Session Key Distribution Made Practical for CAN and CAN-FD Message Authentication

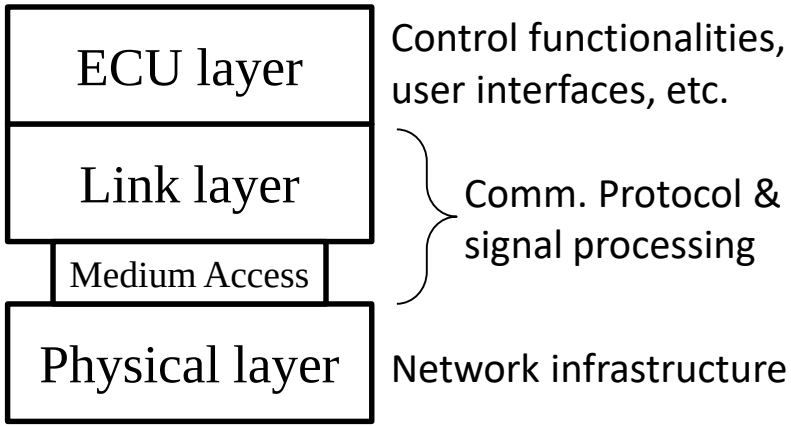
Yang Xiao, Shanghao Shi, Ning Zhang, Wenjing Lou, Y. Thomas Hou

Presenter: Yang Xiao <xiaoy@vt.edu>

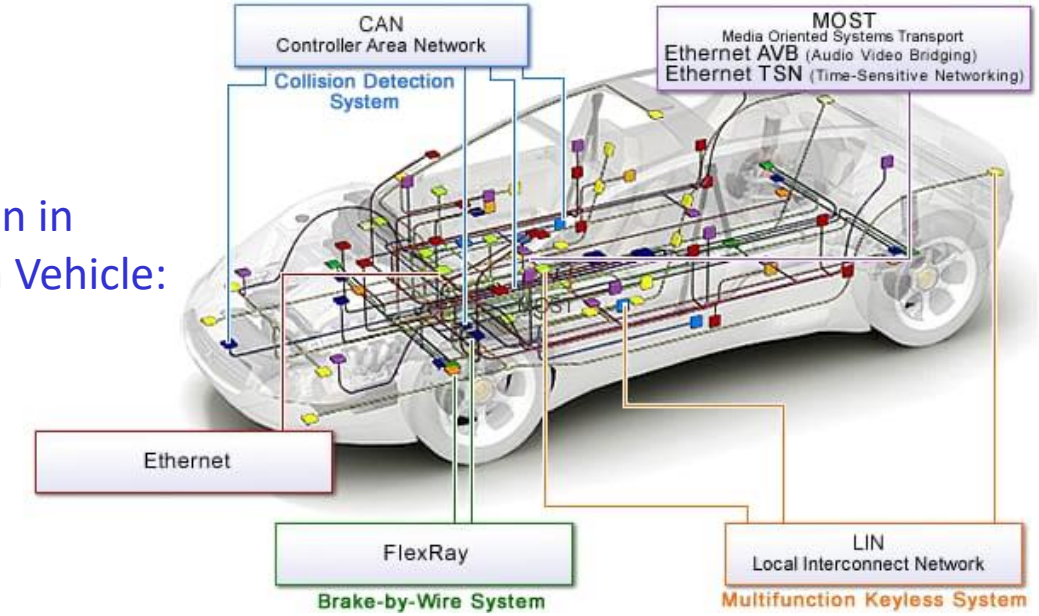
Controller Area Network (CAN)

- Automotive Communication Networks
  - For enabling in-vehicle communication between Electronic Control Units (ECUs)

Layered Model:



Adoption in Modern Vehicle:

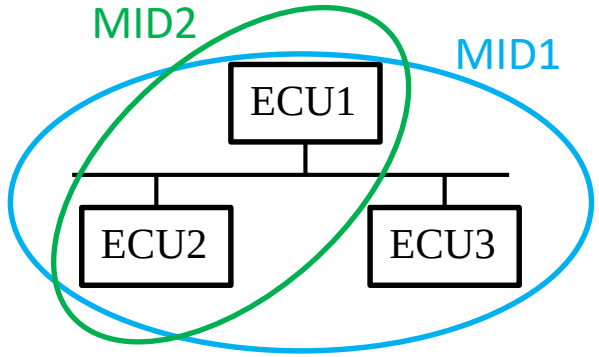


- Controller Area Network (CAN bus)
  - Used for the control of powertrain or other safety-critical subsystems
  - CAN Flexible Data-rate (CAN FD) – to accommodate for faster data transmission

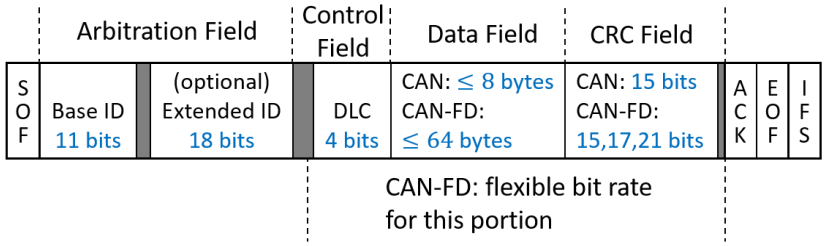
Image source: https://www.renesas.com/us/en/solutions/automotive/technology/networking-solutions.html

Basics of CAN and CAN FD Messaging

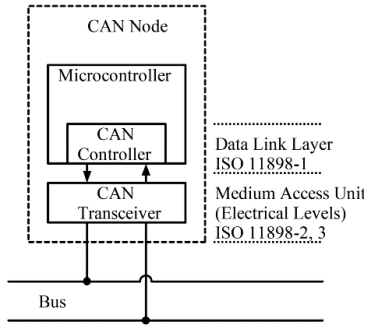
- Broadcast-and-Subscribe Messaging Paradigm
  - Only message ID, no sender or receiver ID
- Message Arbitration
  - Lower message ID → higher priority
- Physical Layer
  - CAN: fixed data rate (eg., 500kbs)
  - CAN FD: flexible data rate for data + CRC fields



ECU-Message ID Subscription Example



CAN/CAN FD Data Frame Format



CAN/CAN FD Node Architecture

Image credit: https://en.wikipedia.org/wiki/CAN_bus

Attack on Vehicles

- Gain Access to Internal Control of Vehicle
  - Through the OBD-II port or exposed wired/wireless interfaces
  - → Eavesdrop
  - → Spoof messages to critical ECUs
  - → Knocking a ECU offline



OBD-II port

Hacker Finds He Can Remotely Kill Car Engines After Breaking Into GPS Tracking Apps

"I can absolutely make a big traffic problem all over the world," the hacker said.

ANDY GREENBERG SECURITY 08.01.2016 03:30 PM

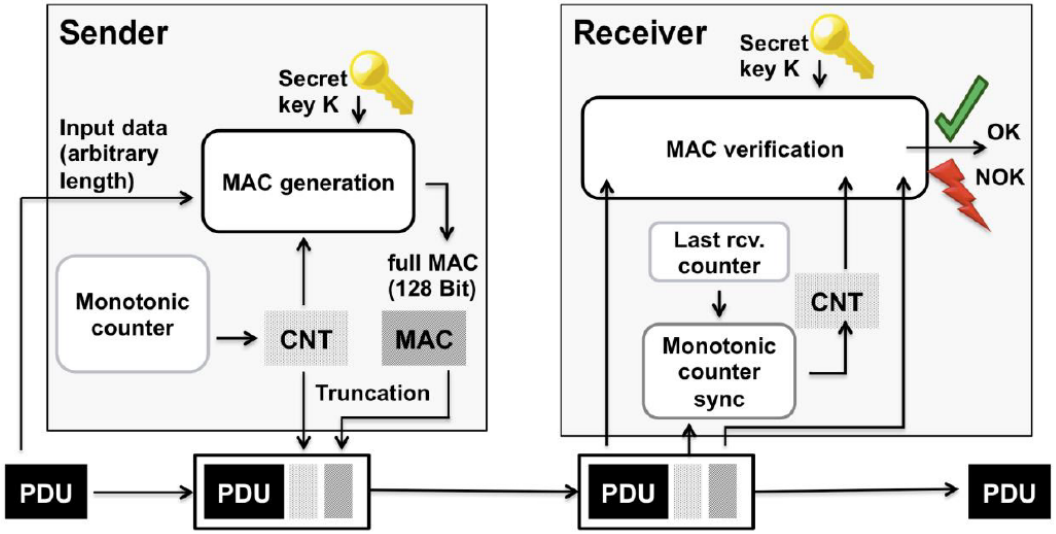
The Jeep Hackers Are Back to Prove Car Hacking Can Get Much Worse

After sparking a 1.4 million vehicle Chrysler recall, the security researchers offer a new lesson: It could have been---and could still be---much worse.

