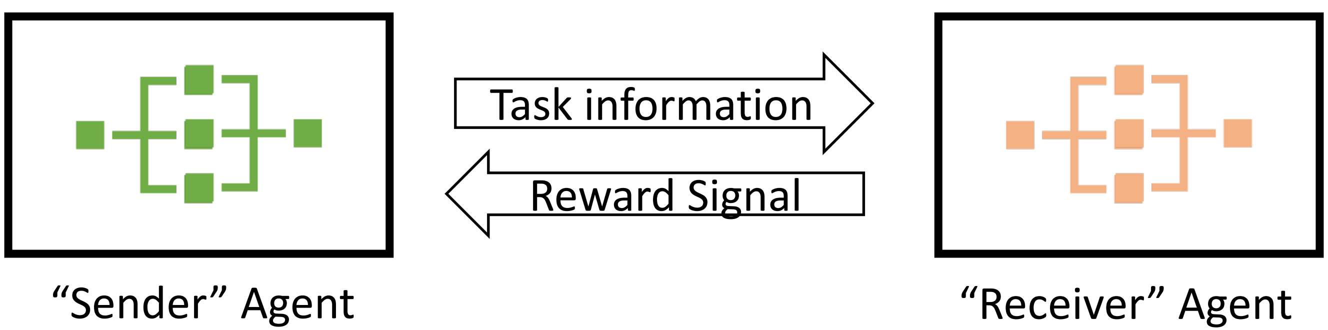
Characterizing and Protecting Multi-Agent Computation

Protecting Information

- How to ensure that data communicated by autonomous agents stays secure and private?
- Communication privacy:
  - TLS – the default for secure communication
  - Differential privacy, PIR, ORAM
- Computation privacy:
  - Partial and fully homomorphic encryption (PHE/FHE)
  - Secure multiparty computation (SMC) for 2 or more parties
  - Hardware assistance through trusted execution environments (TEEs)
- What is efficient and practical for real-world agents given the constraints of space environments?

Multi-Agent Communication

- How do agents characterize their surroundings?
- Can adversaries understand the agent’s conceptual model without access to their reasoning?
- How can agents learn and resolve concepts without human-in-the-loop feedback?



UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

Securing Satellite Collision Avoidance using Secure Multiparty Computation

Caroline Fedele

PhD student, University of Florida

Space Vehicles Directorate

12 October 2022

Project Overview

General goal: ensure data on autonomous agents remains secure and private during computation

Demonstrate feasibility **secure multiparty computation (SMC)**, a method of operating on encrypted data, allowing **collision avoidance (CA)** to be conducted between mutually-distrustful agents without revealing location or trajectory data

- Determined software toolkit/library to implement SMC into a standard CA algorithm
- Constructed hardware setup with embedded processors that survive a space environment, tested simple algorithms and networking setup
- Developed version of CA algorithm to test, determining which parameters need security
- Testing CA program on boards both with and without SMC and benchmark results

Motivation: Security on Satellites

Problem: How can we prevent collisions without revealing exact locations of strategic satellites?

Difficulties:

- need for satellites to share location/trajectory data to prevent collision, poses security risk for satellite owner
- cybersecurity measures to protect data are often computationally expensive and slow
- space, contested and harsh environment, limiting electronics used on satellites

A solution:

→ Privacy-preserving computation (PPC)

- allows for data to remain encrypted during computation
- protects both physical integrity of satellite (allowing collision analysis) and data privacy (preventing unencrypted data from being shared)
- secure multiparty computation (SMC) – promising, most-developed method of PPC

Background: What is SMC?

Secure Multiparty Computation (SMC):

- cryptographic protocol that allows set of mutually-distrusting parties to jointly compute a function on their inputs, without revealing any information about the inputs; enables privacy-preserving computation.
- uses a) *garbled circuits* (2 parties) or b) *linear secret sharing* (>2 parties)
- advance trustworthy machine learning and data mining, helping data privacy in medicine, finance, etc.

Linear Secret Sharing (LSS) scheme:

- keyless distributed encryption process.
- divides the "secret" (inputs) into randomly-generated shares and distributes to independent computing parties.

Background: What is SMC?

