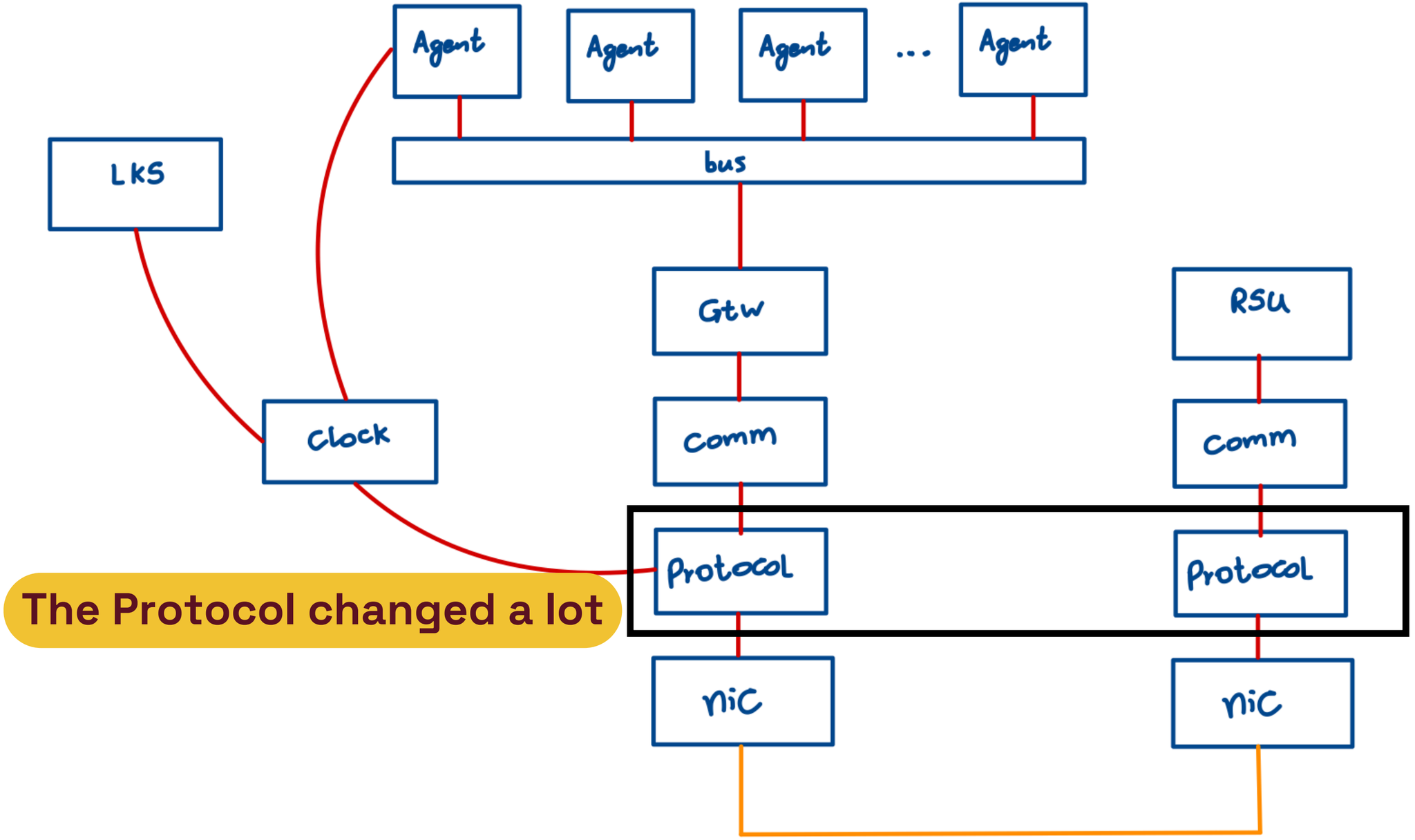
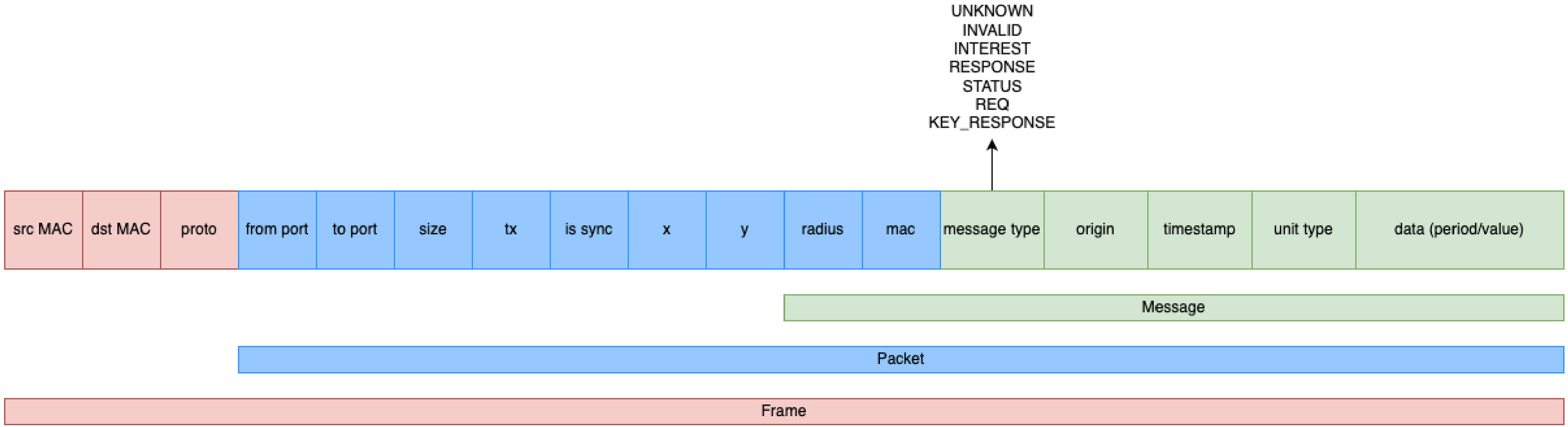
Reliable and secure communication library for critical autonomous systems