- Root Cause
  - No security mechanism built in the communication protocol
  - "Security by obscurity" is no longer safe for in-car systems

AUTOSAR Specifications on Secure Communication

- Security Goal Specified in AUTOSAR-SecOC
  - ECU entity authentication
  - Message authentication
    - Freshness verification
      (msg counter for replay attack resistance)
    - Each message ID is assigned a MAC key
  - Cryptography (symmetric)
    - 128-bit keys
    - 64-bit MACs



Message authentication with freshness verification [SecOC 4.2.2]

- What is Missing – Specification on Session Key Establishment for MAC Purposes
  - Critical to real-world deployment
  - → **Goal of this work**

[SecOC 4.2.2] AUTOSAR, "AUTOSAR release 4.2.2: Specification of module secure onboard communication," 2017

Selected Previous Work on Authenticated CAN Messaging

	Publication	Assumptions	Key Establishment Highlights	Message Authentication Highlights
[NLJ08]	VTC'08	Pre-distr. pairwise MAC keys in place		Delayed auth. with compound MAC, CBC-MAC
[OY+08]	GLOBECOM'08	Use TPM (remote attest); Master ECU	Pairwise ECU key via master-aided KPS	AES-CMAC for enc. and HMAC for MAC
[SR+11]	VTC'11	Key master ECU	"Asymmetric use of symmetric keys"	
CANAuth [HSV11]	ECRYPT-LC'11	Send MAC with CAN+; Pre-distr. LT keys	LT keys → session key	Mixed MAC (M-MAC)
LiBrA-CAN [GM+12]	CANS'12	Send MAC with CAN+; Need a master ECU	Master splits keys; each t-group share a key	M-MAC and Linearly Mixed MAC (LM-MAC)
MaCAN [HRS12]	ESCAR'12	Key server and time server	Pairwise ECU key via server-aided KPS	4-byte CMAC in the same data payload
LCAP [HF12]	ESCAR'12	5 new msg IDs for each S-R pair	Handshakes for each S-R pair → key	2-byte magic number for auth.
CaCAN [KM+14]	ESCAR'14	Centralized authentication; monitor node	Challenge-response authentication	M-MAC
VeCure [WS14]	IOT'14	Grouping ECUs based on trust levels	Each group share one secret key	A 2-part MAC scheme with offline processing
vatiCAN [NR16]	CHES'16	Pre-distr. LT key for each msg ID	Directly using LT keys for message auth (msg group key)	HMAC a subsequent data frame
LeiA [RG16]	ESORICS'16	Pre-distr. LT key for each msg ID	LT key derives session keys (msg group key)	MAC in a subsequent data frame
VulCAN [VMP17]	ACSAC'17	TPM are used for software integrity	Key delivery via attested secure channel (msg group key)	Following LeiA and vatiCAN
[FE+17]	WINCOM'17	PKI pre-stablished	PKI → session keys	
Mauth-CAN [JK+20]	TVT'20	Need a master ECU called "Authenticator"	Master-aided derivation; diff key for same msg at diff ECU	Only master checks MAC, reporting if no match

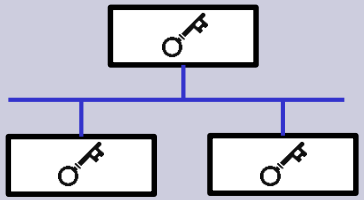
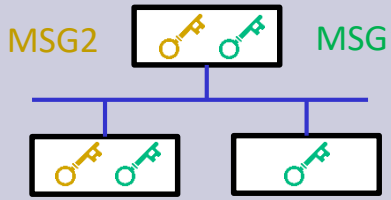
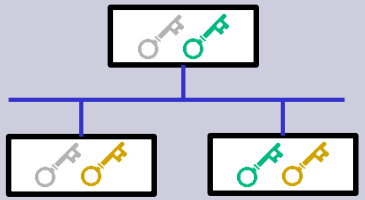
Relevant & good theory

Relevant & AUTOSAR-compliant

[NLJ08] D. K. Nilsson, U. E. Larson, and E. Jonsson, "**Efficient in-vehicle delayed data authentication based on compound message authentication codes**," in 2008 IEEE 68th Vehicular Technology Conference (**VTC 2008**)
[OY+08] H. Oguma, A. Yoshioka, M. Nishikawa, R. Shigetomi, A. Otsuka, and H. Imai, "**New attestation based security architecture for in-vehicle communication**," in 2008 IEEE Global Telecommunications Conference (**GLOBECOM 2008**)
[SR+11] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille, and D. Scheuermann, "**Car2X communication: securing the last meter-a cost-effective approach for ensuring trust in car2x applications using in-vehicle symmetric cryptography**," in 2011 IEEE Vehicular Technology Conference (**VTC 2011**)
[HSV11] Van Herrewege, D. Singelee, and I. Verbauwhede, "**CANAuth-a simple, backward compatible broadcast authentication protocol for can bus**," in ECRYPT Workshop on Lightweight Cryptography (**ECRYPT-LC 2011**)
[GM+12] B. Groza, S. Murvay, A. Van Herrewege, and I. Verbauwhede, "**LiBrA-CAN: a lightweight broadcast authentication protocol for controller area networks**," in International Conference on Cryptology and Network Security (**CANS 2012**)
[HRS12] O. Hartkopp, C. Reuber, and R. Schilling, "**MaCAN-message authenticated CAN**," in 10th Int. Conf. on Embedded Security in Cars (**ESCAR 2012**)
[HF12] A. Hazem and H. Fahmy, "**LCAP-a lightweight can authentication protocol for securing in-vehicle networks**," in 10th Int. Conf. on Embedded Security in Cars (**ESCAR 2012**)
[KM+14] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, "**CaCAN-centralized authentication system in CAN (controller area network)**," in 14th Int. Conf. on Embedded Security in Cars (**ESCAR 2014**)
[WS14] Q. Wang and S. Sawhney, "**VeCure: A practical security framework to protect the can bus of vehicles**," in 2014 International Conference on the Internet of Things (**IOT 2014**)
[RG16] A. I. Radu and F. D. Garcia, "**LeiA: A lightweight authentication protocol for CAN**," in European Symposium on Research in Computer Security (**ESORICS 2016**)
[NR16] Nürnberger, Stefan, and Christian Rossow. "**–vatican–vetted, authenticated can bus.**" International Conference on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2016. (**CHES 2016**)
[FE+17] S. Fassak, Y. E. H. El Idrissi, N. Zahid, and M. Jedra, "**A secure protocol for session keys establishment between ECUs in the CAN bus**,"in 2017 International Conference on Wireless Networks and Mobile Communications (**WINCOM 2017**)
[VMP17] Van Bulck, Jo, Jan Tobias Mühlberg, and Frank Piessens. "**VulCAN: Efficient component authentication and software isolation for automotive control networks.**" in Proceedings of the 33rd Annual Computer Security Applications Conference (**ACSAC 2017**)
[JK+19] Hyo Jin Jo, Jin Hyun Kim, Hyon-Young Choi, Wonsuk Choi, Dong Hoon Lee. "MAuth-CAN: Masquerade-Attack-Proof Authentication for In-Vehicle Networks." IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 69, NO. 2, FEBRUARY 2020

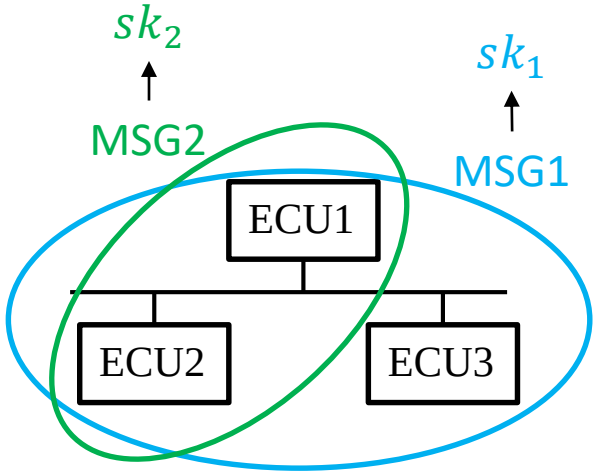
On Key Management and Establishment

AUTOSAR Compliant

	One Key for All	One Key for Each MSG ID	ECU-pairwise Keys
			
Security Goal:	Auth. to send	Auth. to send a specific message	Entity auth.
Session Key Storage per ECU:	One	# of MSG subscribed	ECU population
Key Establishment Styles [Communication Complexity] **Key Agreement:**	High	(No LT key needed) High	High Eg: **LCAP** [HF12]
Key Derivation:	Low	(Msg-specific LT keys needed) Low Eg: **vatiCAN** [NR16], **LeiA** [RG16]	Low
Key Distribution:	Low Eg: **CANAuth** [HSV11]	(ECU-specific LT keys needed) Medium Eg: **VulCAN** [VMP17] **Our proposal also fits here**	High Eg: **LiBrA-CAN** [GM+12], **MaCAN** [HRS12]

Our Proposed Key Management Architecture

- System Model
  - N ECUs, M message IDs
  - ECU i has a Subscription List ($\mathbf{SL}_i$) of message IDs
  - **Goal:** All ECUs subscribing message j shall get a shared session key sk_j
- Threat Model
  - Message eavesdropping, tampering, spoofing and replaying in the bus



- **Practical Requirements for Key Establishment**
  - **R1:** Lightweight Computation and Storage
  - **R2:** Communication Efficiency
  - **R3:** AUTOSAR-compliant Security
  - **R4:** Flexibility with On-demand ECU

Key Establishment

Key Distribution



Our Proposed Key Management Architecture

- Key Server (KS)
  - Shares a long-term key ek_i with every ECU i
  - Keeps a Required ECU List (**REL**) and a Required Message List (**RML**)
  - To generate 128-bit session keys $sk_1, \ldots, sk_M$
  - To maintain the 64-bit system epoch e
- Key Distribution Protocols

Primitives:

KDC (Key Distribution Center)

SKT (Secret-sharing-based Key Transfer)

Adapted for session key distribution in CAN/CAN-FD:

SKDC Protocol

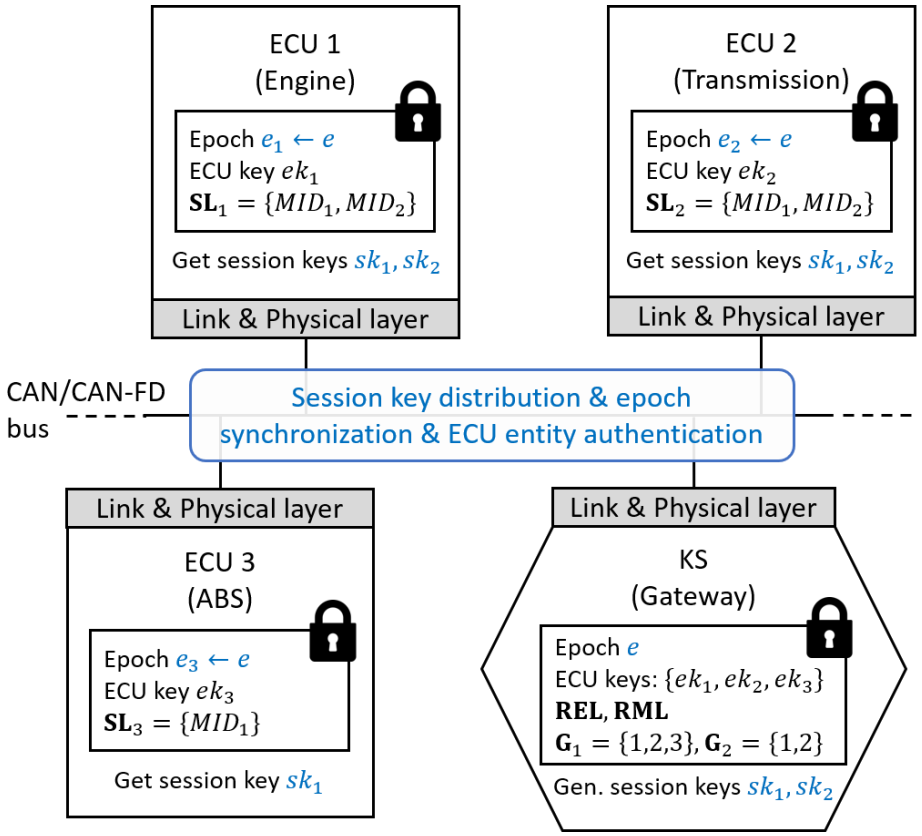
SSKT Protocol

Baseline

Novel Scheme

(for better communication efficiency)

