# Encrypted messaging on CAN bus

**Dr. Ken Tindell, CTO Canis Labs**
ken@canislabs.com


# Encryption on CAN bus

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

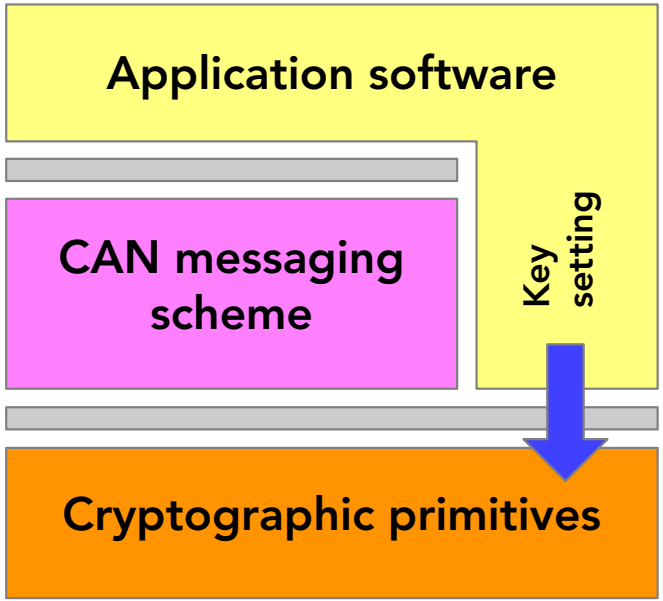
- **CAN was never designed for this**
- **But it can be done – *if you do it right***
  - Must meet very specific requirements for CAN bus messaging
  - Mainstream IT solutions don't fit well
- **Encryption on CAN does help – *for some problems***
  - It can keep messages **secret** (makes reverse engineering really difficult)
  - It can ensure messages are **authentic** (stops spoofing / fuzzing)
- **Encryption on CAN is not a complete CAN security solution**
  - It doesn't prevent denial-of-service attacks, for example

# Requirements

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

- **#1 Must support publish/subscribe model of communication**
  - This is how CAN works: atomic broadcast from a single sender to many receivers
- **#2 Real-time messaging**
  - A distributed real-time control system: messages must have bounded latencies
- **#3 Must fit into CAN sized messages**
  - Small payloads (8 bytes on classic CAN)
- **#4 Must fit into limited-resource hardware**
  - Tiny microcontrollers with limited CPU power, RAM and flash
- **#5 Fast start communications**
  - Receiver must be able to start listening long after sender has started

# The messaging stack




- **Cryptographic primitives**
  - Encryption algorithm (e.g. AES-128)
  - Authentication algorithm (e.g. AES-CMAC)
  - Random numbers (“CSPRNG”)
  - Key distribution and storage
- **CAN messaging scheme**
  - TX: Plaintext frame in, ciphertext out
  - RX: Ciphertext in, plaintext frame out

# Cryptographic primitives

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

- **Standard algorithms (FIPS/NIST)**
  - AES-128: encrypts a 16-byte block with a 128-bit key
  - AES-CMAC: produces a 128-bit **message authentication code** (MAC) from a 'message' (i.e. a chunk of data)
  - Random number generator
- **Different implementations**
  - **Hardware Security Modules** (HSMs) provide algorithms **plus key storage**
  - On-chip **accelerator** for AES-128 with software for CMAC and key storage (in EEPROM/flash)
  - Pure software (AES-128, CMAC, key storage)

# SHE HSMs

- **SHE HSM is common on automotive microcontroller silicon**
  - "**S**ecure **H**ardware **E**xtensions"
  - E.g. CSEc on NXP S32K144
- **Standardized by AUTOSAR**
  - *Specification of Secure Hardware Extensions* (948 R20-11)
- **Key management**
  - 10 user keys with key permissions (encryption or authentication – but not both)
  - AES-128, AES-CMAC, RNG, etc.
  - Secure key loading operations

# Pure software SHE HSM

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

| Cryptographic operation | Execution time (constant) | Without round-key cache |
|---|---|---|
| Random number | 13.654µs | |
| AES-128 encrypt | 13.662µs | 19.214µs |
| AES-CMAC generate | 14.552µs | 20.086µs |
| AES-CMAC verify | 15.470µs | 20.998µs |

| ROM table | Size /bytes |
|---|---|
| Encrypt | 1024 |
| Decrypt | 1280 |
| Other | 72 |