Secret Sharing

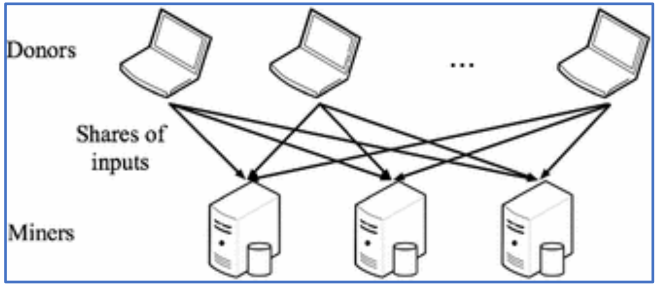
Distribute secret (input) among *n* parties, i.e. location data of 2 satellites. Predefined qualified subsets of *n* can reconstruct secret and return to user

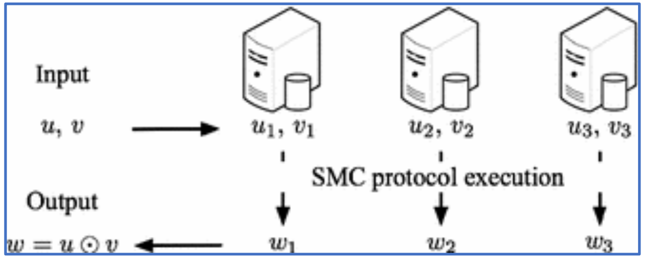
Threshold Secret Sharing

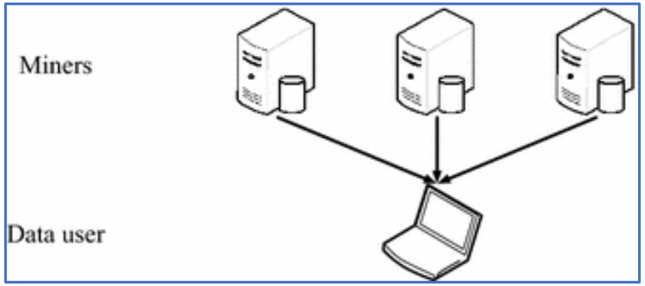
- k-out-of-n scheme
- secret *S* divided into *n* shares: $S = (s_1,\ldots,s_n)$
  - *S* = element of finite field
  - shares = mapping to *S* + several random elements
- compromise of k-1 shares gives no info about S

Secret Sharing

- Donors/data users = satellites participating in collision avoidance
- Miners = 3 computation servers





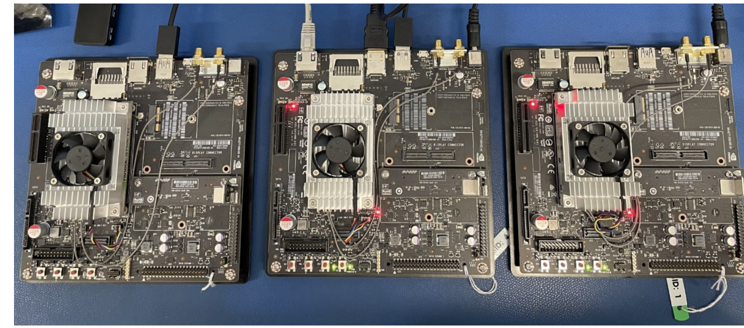


DOI 10.1007/s10207-014-0271-8

Methodology: how it works

Hardware: emulate satellite cluster

- 3 NVIDIA TX2 boards with ARM processors
- ethernet connections, ssh to each other to share data



Hardware setup

Software: integrate SMC into collision avoidance

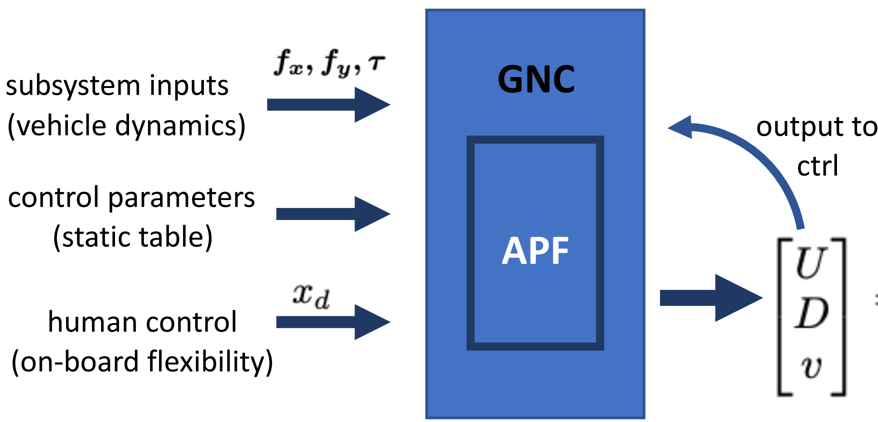
- Sharemind MPC platform
  - 3-party linear secret sharing
  - provides hosts for SMC operations
  - System of libraries compatible with C/C++
- Testing algorithms
  - Simple vector multiplication
  - Collision avoidance