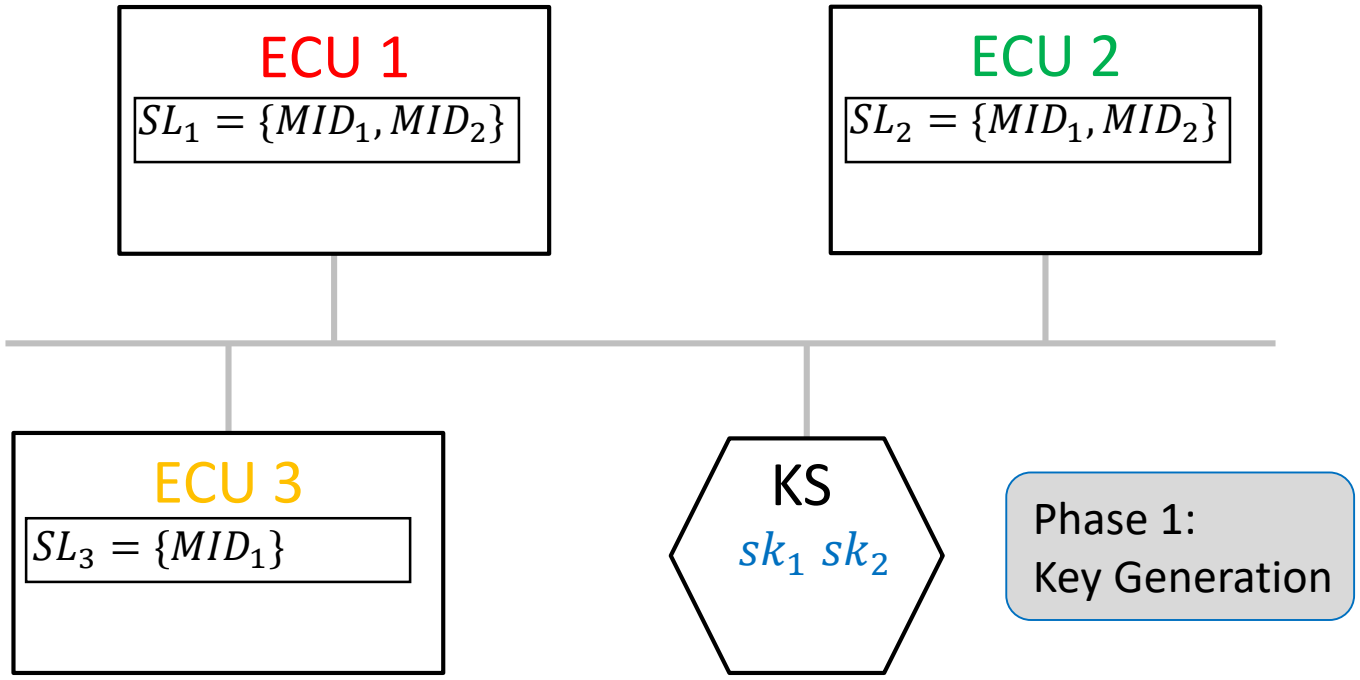
Example:
ECU 1, 2, 3 subscribe msg 1
ECU 1, 2 subscribes msg 2



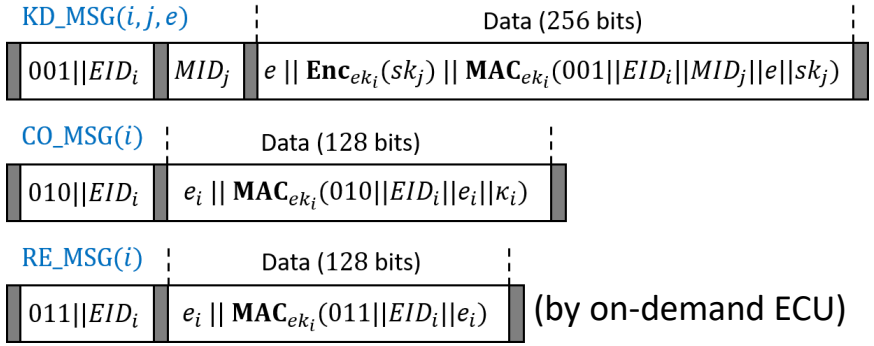
Baseline: the **SKDC** Protocol

- Highlights
  - KS uses ek_i as key-encryption key (KEK) to encrypt each session key to each ECU i
  - Epoch e for freshness; MAC for verification

- Workflow (Example)



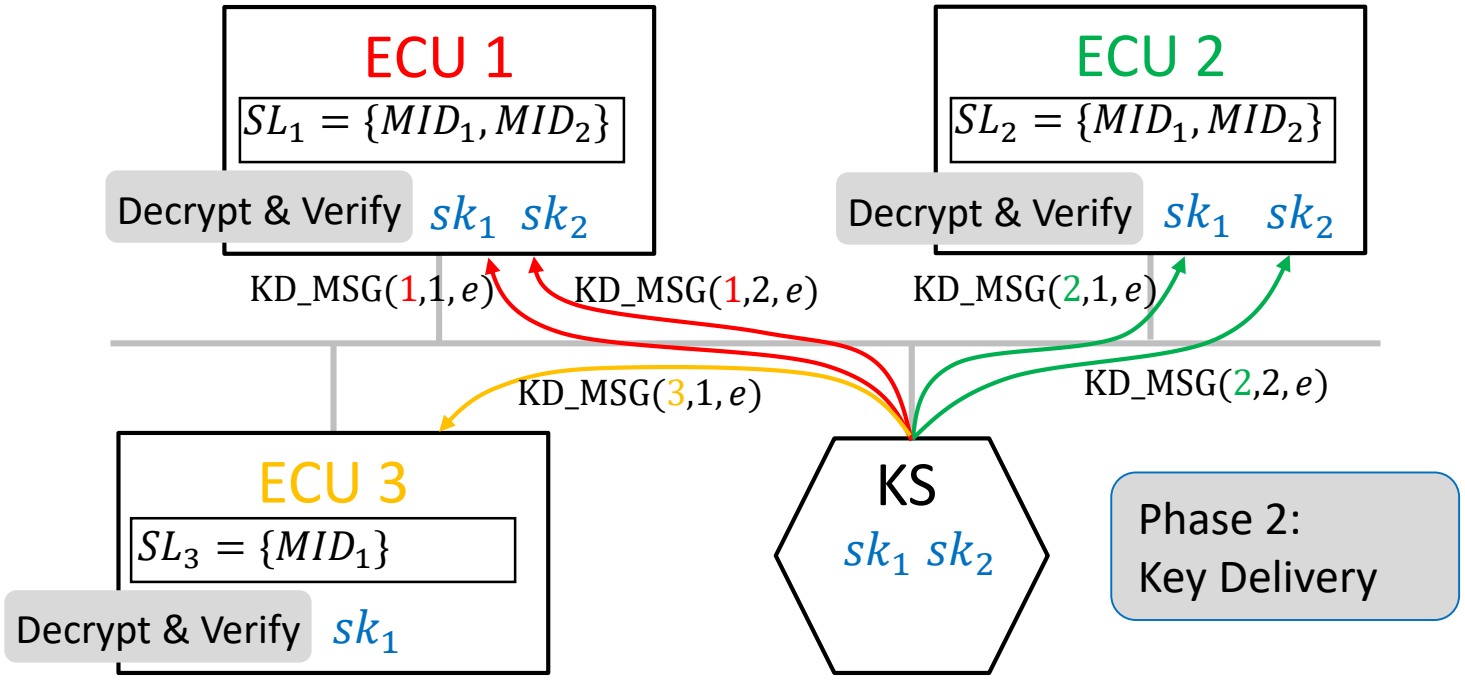
SKDC Protocol Message Formats



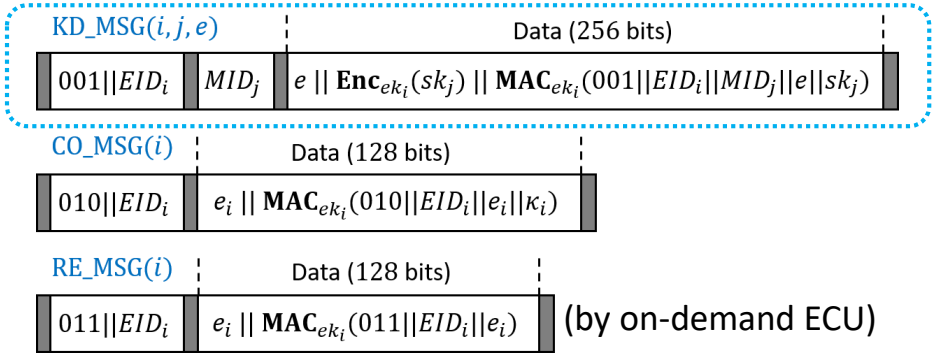
Baseline: the **SKDC** Protocol

- Highlights
  - KS uses ek_i as key-encryption key (KEK) to encrypt each session key to each ECU i
  - Epoch e for freshness; MAC for verification

- Workflow (Example)