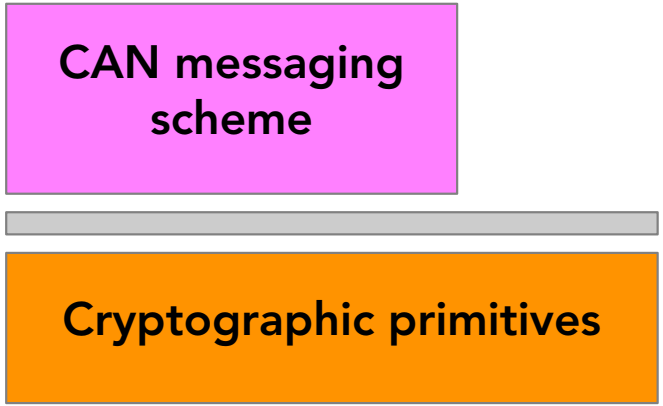
| RAM table | Size /bytes |
|---|---|
| Round-key cache | 3072 |
| PRNG round-key | 176 |
| Other | 16 |

| Cryptographic operation | Code size /bytes |
|---|---|
| CMAC | 680 |
| Encrypt | 860 |
| Decrypt | 1076 |
| Load key | 1048 |
| RNG | 232 |
| Other | 744 |

* Times for a Cortex M0 at 133MHz with 0 wait-state code memory

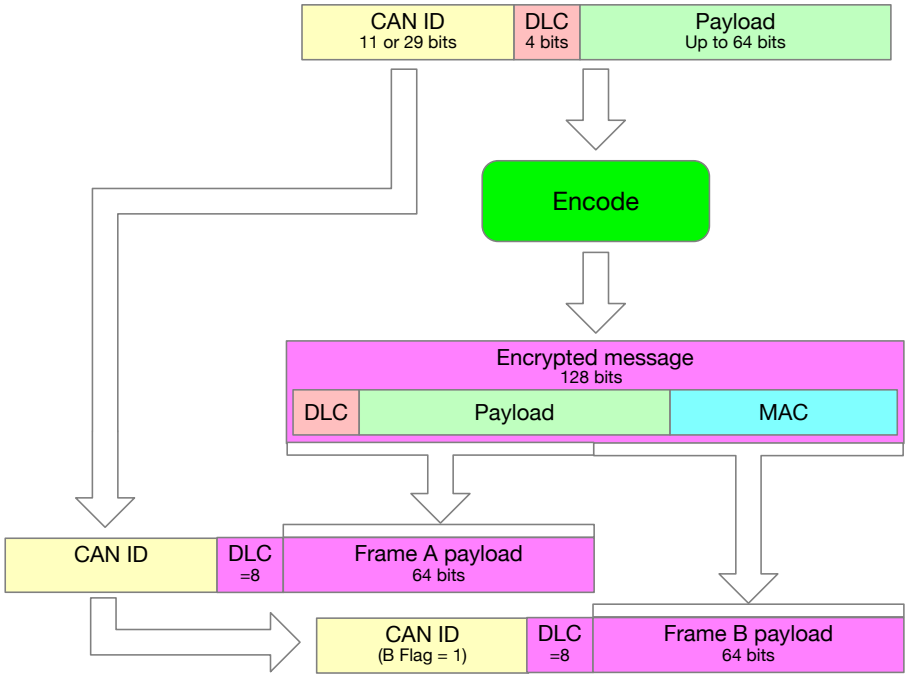
# CAN messaging scheme

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry



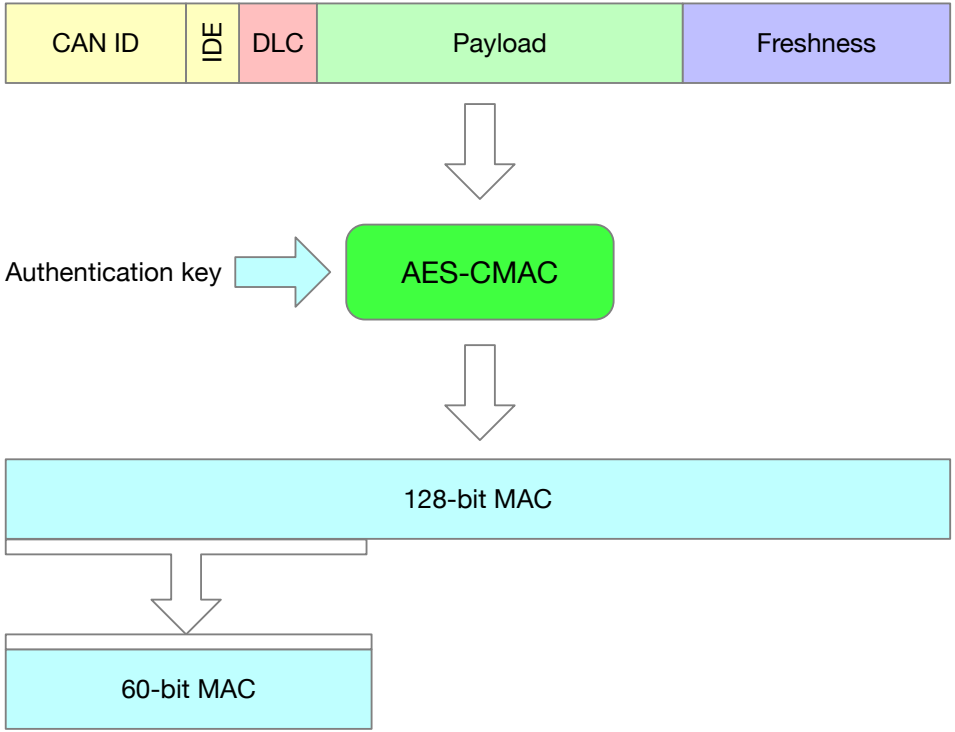

- **Needs some SHE HSM functions**
  - Encrypt (but **not decrypt)**
  - MAC (generate and verify)
  - Random number generator
- **Sender creates a 128-bit message**
  - Plaintext payload (up to 64 bits)
  - Plaintext DLC (4 bits)
  - MAC (60 bits)
- **Generate MAC then encrypt**
  - MtE scheme

