Dynamic Address Validation Array (DAVA): A Moving Target Defense Protocol for CAN bus

Richard Brown, Alex Marti, Chris Jenkins, Advisor: Dr. Susmit Shannigrahi
Department of Computer Science, Tennessee Tech University

ABSTRACT

Dynamic Address Validation Array (DAVA) is a novel moving target defense protocol for the Controller Area Network Bus (CAN bus). DAVA's primary goal is to mitigate the common CAN bus vulnerability of an unauthorized entity misappropriating components in the vehicle through sniffing and reusing ECU IDs for replaying messages. Using a dynamically allocated array stored in the ECU that is updated and validated frequently, DAVA limits an attacker's ability to reuse ECU IDs for replay attacks. The protocol strives to be minimally invasive and lightweight for application in CAN bus while still being secure. The following discusses the DAVA protocol, a proof of concept implementation, and initial performance measurements. The implementation will explain how DAVA is able to provide a robust security framework for CAN bus without the need for a large amount of storage or CAN bus standard modification.

INTRODUCTION

- The Controller Area Network bus (CAN bus) [1] is a non-IP based network that allows many devices and microcontrollers to communicate using the CAN standard[1] and is widely used in numerous every-day products.
- CAN has several features that make it ideal for these use cases including robustness, priority of messages, and the ability to support real-time communication with deadlines.

PROBLEM DEFINITION

- The CAN standard lacks encryption or authentication methods [2] due to the limited processing capacity of the microcontrollers on the bus and the fact that CAN uses eight-byte packets that are too small for most standard security mechanisms [2].
- The CAN packets arrive and are decoded within a deadline, making the use of multiple packets for carrying an encrypted payload difficult [3].
- Due to this lack of security, an attacker can sniff the bus for the device IDs of the devices and use them to send malicious messages to seize control of various components of the bus.
- A mitigation of the CAN standard's vulnerabilities must not be allowed to compromise the speed of the CAN protocol.

OBJECTIVES

- Design a lightweight CAN security protocol that mitigates sniffing the bus and replay attacks based off Moving Target Defense ideas
- Create a simulator that demonstrates the basic functions of the protocol.
- Evaluate the protocol by identifying its time complexity, space complexity, and security effectiveness provided.

METHODOLOGY

- DAVA executes in three phases:
  - Initialization Phase
    - All nodes on the CAN bus create an MTD array that stores information of every other node on the bus.
    - With this information, every node knows who they are and who the other nodes on the bus are.
  - Operational Phase
    - Normal CAN bus activity where nodes use the device IDs in their MTD array to identify themselves and others when addressing and sending messages.
  - Update Phase
    - When any node receives a message, all nodes on the bus will change the device ID of the recipient to a random number based off a shared seed to ensure the ID is the same across the bus.