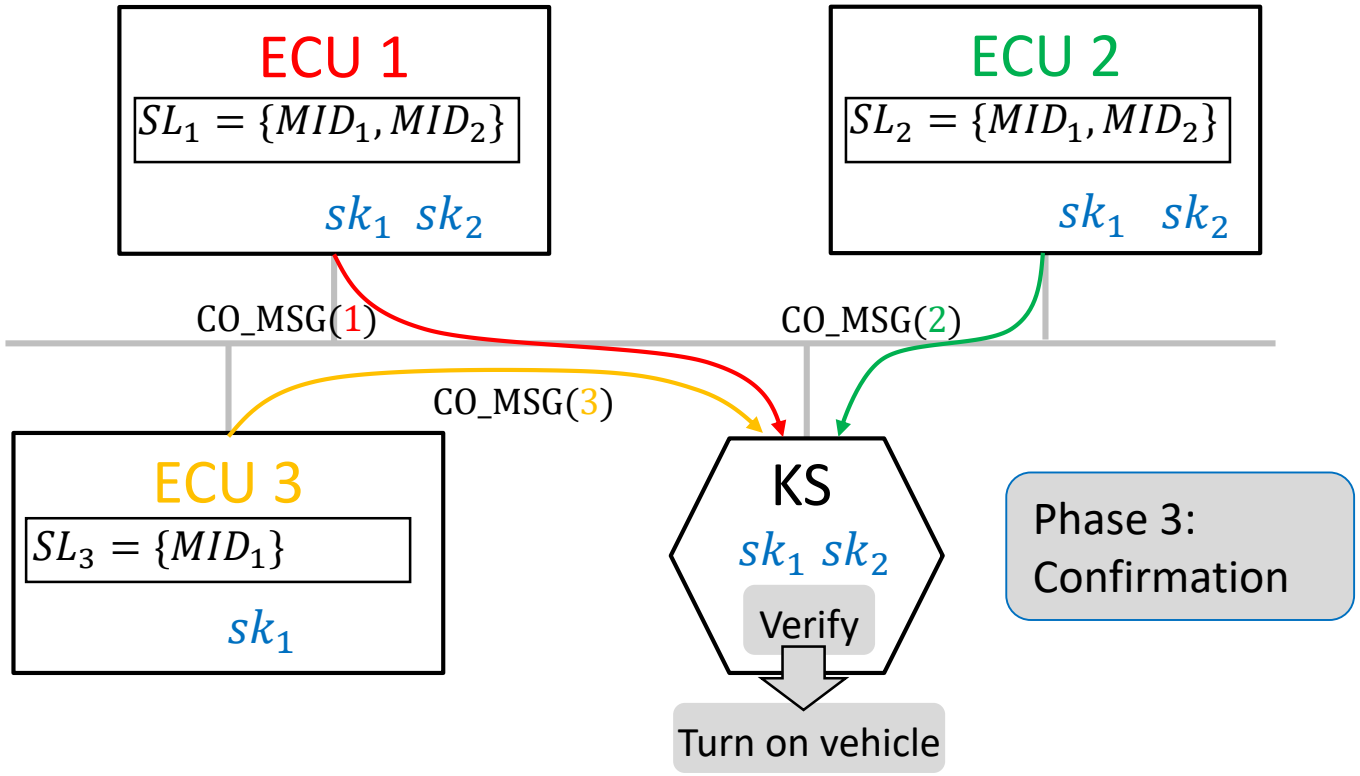
SKDC Protocol Message Formats



Baseline: the SKDC Protocol

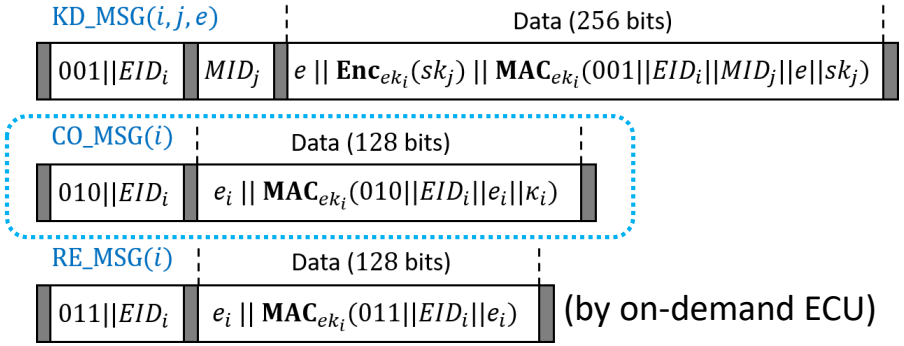
- Highlights
  - KS uses ek_i as key-encryption key (KEK) to encrypt each session key to each ECU i
  - Epoch e for freshness; MAC for verification

- Workflow (Example)



Phase 3: Confirmation

SKDC Protocol Message Formats



- Protocol message is sent in separate CAN/CAN-FD frames if payload exceeds frame limit.
- On-demand ECU sends RE_MSG during driving for requesting session keys.

Alternative: Secret Sharing-based Key Transfer (SKT)

- **Proposed in [HL10]:** Based on Shamir's Secret Sharing ((k, n)-threshold scheme)
- **Assumption:** Each member i keeps a secret pair (x_i, y_i) with KS

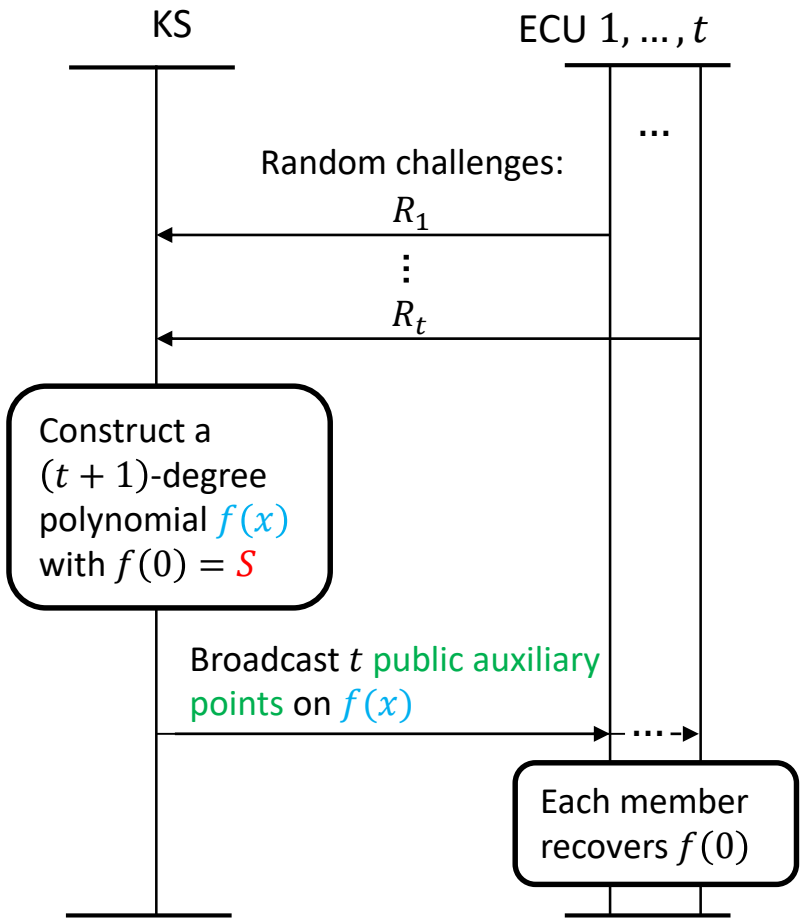
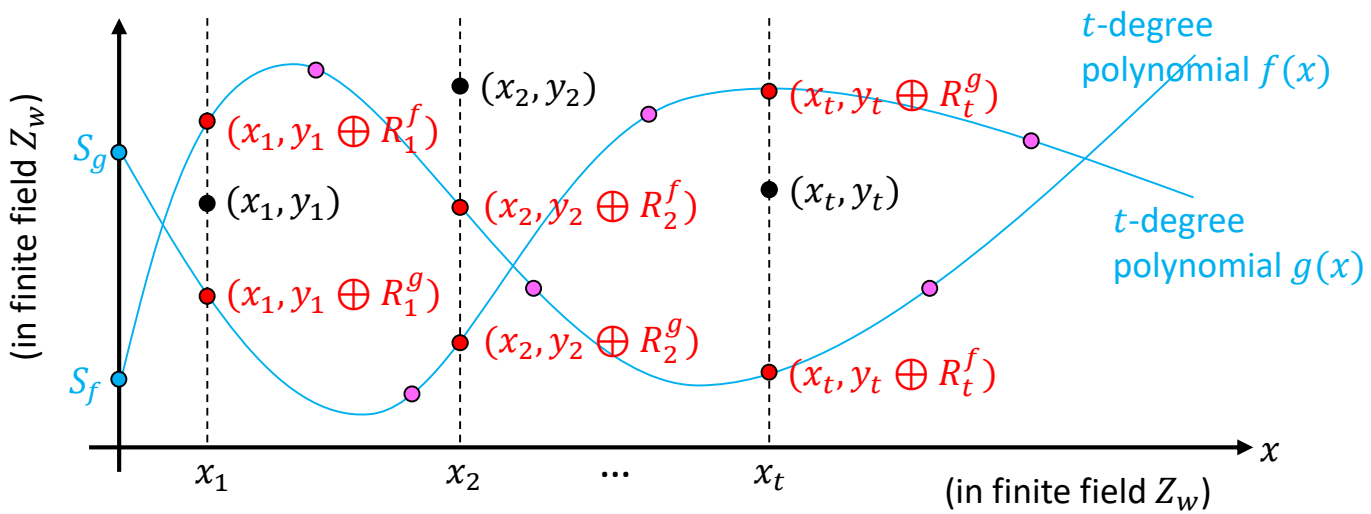


Illustration: transferring two secrets S_f and S_g



- The key $S^f = f(0)$ can be recovered independently by each member i, via **Lagrange interpolation** on $t+1$ points—the t public auxiliary points and its secret point $(x_i, y_i \oplus R_i^f)$. Abbreviate them as $\{(u_i, v_i)\}_{i=1 \to t+1}$, then:

$$f(0) = \sum_{m=1}^{t+1} v_m \left(\prod_{n=1, n \neq m}^{t+1} \frac{u_n}{u_n - u_m} \right)$$

[HL10] L. Harn and C. Lin, "Authenticated group key transfer protocol based on secret sharing," IEEE Transactions on Computers, vol. 59, no. 6, pp. 842–846, 2010.