# Encrypting CAN messages



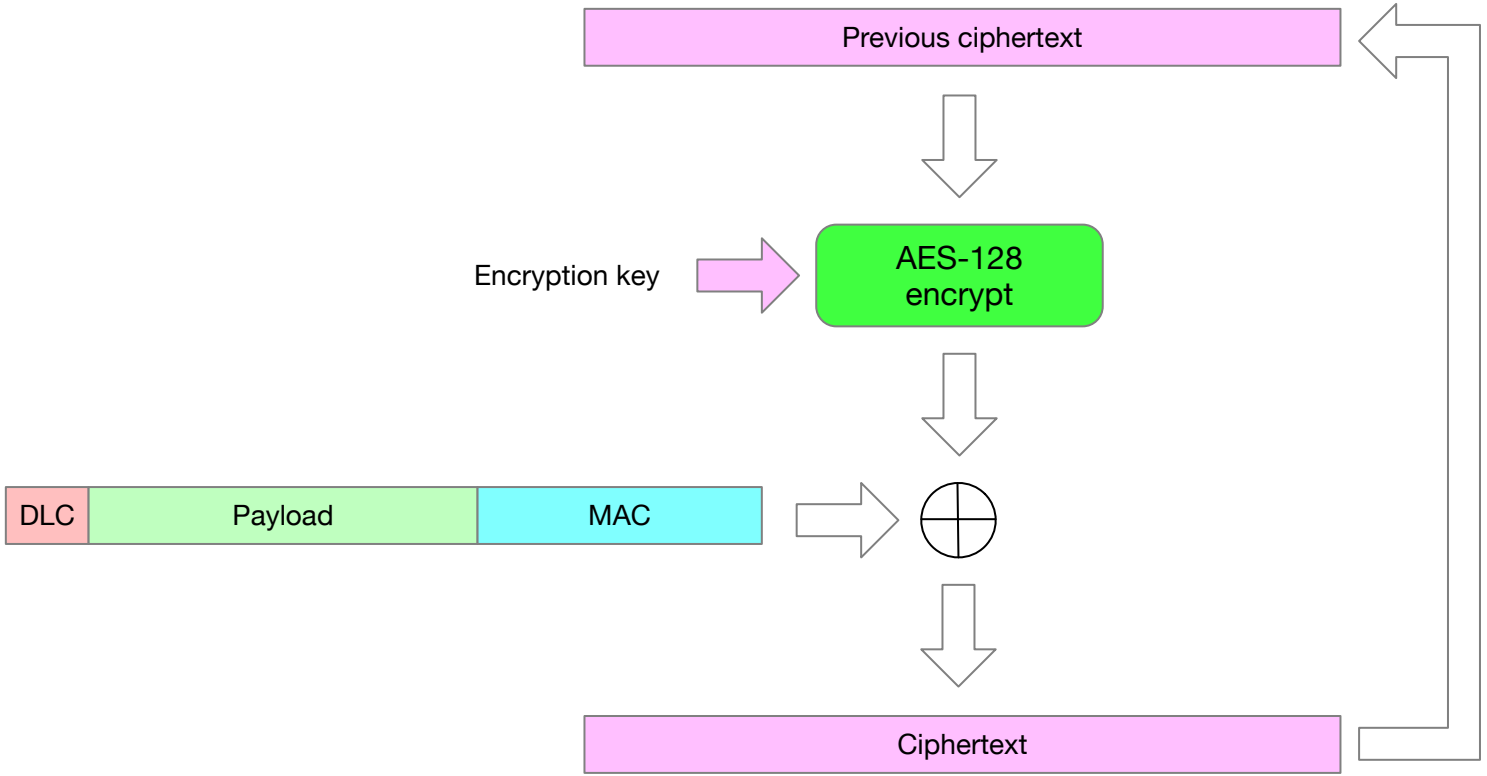

- **Start with a plaintext CAN frame**
  - 11- or 29-bit ID, 0-8 bytes
- **Generate a MAC**
  - 60 bits
- **Encrypt DLC, payload, MAC**
  - One AES block
- **Split into two frames**
  - A and B
  - A is higher priority