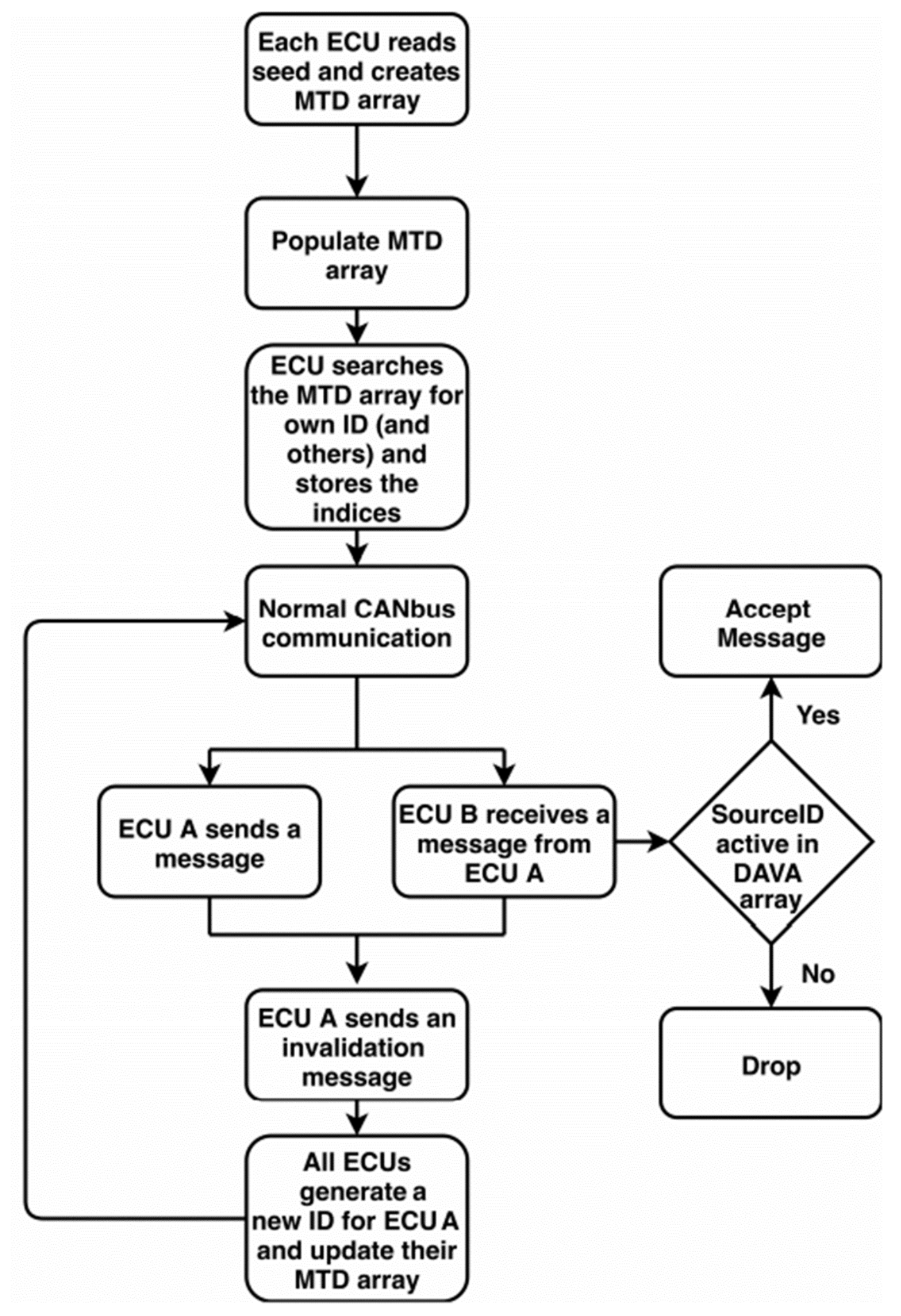
Each ECU reads seed and creates MTD array
Populate MTD array
ECU searches the MTD array for own ID (and others) and stores the indices
Normal CANbus communication
ECU A sends a message
ECU B receives a message from ECU A
SourceID active in DAVA array
Yes
Accept Message
No
Drop
ECU A sends an invalidation message
All ECUs generate a new ID for ECU A and update their MTD array