SKT Pros and Cons

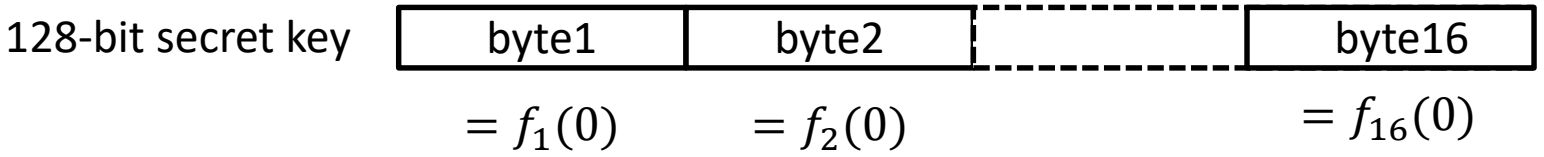
- SKT's Advantage – Communication efficiency
  - Broadcast nature
  - Good for our automotive scenario

- SKT's Disadvantage – All About Finite Field Arithmetic
  - For the finite field Z_w used, SKT needs $w = pq$, p and q needs to be large primes
    - For resisting the insider spoofing attack (by impersonating a victim member, attacker deduces the victim's secret pair through multiple sessions of operation)
    - Security relies on the intractability of large number factorization
  - w should be large
    - Thousands of bits, akin to asymmetric crypto
    - → **Impractical for resource-constrained ECUs**

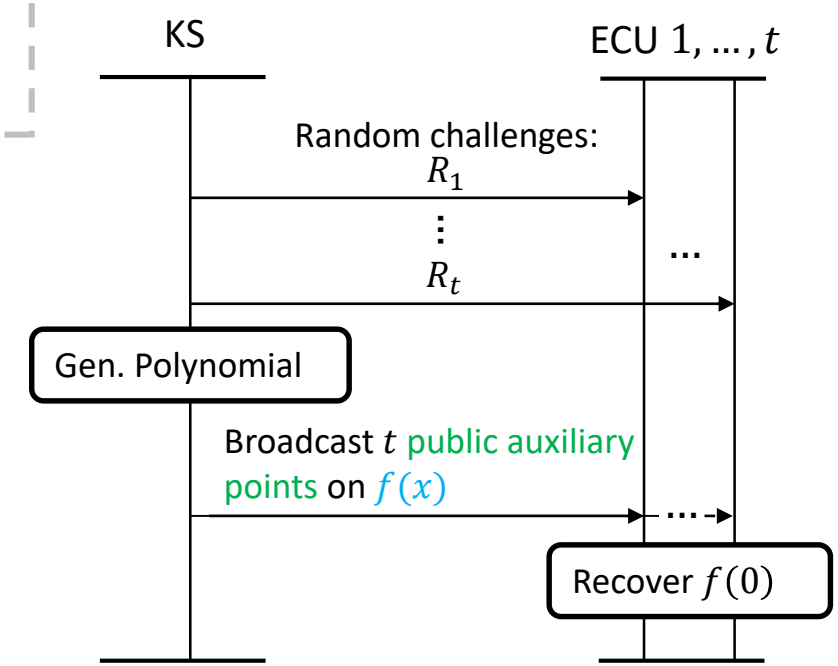
Adapting SKT into SSKT for Practical Session Key Distribution

- Byte-wise Operation in $GF(256)$
  - 1) to reduce computation complexity
  - 2) to cope with integer size limitations of ECU processor
  - 3) ECU population in a message group is small in practice
  - Each byte of a secret key (16 bytes in total) is associated with a polynomial

- We need to assume group size < 128 to use byte-wise operation in $GF(256)$.
- For future work that considers bigger group size, a bigger finite field (and thus more power processor) should be used.

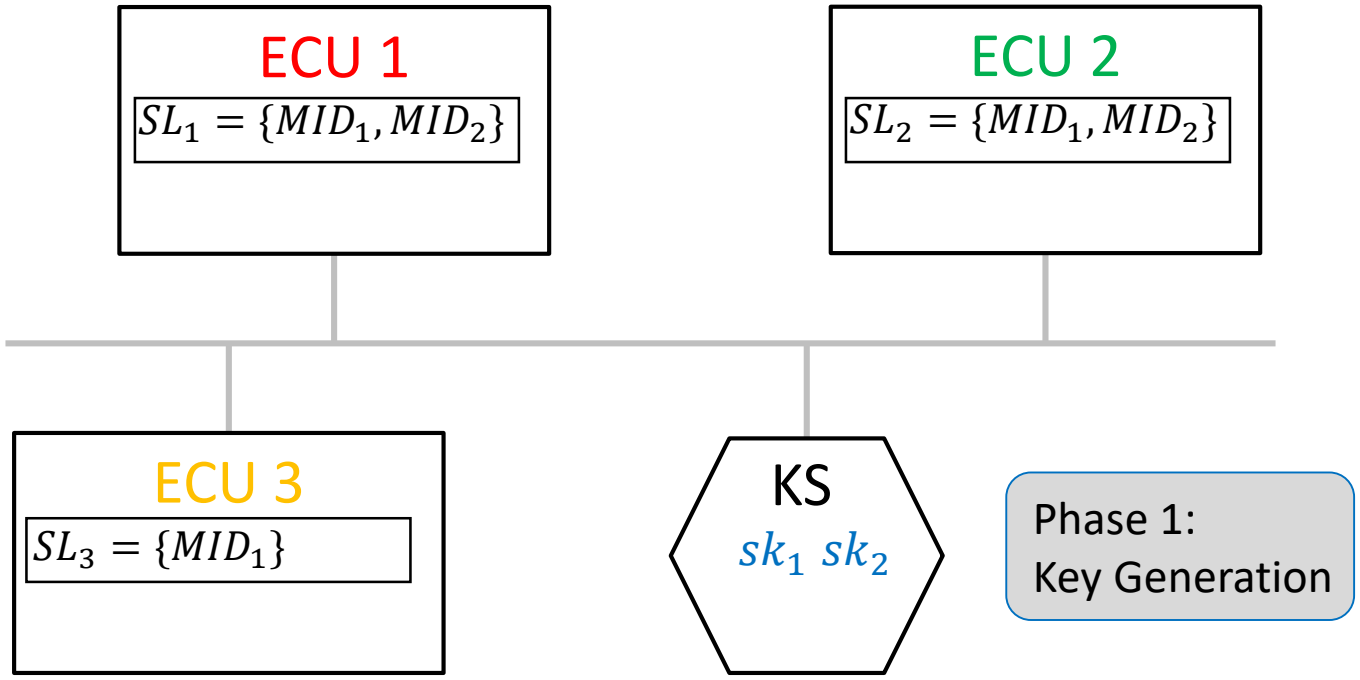


- Addressing Insider Attack from Server End
  - By having KS generate random challenges and distribute them to ECUs, not the other way
  - KS and ECUs derive secret offsets from the challenges for ensuing Polynomial operations

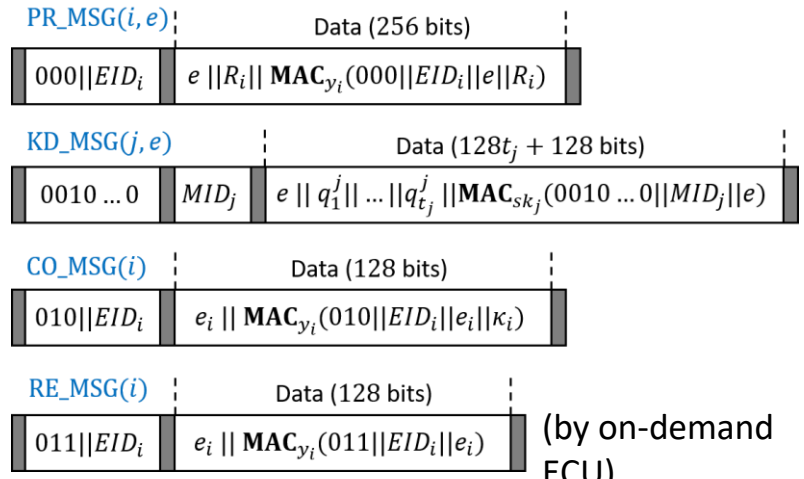


SSKT Workflow

- Highlights
  - Long-term key pair $ek_i = (x_i, y_i)$
  - ECU recovers session key segments by Lagrange polynomial interpolation in $GF(256)$
- Workflow (Example)