- **B Flag in CAN ID**

# Encode step #1: Generate MAC

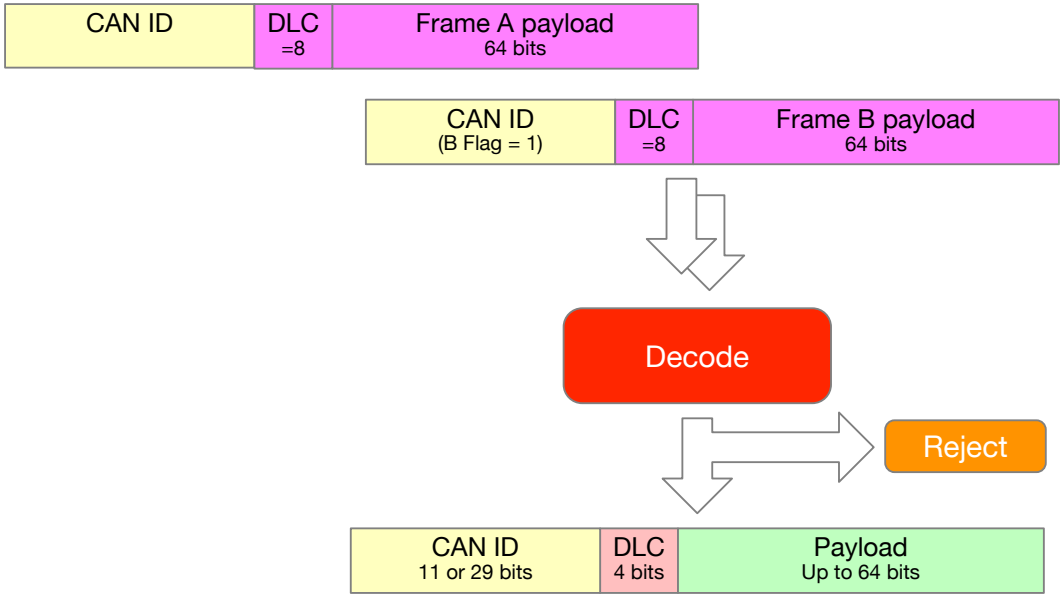



- **Create a 128-bit 'message'**
  - CAN ID, DLC, Payload
  - Plus a **freshness value**
- **Use HSM to generate MAC**
  - 128 bits
- **Truncate the MAC**
  - 60 bits