CÉSAR AUGUSTO PEREIRA DE SOUZA
ENZO NICOLÁS SPOTORNO BIEGER
JOÃO PEDRO PEREZ RESMER
JOÃO PEDRO SCHMIDT CORDEIRO

P5 09/06/2025

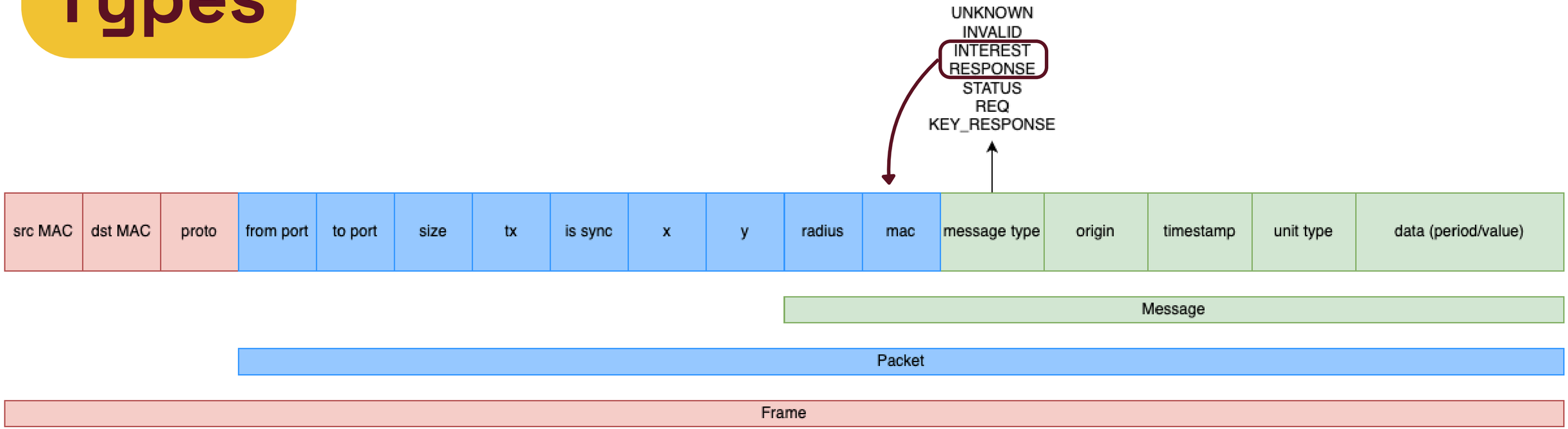

Agent
Agent
Agent
...
Agent
bus
LKS
Gtw
RSU
Clock
comm
comm
Protocol
Protocol
NIC
NIC
The Protocol changed a lot

Messsage


UNKNOWN
INVALID
INTEREST
RESPONSE
STATUS
REQ
KEY_RESPONSE
src MAC
dst MAC
proto
from port
to port
size
tx
is sync
x
y
radius
mac
message type
origin
timestamp
unit type
data (period/value)
Message
Packet
Frame

