech solution:
  - Find a trusted external party
  - Send all the inputs to that party
  - Have that party run the computation and announce the results
- But what if no such trusted party is available?
  - Homomorphic encryption
  - Garbled circuits
  - Secret sharing

HOMOMORPHIC ENCRYPTION

- Encryption: Enc(k, m) → c, Dec(k, c) → m
- Homomorphic: Add(Enc(k, m_1), Enc(k, m_2)) → Enc(k, m_1+m_2)
- Partially homomorphic: addition or multiplication
- Fully homomorphic: addition and multiplication

GARBLED CIRCUITS

- Function represented as a Boolean circuit
- First party generates encrypted values for 0 and 1 on each wire
- Circuit transferred to second party
- Encryptions of first party's inputs transferred to second party
- Encryptions of second party's inputs transferred obliviously
- Second party evaluates the circuit and reveals the encrypted output
- First party decrypts the result

SECRET SHARING

- Each input split into shares
- Shares distributed among parties
- Computation protocol yields shares of the result
- Shares of the result combined to reveal it

SECURITY LEVELS AND MODELS

- Computational security
  - Breaking computationally infeasible
- Information-theoretic security
  - Secure against any amount of computational power

- Passive adversary
  - Observes, but does not interfere
- Active adversary
  - May forge and suppress messages