Figure 1: DAVA Flowchart

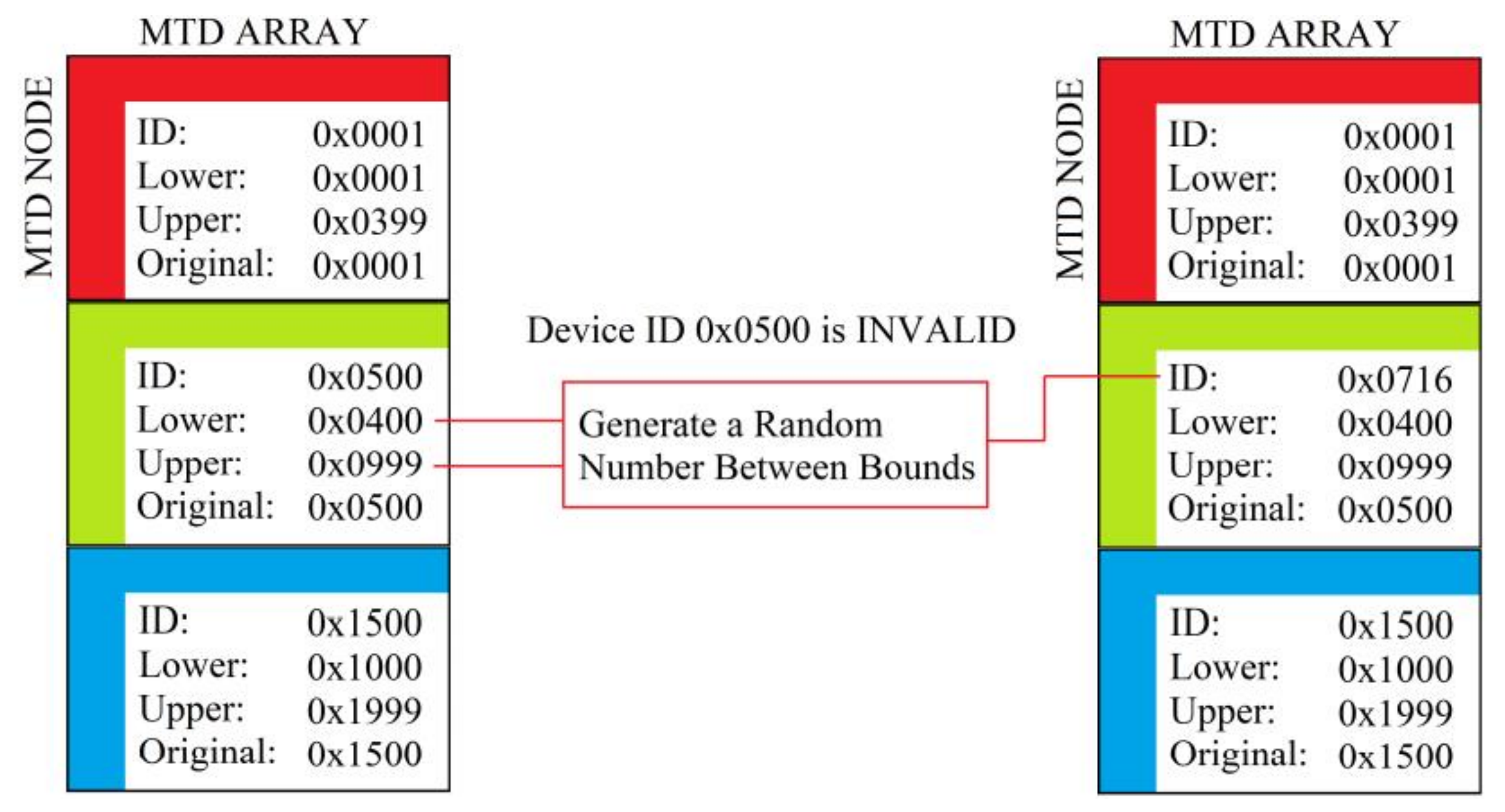

Figure 2: MTD arrays updating a Device ID

- Every device on the bus is given an initialization file that is inputted prior to use that contains the number of devices on the CAN bus, the device IDs of all devices, the lower and upper bounds for all device IDs, and a randomization seed.
- The MTD Array that every node generates at the initialization phase is the key to DAVA's security.
  - The array contains a Device ID, the lower bound, the upper bound, and the original device ID for every node on the bus.

SIMULATOR IMPLEMENTATION

- Designed a basic CAN bus simulator written in C to test DAVA's functionality as a proof of concept.
- Executes on Ubuntu 18.04 virtual machine with a single 2.6 GHz processor.
- Calculates average update time with a bus of two ECUs.
  - Average update time is 0.3 milliseconds.

SECURITY RESULTS

- Sniffing the bus is no longer effective, an attacker can read the messages sent on the bus, but will not know who is the sender and recipient.
- Nearly impossible to execute a malicious command on any CAN bus nodes.
- Replay attacks are no longer possible when the IDs change after every message.
  - All device IDs become one-time-use.
- There are 2^29-1 device IDs that would need to be tried. DAVA imposes 1 in 536,870, 911 chance of performing a successful attack

EVALUATION

- Space Complexity
  - 20 Devices:
    - >= 336 Bytes per device
    - >= 6416 Bytes per bus
  - 100 Devices:
    - >= 1,616 Bytes per device
    - >= 160,016 Bytes per bus
- Time Complexity is linear.

	Update After Every Message	Update After N Messages
CAN Standard Modification	No	No
Time Complexity	$O(n)$	$O(n)$
Per-node Space Complexity	$O(n)$	$O(n)$
Total Space Complexity	$O(n^2)$	$O(n^2)$

Figure 3: DAVA protocol properties

CONCLUSION & FUTURE WORK

- DAVA prevents reconnaissance and replay attacks.
- Reliable and lightweight design, allowing for easy adoption to the CAN standard.
- Future work
  - Investigate better randomization seeds and worst case scenarios
  - Utilize a random number generator that is closer to more random.
  - Implement DAVA on a CAN bus emulator

ACKNOWLEDGEMENTS

- We would like to thank Chris Jenkins from Sandia National Labs for supervising our research and for providing mentorship in formulating Moving Target Defense algorithms.

REFERENCES

[1] 2020. ISO 11898-1:2015. https://www.iso.org/standard/63648.html [Online; accessed 13. Jun. 2020].
[2] Steve Corrigan HPL. 2002. Introduction to the controller area network (CAN). Appl. Rep. SLOA101 (2002), 1–17.
[3] Roderick Currie. 2017. Hacking the can bus: basic manipulation of a modern automobile through can bus reverse engineering. SANS Institute (2017).