Messsage Types

Authenticable



INTEREST and RESPONSE messages will have its mac field filled

Calculating MAC

protocol.h

include > api > network > protocol.h > Protocol<NIC>::calculate_mac

```
729  template <typename NIC>
730  MacKeyType Protocol<NIC>::calculate_mac(const void* message_data, unsigned int message_size,
731                                          const Header* header, const TimestampFields* timestamps,
732                                          const Coordinates* coordinates, const MacKeyType& key) {
733      MacKeyType result;
734      result.fill(u: 0);
735
736      // Create a buffer containing all authenticated fields in deterministic order
737      std::vector<uint8_t> auth_data;
738      unsigned int total_size = 0;
739
740      // 1. Add Header fields (from_port, to_port - excluding size to avoid circular dependency)
741      unsigned int header_auth_size = sizeof(Port) * 2; // from_port + to_port
742      total_size += header_auth_size;
743
744      // 2. Add TimestampFields (only sync status, exclude tx_timestamp for architectural cleanliness)
745      unsigned int timestamp_auth_size = sizeof(bool); // Only is_clock_synchronized
746      total_size += timestamp_auth_size;
747
748      // 3. Add Coordinates (full structure for location integrity)
749      unsigned int coords_auth_size = sizeof(Coordinates);
750      total_size += coords_auth_size;
751
752      // 4. Add message payload
753      total_size += message_size;
```

Against tampering and replay attacks

Calculating MAC

protocol.h

include > api > network > protocol.h > Protocol<NIC>::calculate_mac

```
732                                     const Coordinates* coordinates, const MacKeyType& key) {
751
752     // 4. Add message payload
753     total_size += message_size;
754
755     // Allocate buffer for all authenticated data
756     auth_data.resize(sz: total_size);
757     unsigned int offset = 0;
758
759     // Serialize Header fields
760     Port from_port = header->from_port();
761     Port to_port = header->to_port();
762     std::memcpy(dst: auth_data.data() + offset, src: &from_port, n: sizeof(Port));
763     offset += sizeof(Port);
764     std::memcpy(dst: auth_data.data() + offset, src: &to_port, n: sizeof(Port));
765     offset += sizeof(Port);
766
767     // Serialize TimestampFields (only sync status, exclude tx_timestamp)
768     bool sync_status = timestamps->is_clock_synchronized;
769     std::memcpy(dst: auth_data.data() + offset, src: &sync_status, n: sizeof(bool));
770     offset += sizeof(bool);
771     // Note: tx_timestamp is intentionally excluded from MAC calculation
772
773     // Serialize Coordinates
774     std::memcpy(dst: auth_data.data() + offset, src: coordinates, n: sizeof(Coordinates));
775     offset += sizeof(Coordinates);
```

Against tampering and replay attacks

Calculating MAC

protocol.h

include > api > network > protocol.h > Protocol<NIC>::calculate_mac

```
732                                     const Coordinates* coordinates, const MacKeyType& key) {
772
773     // Serialize Coordinates
774     std::memcpy(dst: auth_data.data() + offset, src: coordinates, n: sizeof(Coordinates));
775     offset += sizeof(Coordinates);
776
777     // Serialize message payload
778     std::memcpy(dst: auth_data.data() + offset, src: message_data, n: message_size);
779
780     // Debug logging: Show what's being authenticated
781     db<Protocol>(INF) << "[Protocol] Hybrid MAC - Authenticating " << total_size << " bytes total:\n";
782     db<Protocol>(INF) << "[Protocol] Hybrid MAC - Header: from_port=" << from_port
783                       << ", to_port=" << to_port << " (" << header_auth_size << " bytes)\n";
784     db<Protocol>(INF) << "[Protocol] Hybrid MAC - Timestamps: sync=" << sync_status
785                       << " (" << timestamp_auth_size << " bytes, tx_timestamp excluded)\n";
786     db<Protocol>(INF) << "[Protocol] Hybrid MAC - Coordinates: x=" << coordinates->x      enzoniko, 2 days ago • ch
787                       << ", y=" << coordinates->y << ", radius=" << coordinates->radius
788                       << " (" << coords_auth_size << " bytes)\n";
789     db<Protocol>(INF) << "[Protocol] Hybrid MAC - Message payload: " << message_size << " bytes\n";
790
791     // XOR-based MAC calculation on combined authenticated data
792     const uint8_t* combined_data = auth_data.data();
793     for (unsigned int i = 0; i < total_size; ++i) {
794         result[i % 16] ^= combined_data[i];
795     }
796
```