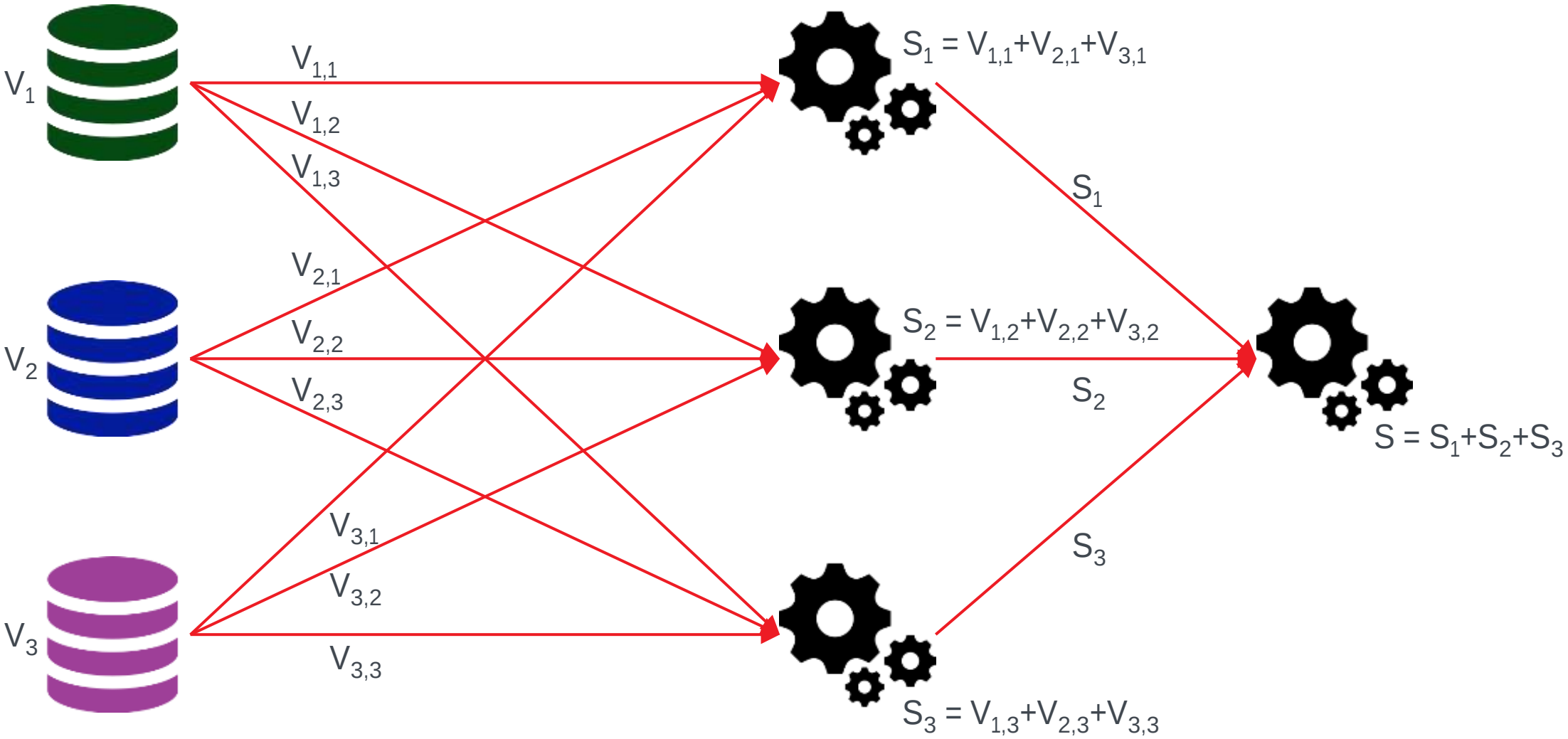
ADDITIVE SECRET SHARING

- To split value V into N shares:
  - Pick V_1, …, V_{N-1} randomly from 0…M-1
  - Pick V_N such that $(V_1+V_2+…+V_N) \bmod M = V$
  - Only works for V < M, so must use sufficiently large M
- Information-theoretically secure with any set of K < N shares

ADDING WITH ADDITIVE SECRET SHARING

- Shares of V_1 and V_2:
  - $V_1 = V_{1,1}+V_{1,2}+V_{1,3}$; $V_2 = V_{2,1}+V_{2,2}+V_{2,3}$
- Adding component-wise:
  - $S_1 = V_{1,1}+V_{2,1}$; $S_2 = V_{1,2}+V_{2,2}$; $S_3 = V_{1,3}+V_{2,3}$
- Recovering sum
  - $S_1+ S_2+ S_3 = (V_{1,1}+V_{2,1}) + (V_{1,2}+V_{2,2}) + (V_{1,2}+V_{2,2}) =$
    $(V_{1,1}+V_{1,2}+V_{1,3}) + (V_{2,1}+V_{2,2}+V_{2,3}) = V_1+V_2$

SUMMING WITH ADDITIVE SECRET SHARING



MULTIPLYING WITH ADDITIVE SECRET SHARING

- Shares of X and Y: $X = X_1+X_2+X_3$, $Y = Y_1+Y_2+Y_3$
- Product $Z = XY = (X_1+X_2+X_3)(Y_1+Y_2+Y_3) =$

$$X_1Y_1+X_1Y_2+X_1Y_3 +$$
$$X_2Y_1+X_2Y_2+X_2Y_3 +$$
$$X_3Y_1+X_3Y_2+X_3Y_3$$

MULTIPLYING (SHAREMIND)

- Resharing of X as X'
  - P_1 generates $r_{12} \rightarrow P_2$; P_2: $r_{23} \rightarrow P_3$; P_3: $r_{31} \rightarrow P_1$
  - P_1 computes $X'_1 = X_1+r_{12}-r_{31}$; P_2: $X'_2 = X_2+r_{23}-r_{12}$; P_3: $X'_3 = X_3+r_{31}-r_{23}$
  - Now $X' = X'_1+X'_2+X'_3 = (X_1+r_{12}-r_{31}) + (X_2+r_{23}-r_{12}) + (X_3+r_{31}-r_{23}) = X$
- Multiplication XY
  - Reshare X as X', Y as Y'
  - P_1 sends $X'_1, Y'_1 \rightarrow P_2$; P_2: $X'_2, Y'_2 \rightarrow P_3$; P_3: $X'_3, Y'_3 \rightarrow P_1$
  - P_1 computes $Z'_1 = X'_1Y'_1+X'_1Y'_3+X'_3Y'_1$; P_2: $Z'_2 = X'_2Y'_2+X'_2Y'_1+X'_1Y'_2$; P_3: $Z'_3 = X'_3Y'_3+X'_3Y'_2+X'_2Y'_3$
  - Now $Z' = Z'_1+ Z'_2+ Z'_3 = X'Y' = XY$
  - Reshare Z' as Z