# Encode step #2: Encrypt

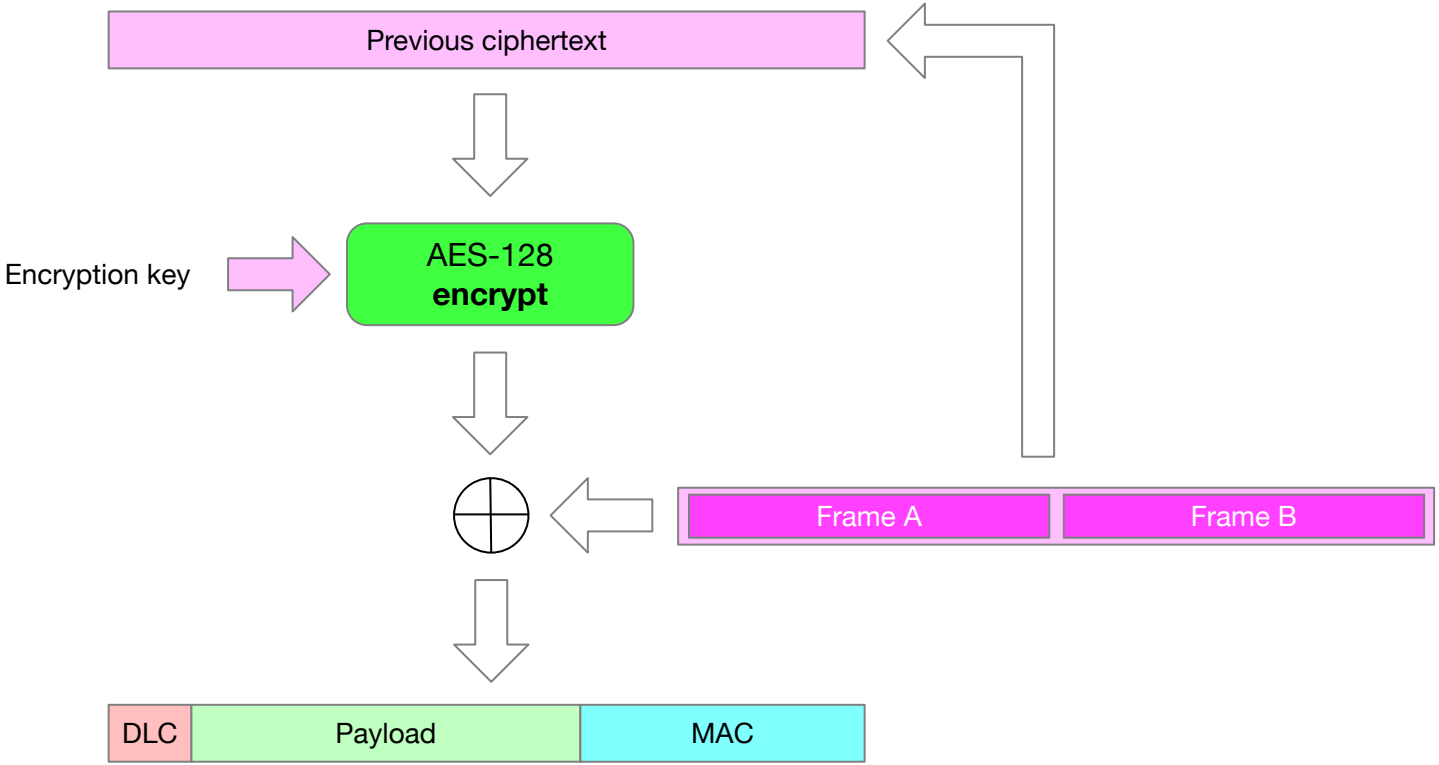


# Decrypting CAN messages

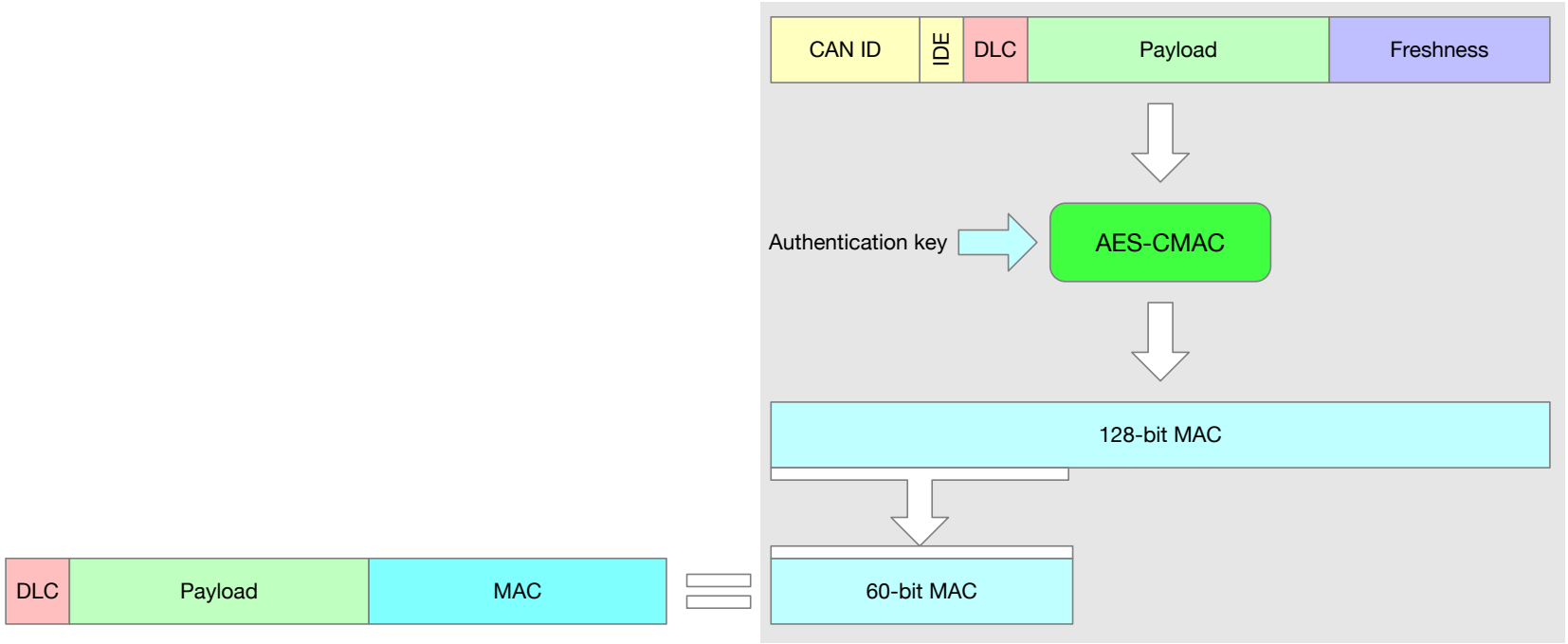



- **Assemble A and B frames**
  - A Frame comes first
- **Decode the block**
  - Reverse of encode operation
- **Verify the MAC**
  - Compare calculated and received MACs
  - Reject if no match
- **Extract the plaintext frame**

# Decode step #1: Decrypt



# Decode step #2: Verify MAC



# SHE HSMs and MACs

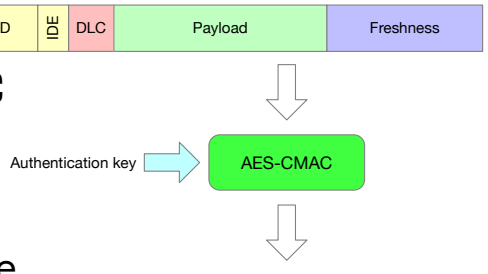
CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

- **Generate and Verify are two separate operations**
  - AES-CMAC is the underlying algorithm
  - Generate takes a ‘message’ and returns a MAC
  - Verify takes a ‘message’ and a MAC and returns pass/fail
- **Key must be given Authentication permission**
  - Can’t use an encryption key
  - CAN encrypted messages use a pair of keys (one for encrypt, one for authenticate)
- **SHE+ adds Verify Only permission**
  - Verify Only = cannot generate MACs, only submit them for verification