Against tampering and replay attacks

Calculating MAC

protocol.h

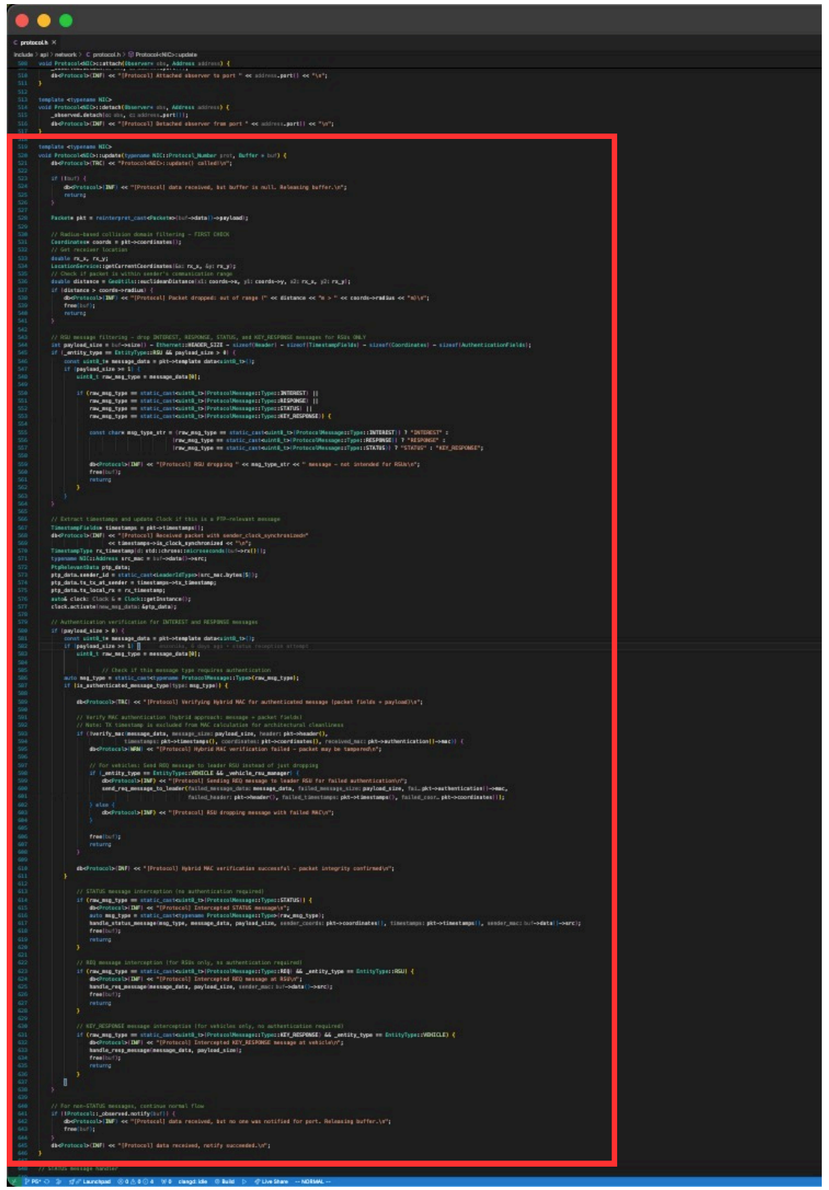
include > api > network > protocol.h > Protocol<NIC>::calculate_mac

```
732                                     const Coordinates* coordinates, const MacKeyType& key) {
791     // XOR-based MAC calculation on combined authenticated data
792     const uint8_t* combined_data = auth_data.data();
793     for (unsigned int i = 0; i < total_size; ++i) {
794         result[i % 16] ^= combined_data[i];
795     }
796
797     // XOR with the key
798     for (size_t i = 0; i < 16; ++i) {
799         result[i] ^= key[i];
800     }
801
802     // Debug logging: Show computed MAC
803     std::string computed_mac_hex = "";
804     for (size_t i = 0; i < 16; ++i) {
805         char hex_byte[4];
806         snprintf(str: hex_byte, size: sizeof(hex_byte), format: "%02X ", result[i]);
807         computed_mac_hex += hex_byte;
808     }
809     db<Protocol>(INF) << "[Protocol] Hybrid MAC - Computed MAC: " << computed_mac_hex << "\n";
810
811     return result;
812 }
813
814 template <typename NIC>
```

Against tampering and replay attacks

MAC

The conditionals are inside the update in Protocol



Leader Selection

- RSUs are the **leaders**
- RSUs generate **STATUS** messages and send them **broadcast**
- Vehícles have **two** lists of **known RSUs**
- The **nearest** RSU is the leader (leader key used for tx)