SHAREMIND

- Product of Cybernetica: https://sharemind.cyber.ee/
- 3-node multi-party computation protocol
- Programmed in SecreC: an MPC-enhanced dialect of C

```
1  void main() {
2    uint64 threshold = arguments("threshold"); // Arguments can be public
3    pd_shared3p uint64[[1]] values = arguments("values"); // ...or private!
4
5    // Computation results are also private.
6    // SIMD-style operations are preferred for parallelisation.
7    pd_shared3p bool[[1]] result = values <= threshold;
8
9    // Results may be published to the client,
10   // computation nodes do not learn these values.
11   publish(result, "result");
12 }
```

SPDZ

- Another protocol based on additive secret sharing
- Improves multiplication speed by pre-computing
- Adds more cryptography, obtains active security
- Original paper: https://eprint.iacr.org/2011/535
- FRESCO: https://github.com/aicis/fresco
- MP-SPDZ: https://github.com/data61/MP-SPDZ
- SCALE-MAMBA: https://github.com/KULeuven-COSIC/SCALE-MAMBA

SHAMIR SECRET SHARING

- To split value V into N shares:
  - Pick a polynomial $P(x) = A_{K-1} \cdot x^{K-1} + A_{K-2} \cdot x^{K-2} + \ldots + A_1 \cdot x + V$
  - Distribute P(1), P(2), ..., P(N) as shares
- Any K shares can be used to recover P and compute V = P(0)
  - K linear equations on K unknowns
  - More efficient recovery of V via Lagrange interpolation
    - $V = \sum_{i=1..k} y_i \prod_{j=1..i-1,i+1..k} \frac{x_j}{x_j - x_i}$
- Provides redundancy
  - Recovery possible even with some shares missing
  - Error checking and recovery with >K shares available

COMPUTING WITH SHAMIR SECRET SHARING

- Suppose X is shared as P(x) and Y is shared as Q(x)
- Adding is trivial: Z = X+Y can be shared as R(x) = P(x)+Q(x)
  - And shares can be computed locally: R(i) = P(i)+Q(i)
- Multiplication looks trivial: Z = X·Y can be shared as R(x) = P(x)·Q(x), but:
  - P(x)·Q(x) is not a (K-1)-degree polynomial
  - Coefficients of P(x)·Q(x) not uniformly distributed
  - Need to execute a “degree reduction” protocol
- Virtual Data Lake, product of Roseman Labs: https://rosemanlabs.com/

PRIVATE SET INTERSECTION

- Each party has a list of values
  - Need to compute the union or intersection of those lists
- Outcome-based pricing
  - List of patients treated with drug X
  - List of patients seen in ER with diagnosis Y
  - Reimbursement for those patients who ended up in ER

REAL-WORLD DATA ENGINE

- Product of Guardtime: https://guardtime.com/health
- Based on commutative encryption: $Enc(k_2, Enc(k_1, m)) = Enc(k_1, Enc(k_2, m))$
  - Each party encrypts the patient IDs in its list with its key
  - Sends the encrypted list to the other party
  - The other party re-encrypts the encrypted IDs with its own key
  - The double-encrypted IDs are comparable due to commutativity
- Distributed auditing capability
  - Each party posts an audit trail to a shared message board
  - Each party separately auditable relative to the board
  - Audits of all parties imply correctness of the complete process

VERIFIABLE MULTI-PARTY COMPUTATION

- Verifiable computation
  - Tools to check that an untrusted party executed a computation correctly
  - See my ACCU 2022 talk for more background
- In the context of multi-party computation
  - Each party can verify that others did their parts correctly
  - An outside recipient can verify the whole computation
- MPyC with passive security: https://github.com/lschoe/mpyc
- MPyCsnark: private research prototype, ask for intro to author

guardtime

THANK YOU

QUESTIONS?

AHTO.TRUU@GUARDTIME.COM
@AHTOTRUU