# So what about this ‘freshness’?

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

- **Problem with cryptographic messaging is replay attacks**
  - We are protected against forged messages
  - But messages captured by an attacker and retransmitted later **aren’t forgeries**
  - Attacker can know what those messages mean (e.g. “unlock the door”) and just send one when needed
- **Solution is to add a freshness value to the MAC calculation**
  - Freshness is known at sender and at receivers
  - Freshness regularly changes (it could be a global time value)
  - Captured messages with an old freshness value will not verify



# Freshness timing issue

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

- **That "known at sender and at receivers" is a bit of a problem**
  - Freshness at sender when the message is **created** vs freshness at receivers when message is **received**
  - The latency of Frame B may be relatively long compared to the rate at which the freshness value changes
- **Receiver needs to know the freshness value used in the past**
  - Freshness is not transmitted in the message, only inferred
  - But the original freshness value must be relatively recent (because CAN frame latencies are relatively short)
- **Solution is to use some extra bits in a CAN frame..**

# Extra bits in a CAN frame

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

- **Every 8-byte CAN frame has 3 spare bits**
  - Six bits across Frame A and Frame B
- **How? Because DLC values of 8 to 15 all mean "8 bytes"**
  - When bit 3 is a '1' then the bottom 3 bits are "don't care" under the CAN protocol
- **Frame A and B DLCs can carry least-significant 6 bits of freshness value**
  - Receivers can use these to infer the original freshness value
  - MAC verification then is done on inferred freshness value
  - Freshness used as a sequence number to discard verified but stale messages

# CAN messaging performance

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

| Messaging operation | Execution time | Without replay protection |
|---|---|---|
| Create A/B pair | 31.222µs | 30.158µs |
| Receive A | 14.310µs | 14.310µs |
| Receive B | 18.526µs | 18.206µs |

| Function | Code size /bytes |
|---|---|
| Startup | 44 |
| Create A/B pair | 466 |
| Receive frame | 560 |

| RAM structures | Size /bytes |
|---|---|
| Per transmit context | 24 |
| Per receive context | 40 |

* Times include SHE+ emulation time; data from a Cortex M0 at 133MHz

# So how did we do?

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

## Requirements

- **#1 Must support publish/subscribe model of communication**
  - This is how CAN works: atomic broadcast from a single sender to many receivers
- **#2 Real-time messaging**
  - A distributed real-time control system: messages must have bounded latencies
- **#3 Must fit into CAN sized messages**
  - Small payloads (8 bytes on classic CAN)
- **#4 Must fit into limited-resource hardware**
  - Tiny microcontrollers with limited CPU power, RAM and flash
- **#5 Fast start communications**
  - Receiver must be able to start listening long after sender has started

4

Just like CAN

Latency of Frame B = latency

Frame A/B pair

Fast and small on Cortex M0

CFB mode
for random
access

# Remarks

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

- **Needs two CAN frames per plaintext CAN frame**
  - A bandwidth cost but don't need to protect every CAN frame – only sensitive ones
- **The sender could be remote**
  - Ciphertext frames could come from a domain controller via Ethernet
  - Ciphertext frames could come from a tool located externally, even in the cloud
- **Key management is.. key**
  - "*Cryptography is a machine for turning any problem into a key management problem*"
  - Key management infrastructure is vital

# Canis Labs CryptoCAN

CANIS AUTOMOTIVE LABS
Security Products for the Automotive Industry

- **Implements everything talked about today**
  - Software emulated SHE+
  - Implemented in C with no dependencies
- **Included in free CANPico MicroPython firmware**
  - The CANPico board is the Canis Labs platform for evaluation and prototyping
  - `HSM` and `CryptoCAN` classes
- **More features coming**
  - Key database with SHE+ load key transactions
  - Tool command-and-control messaging protocol (e.g. including key distribution)

# Further information

CANIS
AUTOMOTIVE LABS
Security Products for the Automotive Industry

Canis Labs white paper
“Encryption on CAN Bus”

NDIA GVSETS 2022 paper
“Defending CAN Bus”

**canislabs.com/cryptocan**