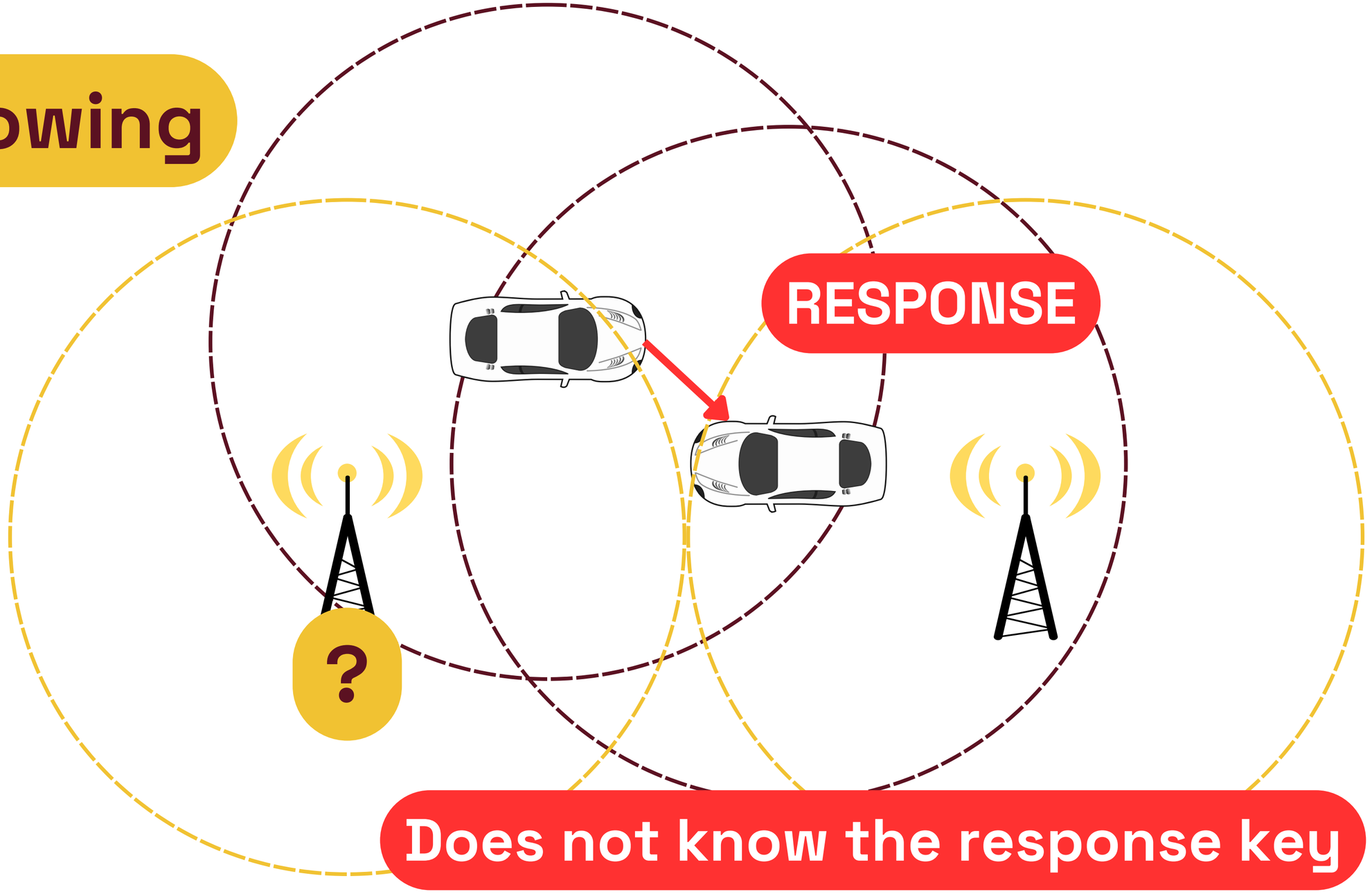
```
53  private:
54      std::vector<RSUInfo> _known_rsus;           // List of known RSUs
55      mutable std::mutex _rsu_list_mutex;         // Thread safety
56      RSUInfo* _current_leader;                   // Current closest RSU
57      std::chrono::seconds _rsu_timeout;          // RSU timeout period
58      unsigned int _vehicle_id;                   // For logging
59
60      // Neighbor RSU keys (just keys, not full info)
61      std::vector<MacKeyType> _neighbor_rsu_keys;
62      mutable std::mutex _neighbor_keys_mutex;    // Thread safety for neighbor keys
```