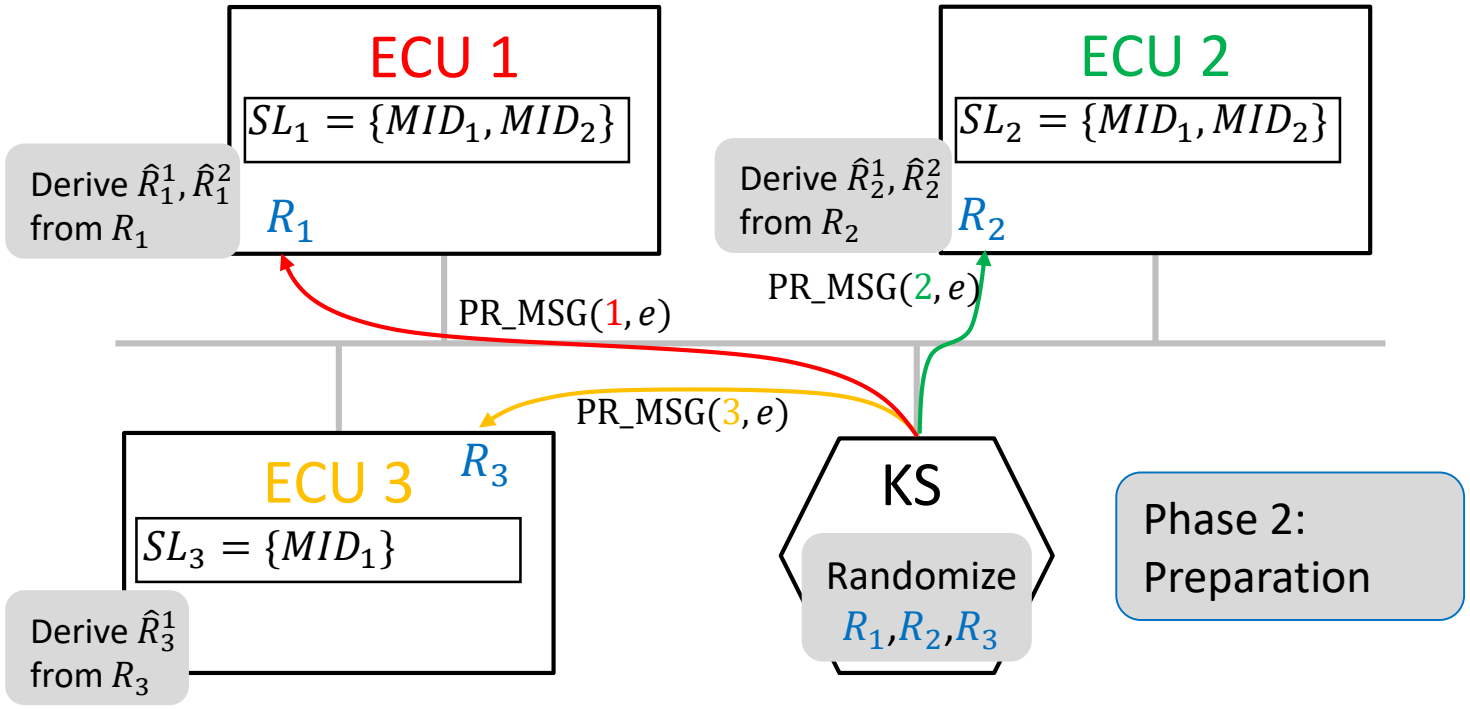
SKDC Protocol Message Formats



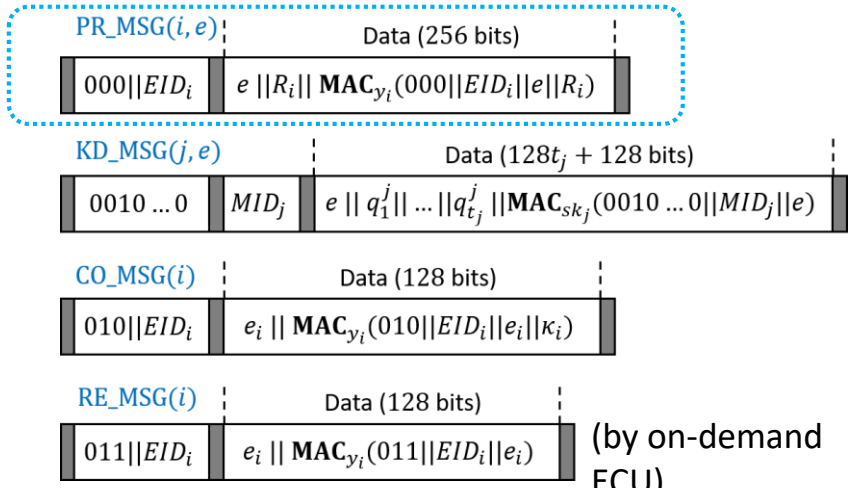
(by on-demand ECU)

SSKT Workflow

- Highlights
  - Long-term key pair $ek_i = (x_i, y_i)$
  - ECU recovers session key segments by Lagrange polynomial interpolation in $GF(256)$
- Workflow (Example)



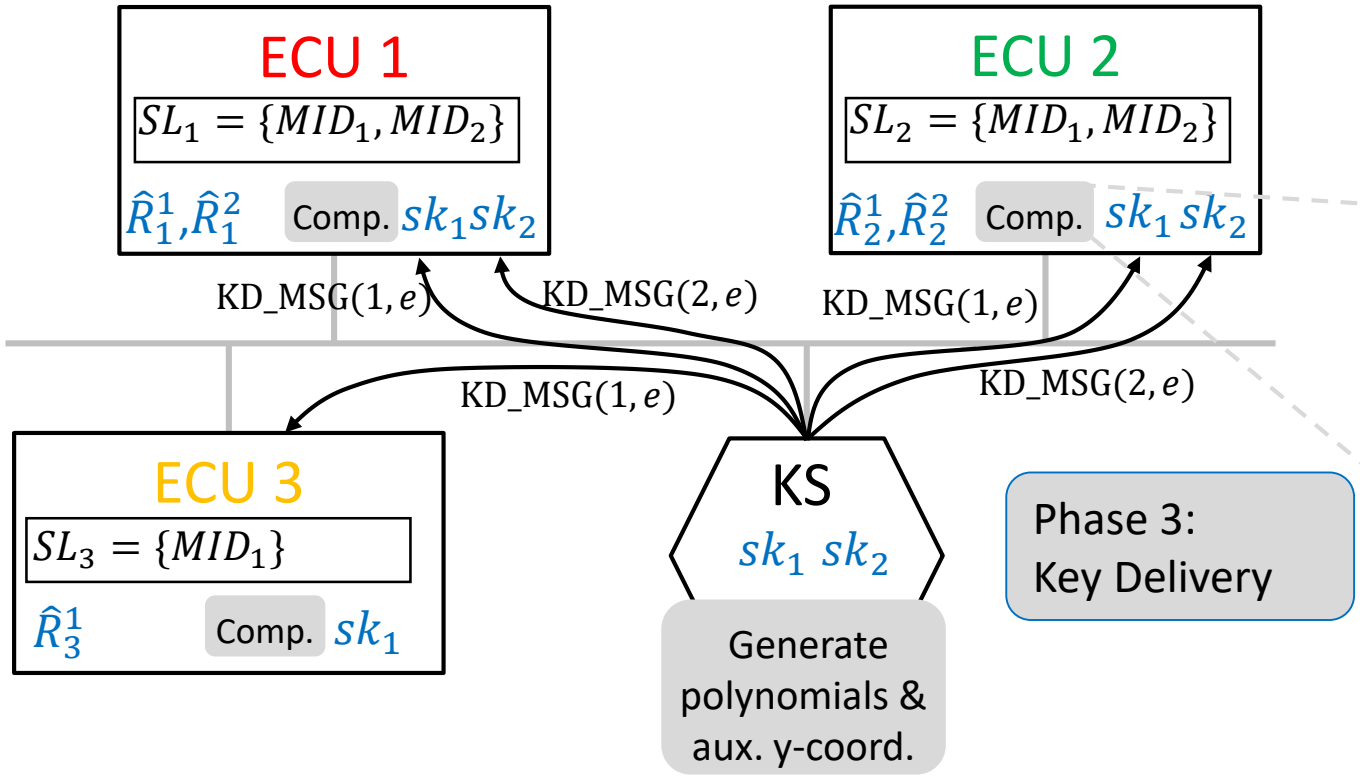
SKDC Protocol Message Formats



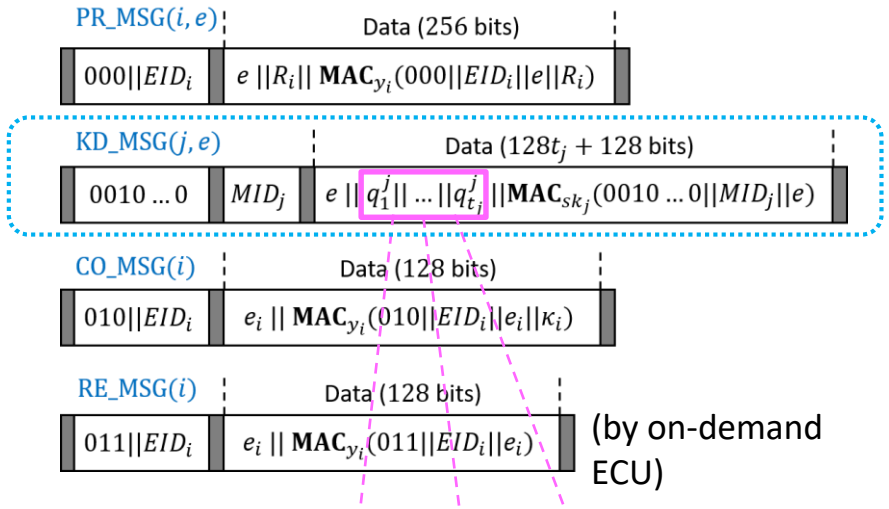
(by on-demand ECU)

SSKT Workflow

- Highlights
  - Long-term key pair $ek_i = (x_i, y_i)$
  - ECU recovers session key segments by Lagrange polynomial interpolation in $GF(256)$
- Workflow (Example)



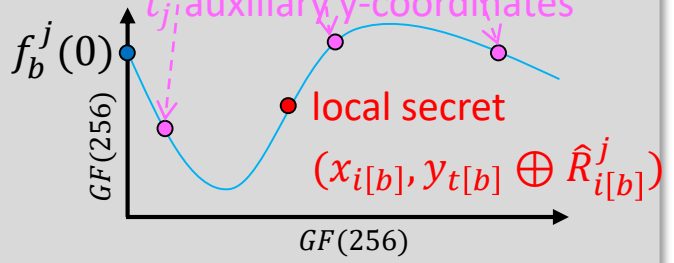
SKDC Protocol Message Formats



(by on-demand ECU)

ECU i upon receiving KD_MSG$(2, e)$:

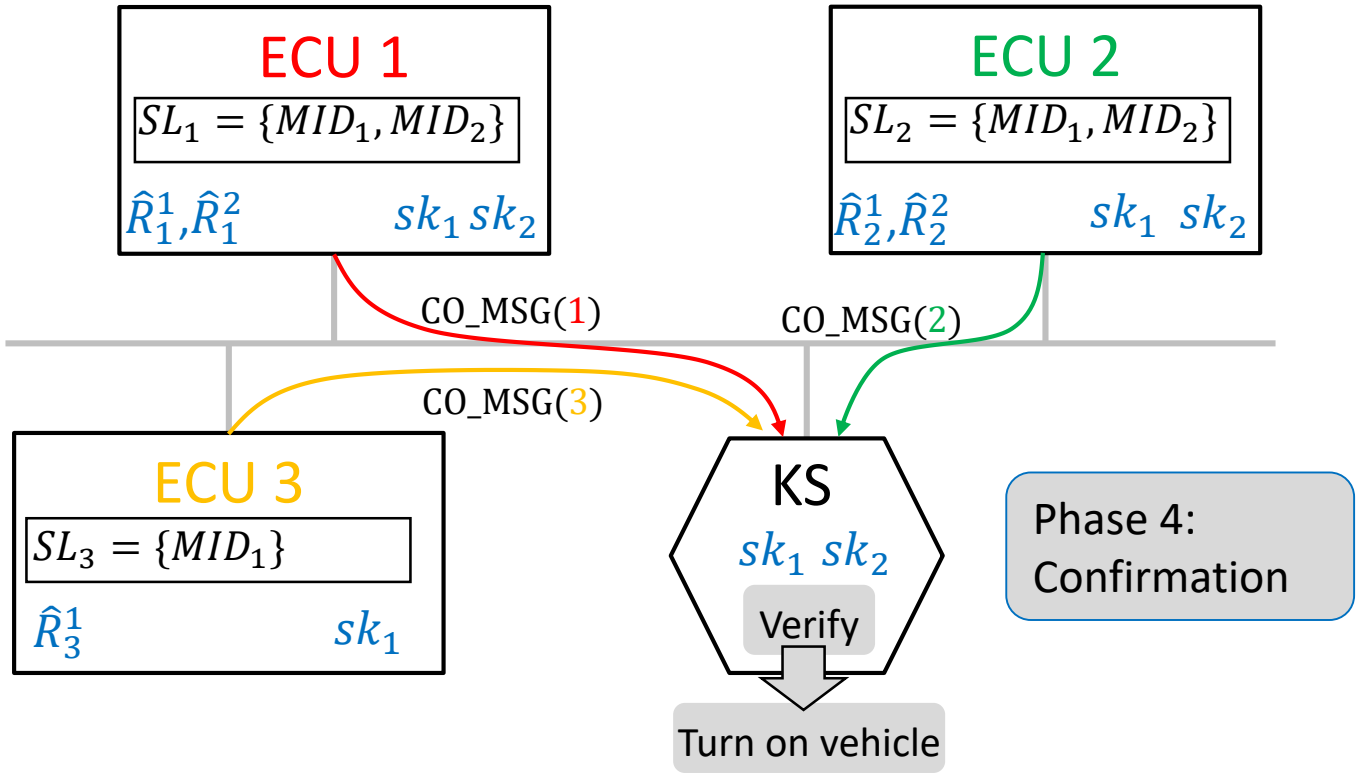
For $b = 1 \rightarrow 16$, interpolate $f_b^j(0)$:



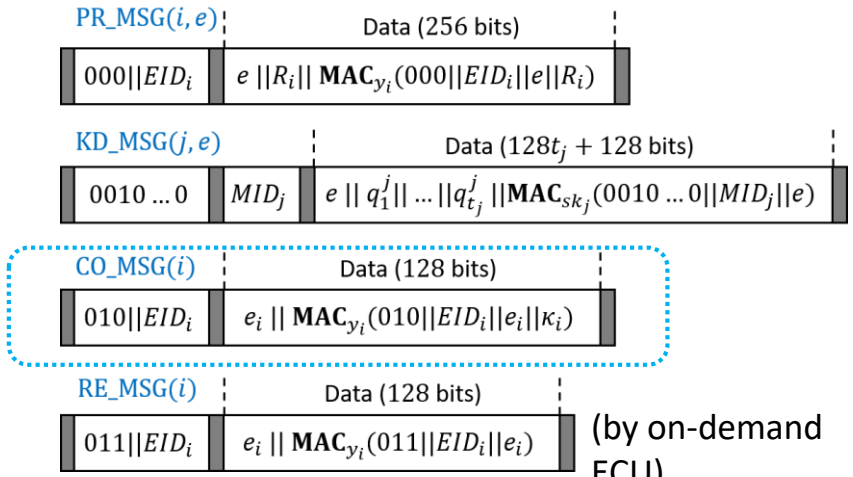
Then $sk_j = f_1^j(0)||\ ...\ ||f_{16}^j(0)$
& verify with MAC

SSKT Workflow

- Highlights
  - Long-term key pair $ek_i = (x_i, y_i)$
  - ECU recovers session key segments by Lagrange polynomial interpolation in $GF(256)$
- Workflow (Example)



SKDC Protocol Message Formats



(by on-demand ECU)

- Protocol message is sent in separate CAN/CAN-FD frames if payload exceeds frame limit.
- On-demand ECU sends RE_MSG during driving for requesting session keys (same as SKDC).

Security Analysis

- Session Key Correctness
  - Reduce to the security of the MAC algorithm
  - Replay attack resistance ← epoch mechanism
- Session Key Confidentiality
  - For SKDC:
    - Reduces to the confidentiality of ECU keys and security of the encryption algorithm
    - 2^{127} MAC evaluations for a successful attack
  - For SSKT:
    - reduces to the confidentiality of ECU key pairs and information theoretic security of secret sharing
    - $\frac{1}{2}\left(2^8 - t_j\right)^{16} \in (2^{111}, 2^{127})$ MAC evaluations for a successful attack
- ECU Integrity and Liveness
  - Reduces to security of MAC
  - ECU entity authentication and liveness detection ← confirmation phase

Implementation for CAN Bus Deployment

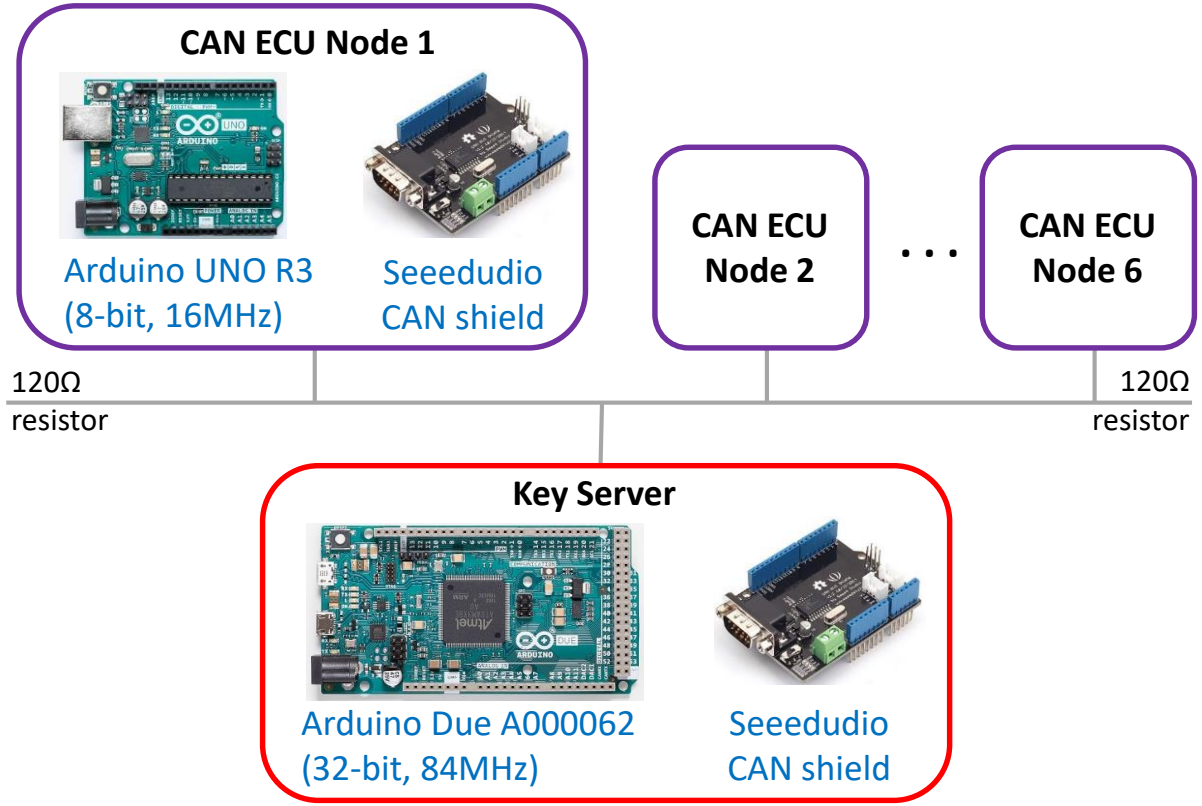


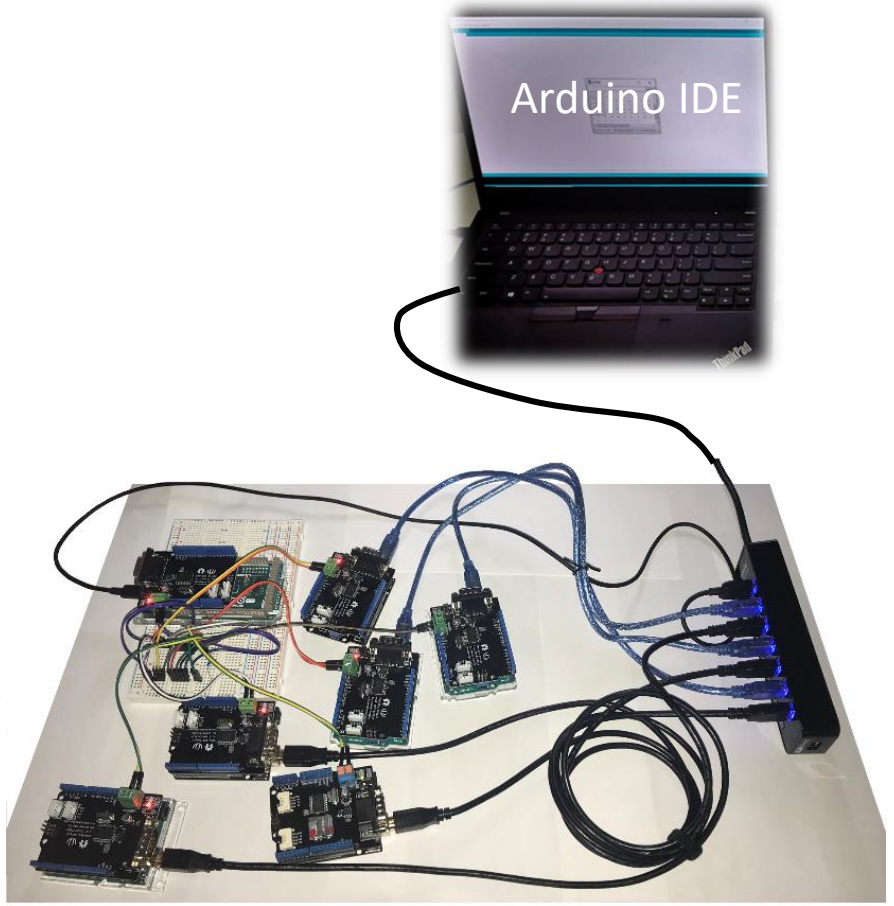
Code: github.com/yang-sec/CAN-SessionKey