Hawkeye 360 satellite cluster

Press Release, 2020. https://www.he360.com/hawkeye-360-completes-milestone-in-preparation-to-launch-second-cluster/

Collision Avoidance algorithm

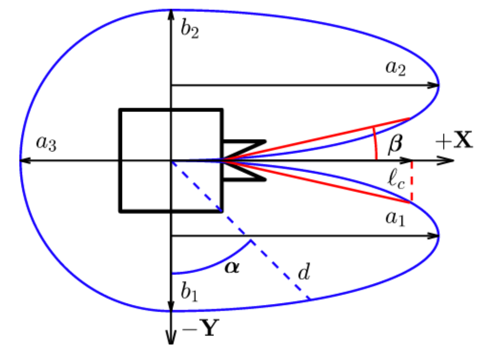
- Artificial Potential Function (APF)
  - On-board trajectory operation and replanning
- Rendezvous and proximity operations (RPO) in guidance navigation and control (GNC) unit of satellite
- Linear (relative) equations of motion



$$\begin{bmatrix} U \\ D \\ v \end{bmatrix} = apf(f_x, f_y, \tau, x_d, \ldots, \text{static params})$$

encrypted values

Keep-out zone potential



$$\phi_r = \sum_{i=1}^{N} \psi_i \exp\left[-\frac{\mathbf{r}_{co_i}^T \mathbf{N}_i \mathbf{r}_{co_i}}{\sigma_i}\right]$$

10.1109/TCST.2018.2866963

Methodology: how it works

Sharemind test program: vector multiplication

- C++ and SecreC working together

```
import stdlib;
import shared3p;

domain pd_shared3p shared3p;


void main() {
        pd_shared3p uint64 ai = argument("ai");
        pd_shared3p uint64 bi = argument("bi");
        pd_shared3p uint64 ci = argument("ci");

    pd_shared3p uint64 product = ai * bi * ci;

    publish("product", product);
}
```

SecreC program

```
namespace sm = sharemind;

inline std::shared_ptr<void> newGlobalBuffer(std::size_t const size) {
    auto * const b = size ? ::operator new(size) : nullptr;
    try {
        return std::shared_ptr<void>(b, sm::GlobalDeleter());
    } catch (...) {
        ::operator delete(b);
        throw;
    }
}

inline std::shared_ptr<void> newGlobalBuffer(void const * const data,
                                             std::size_t const size)
{
    auto r(newGlobalBuffer(size));
    std::memcpy(r.get(), data, size);
    return r;
}

struct ExtraIndentExceptionFormatter {

    template <typename OutStream>
    void operator()(std::size_t const exceptionNumber,
                    std::size_t const totalExceptions,
                    std::exception_ptr e,
                    OutStream out) noexcept
    {
```

C++ program

Conclusions and Future Work

Preliminary results:

- Rewrote CA algorithm in C++ for Sharemind implementation
  - Waiting on data set from AFRL (~2 weeks)
- Simple programs in C++ and SecreC working in sharemind virtual server environment
- Setting up hardware to test on
  - Embedded autonomous development boards
  - UAVs via Docker containers

Future work:

- Test different algorithms
- Look into efficiency improvements
  - parallelization to increase efficiency
  - SIMD vectorization to improve scalability
- Explore other privacy-preserving methods, e.g. partial/fully homomorphic encryption (PHE/FHE)
- Testing on autonomous aerial vehicles
- Testing on other mobile systems

Acknowledgements

Kevin Butler

Tyler Lovelly

Chris Petersen

FICS lab (UF)

SPACER lab (AFRL)