Leader Selection

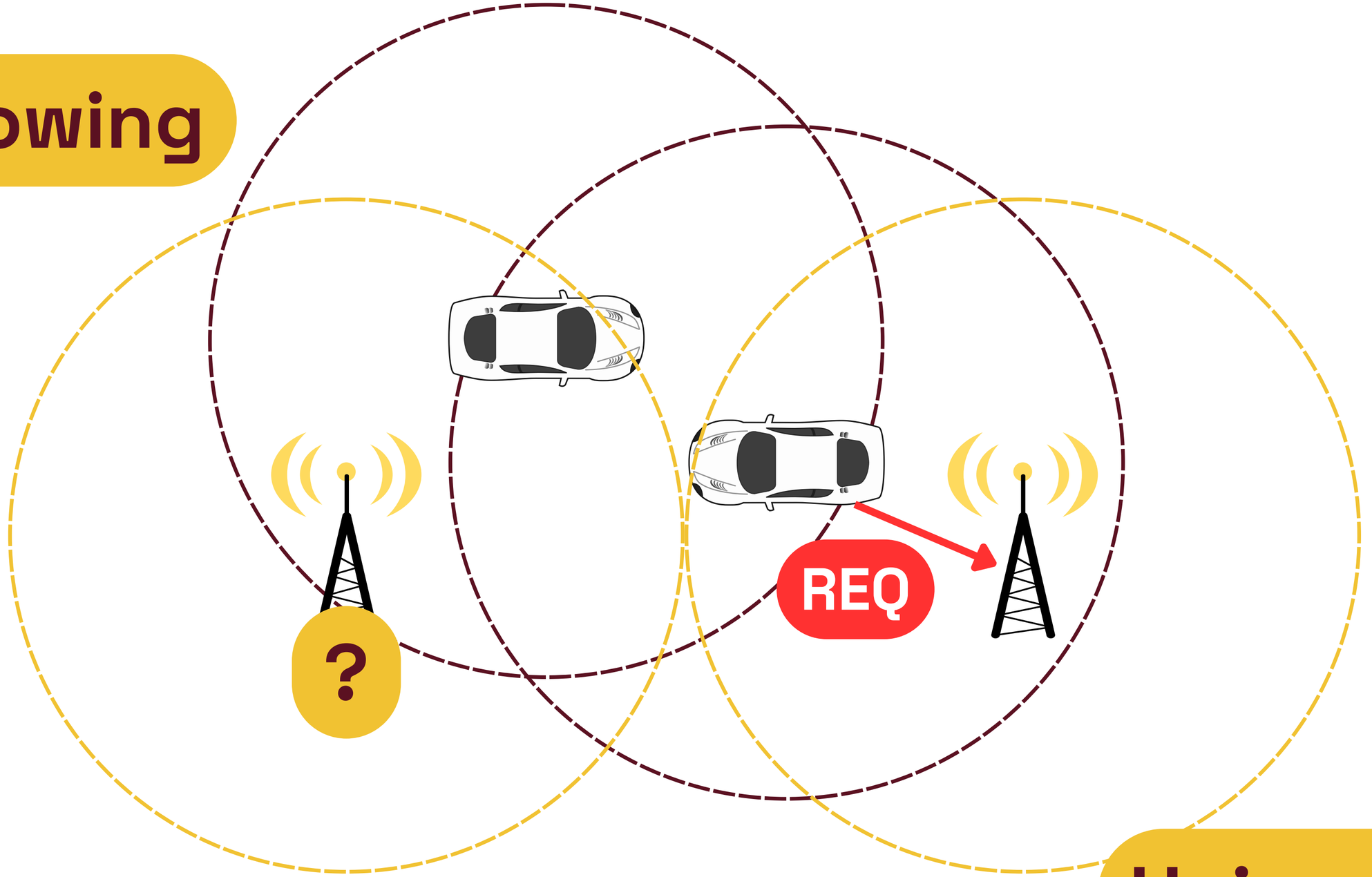
When it receives a message that should be authenticated, it verifies in every RSU that he knows, checking if the message was correctly authenticated.

```
53    private:
54        std::vector<RSUInfo> _known_rsus;           // List of known RSUs
55        mutable std::mutex _rsu_list_mutex;         // Thread safety
56        RSUInfo* _current_leader;                   // Current closest RSU
57        std::chrono::seconds _rsu_timeout;          // RSU timeout period
58        unsigned int _vehicle_id;                   // For logging
59
60        // Neighbor RSU keys (just keys, not full info)
61        std::vector<MacKeyType> _neighbor_rsu_keys;
62        mutable std::mutex _neighbor_keys_mutex;    // Thread safety for neighbor keys
```

Shadowing

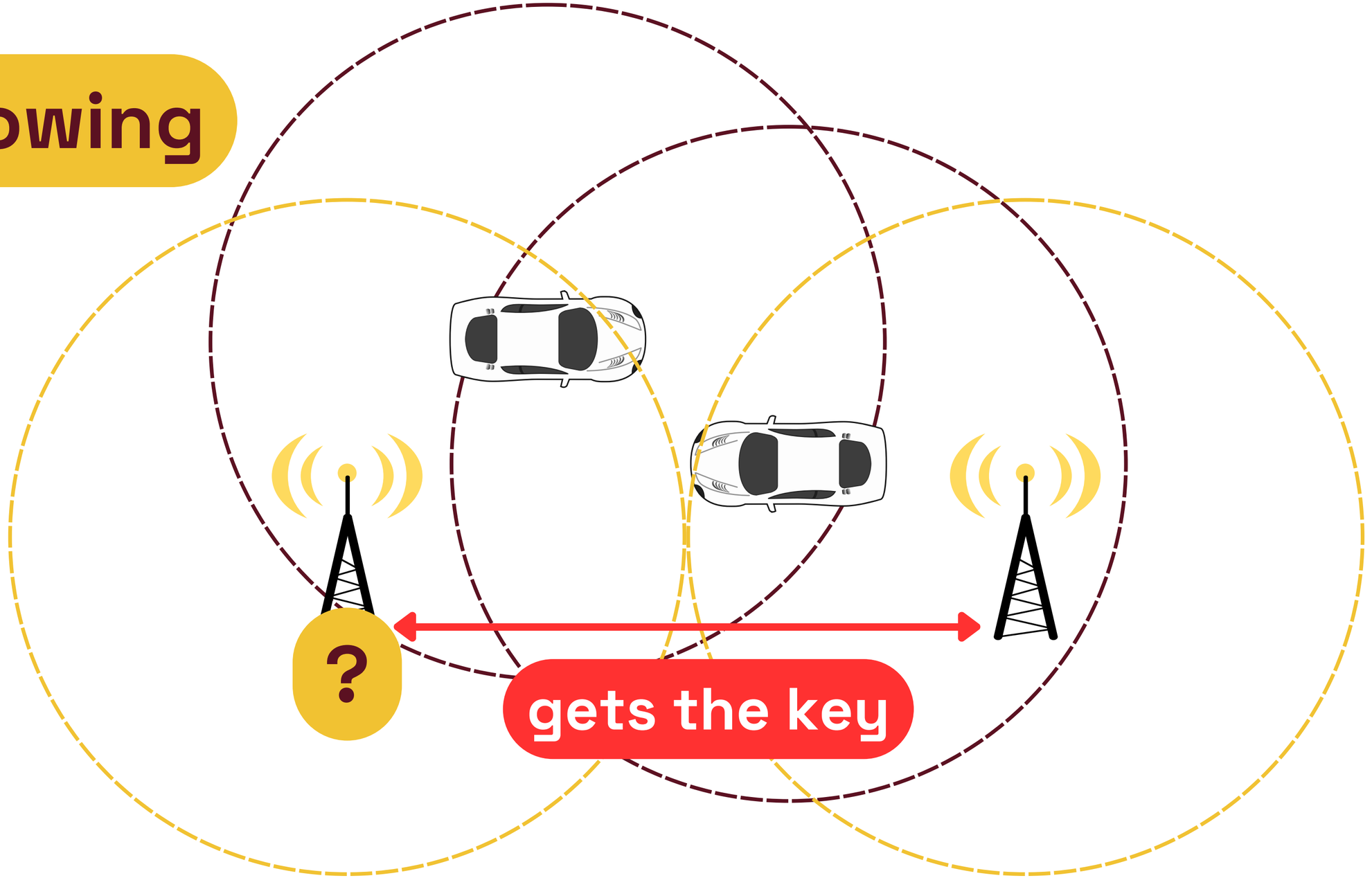


Shadowing

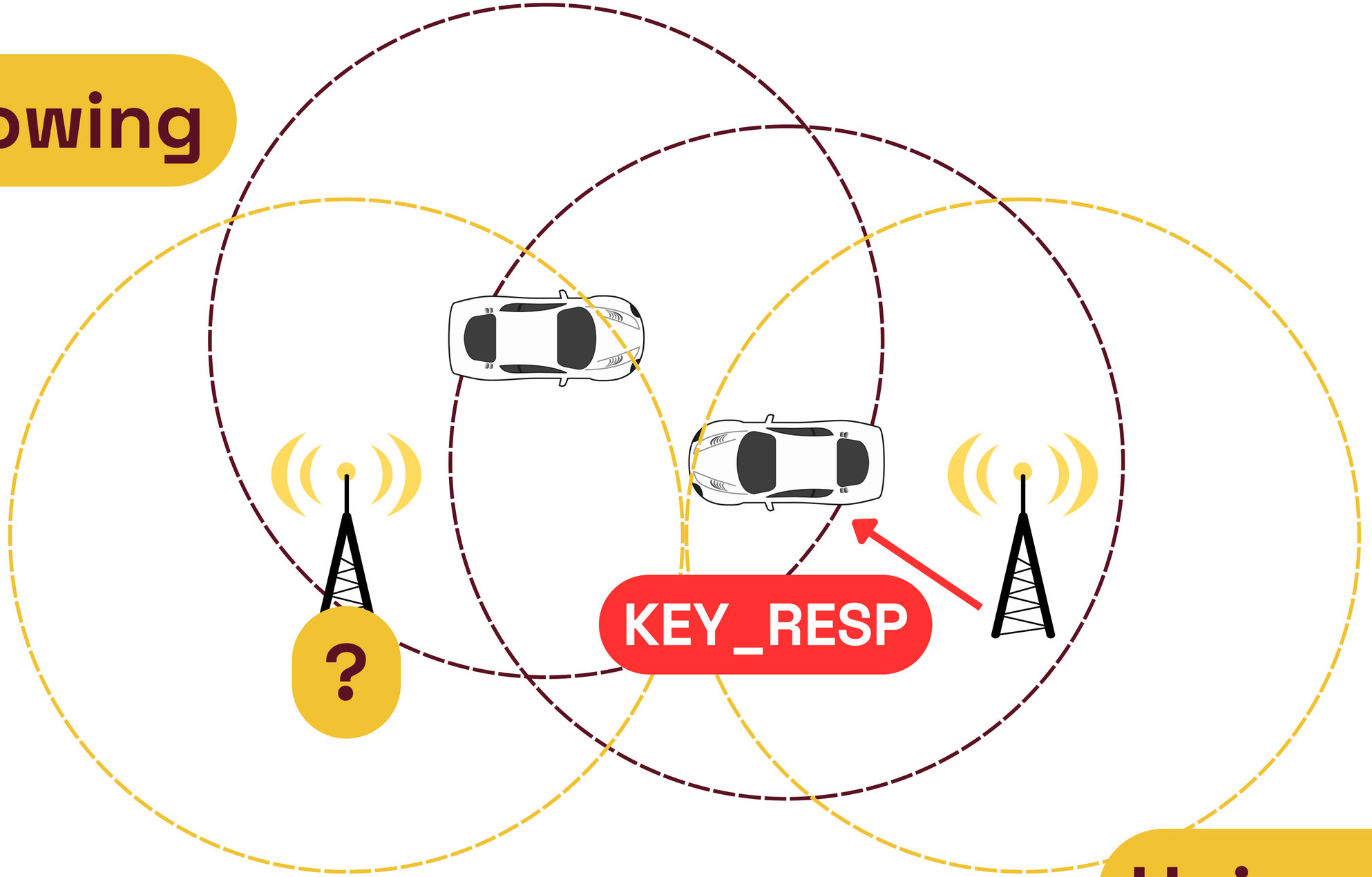


Unicast

Shadowing



Shadowing



Unicast

Other key points:

- **Shadowing:** Verify in the Primary and Secondary RSUs list
- **Implicit ACK:** You have to ask again the key to authenticate
- Test with maps **enabled**

Map example

{} map_1_config.json

config > {} map_1_config.json > ...

enzoniko, 8 hours ago | 1 author (enzoniko)

enzoniko, 7 days ago • simplification of map generation and map 1 but …

```json
{
  "map_info": {
    "name": "Map 1 - Simple Cross",
    "description": "Simple map with one RSU and vehicles crossing its domain"
  },
  "simulation": {
    "duration_s": 10,
    "update_interval_ms": 100,
    "default_transmission_radius_m": 500,
    "trajectory_generator_script": "scripts/trajectory_generator_map_1.py"
  },
  "rsu": {
    "id": 1000,
    "position": {"x": 0, "y": 0},
    "unit": 999,
    "broadcast_period_ms": 250
  },
  "vehicles": {
    "default_count": 10,
    "speed_kmh": 50
  },
  "waypoints": [
    {"name": "west_entry", "x": -400, "y": 0},
    {"name": "east_exit", "x": 400, "y": 0},
    {"name": "north_entry", "x": 0, "y": 400},
    {"name": "south_exit", "x": 0, "y": -400}
  ],
  "routes": [
    {"name": "west_to_east", "waypoints": ["west_entry", "east_exit"]},
    {"name": "east_to_west", "waypoints": ["east_exit", "west_entry"]},
    {"name": "north_to_south", "waypoints": ["north_entry", "south_exit"]},
    {"name": "south_to_north", "waypoints": ["south_exit", "north_entry"]}
  ],
  "logging": {
    "trajectory_dir": "tests/logs/trajectories"
  }
}
```

Reliable and secure communication library for critical autonomous systems

CÉSAR AUGUSTO PEREIRA DE SOUZA
ENZO NICOLÁS SPOTORNO BIEGER
JOÃO PEDRO PEREZ RESMER
JOÃO PEDRO SCHMIDT CORDEIRO

P5 09/06/2025