- *Keyserver* and *node* Programs
  - Arduino IDE (C++)
  - Third-party libraries used
    - Arduino Cryptography Library (rweather.github.io/arduinolibs/crypto.html)
    - → AES for encryption, BLAKE2 for hash
    - Seeed Studio CAN Bus Shield (github.com/Seeed-Studio/CAN_BUS_Shield)
- Optimization for SSKT
  - 1. Three 16×16 lookup tables for $GF(256)$ arithmetic
    - Inverse table
    - Exponentiation table and logarithm table (for realizing multiplication)
  - 2. Pre-computing Lagrange coefficients
    - $f_b^j(0) = \sum_{m=1}^{t_j+1} v_m \left(\prod_{n=1, n\neq m}^{t_j+1} \frac{u_n}{u_n - u_m} \right)$
    - Traded memory for speed

Test Platform with CAN Bus

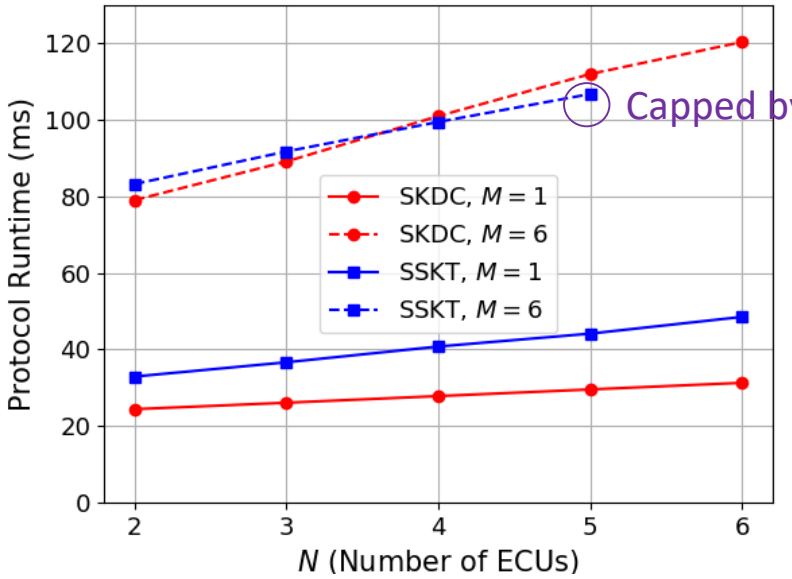
- Setup
  - N ECUs, each subscribes to all M MIDs
  - $N \in \{1,2,3,4,5,6\}$, $M \in \{1,6\}$
  - Data collected using serial terminals @ Arduino IDE





Overall Protocol Runtime

- Protocol Message Timing
  - We minimize the inter-KD_MSG delay (*KdDelay*) at KS, while ensuring ECUs receive every message in full for system stability
  - When M=6:
    - SKDC: *KdDelay* = 6.8ms for all N
    - SSKT: *KdDelay* = {6.5, 6.2, 5.5, 4.4}ms for N = {2, 3, 4, 5}
- Protocol Runtime Results for Distributed One Session Key (ms)



Capped by N=5 due to Uno's RAM limit

SSKT's advantage will continue scaling up for larger M and N, contributed by its better communication efficiency in the key delivery phase.

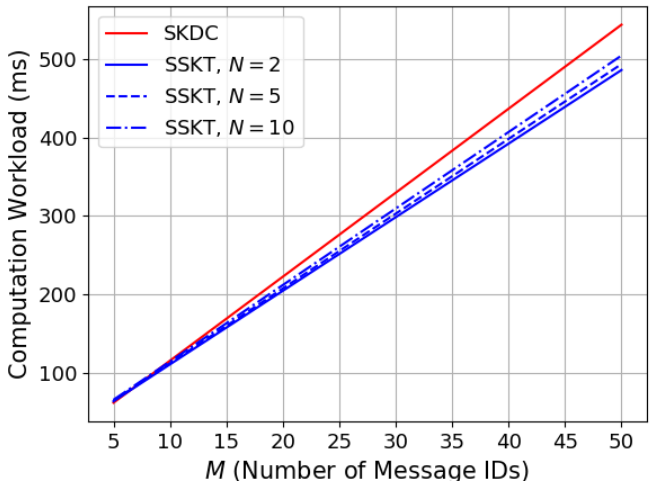
Performance Extrapolation – Computation Workload

- Single-operation Runtimes
  - Evaluated on normal ECU (not KS)
  - Used both Uno and Due for benchmarking

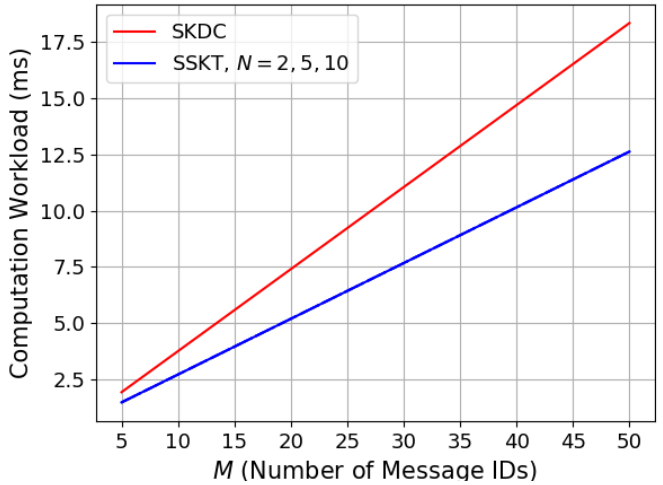
Operation	Uno	Due
AESSmall128-ECB Set Key	131.64	23.66
AESSmall128-ECB Encrypt (per byte)	42.40	7.06
AESSmall128-ECB Decrypt (per byte)	73.66	12.33
AESTiny128-ECB Set Key	9.98	1.25
AESTiny128-ECB Encrypt (per byte)	42.39	7.23
BLAKE2s Keyed Reset	3512.94	55.09
BLAKE2s Hash (per byte)	54.61	0.80
BLAKE2s Finalize	3508.25	53.14
Degree-2 Polynomial $f(0)$ Recovery (per byte)	10.40	~ 0
Degree-5 Polynomial $f(0)$ Recovery (per byte)	19.86	~ 0
Degree-10 Polynomial $f(0)$ Recovery (per byte)	33.56	~ 0

- Extrapolated ECU Computation Workload per Protocol Session

Arduino Uno as ECU



Arduino Due as ECU



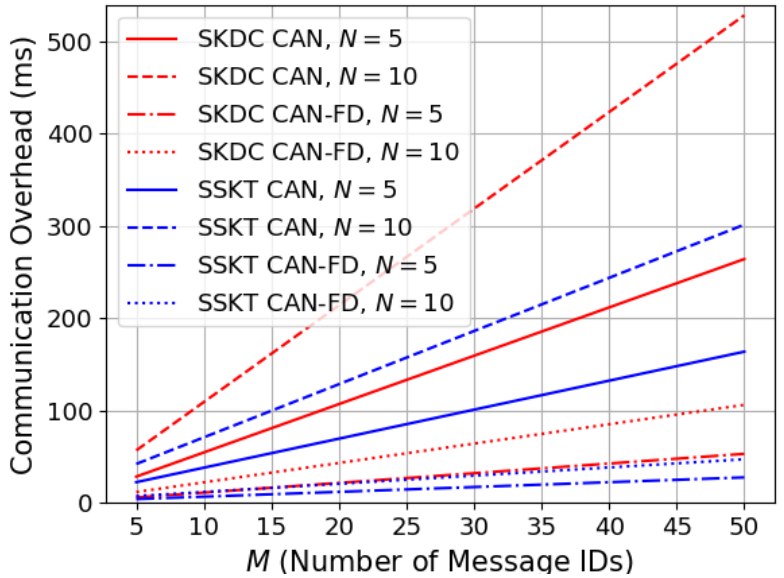
SSKT achieves better computation efficiency for larger M.
Tradeoff: RAM cost.

Performance Extrapolation – Communication Overhead

- Protocol Message Count per Protocol Session
  - Assume CAN-FD bit rate is 5 times of CAN's

		Message size (In CAN bits)	Message count
SKDC (CAN):	KD_MSG	524	MN
	CO_MSG	222	N
SKDC (CAN-FD):	KD_MSG	105	MN
	CO_MSG	60	N
SSKT (CAN):	PR_MSG	444	N
	KD_MSG	$262(1+N)$	M
	CO_MSG	222	N
SSKT (CAN-FD):	PR_MSG	86	N
	KD_MSG	avg. $60+39N$	M
	CO_MSG	60	N

- Extrapolated Communication Overhead per Protocol Session
  - Assume CAN bit rate is 500Kbs



SSKT achieves better communication efficiency.

Communication overhead is of more concern than computation workload, because it is largely capped by the bus infrastructure, little room for improvement.

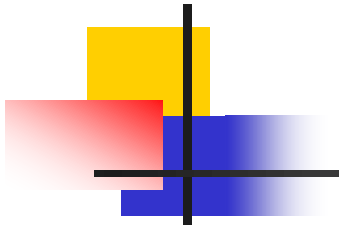
Conclusion

- Proposed a Key management Framework for ECU Message Authentication
  - AUTOSAR-compliant
  - Considered four practical requirements
  - Session key distribution protocols for CAN/CAN-FD
    - SKDC (baseline)
    - SSKT (secret-sharing-based, better communication efficiency)
    - → Both secure under common network-based attacks
  - Hardware implementation and evaluation, extrapolation analysis
    - Verifies SSKT's advantage in efficiency.

- Future Directions for Improvement SSKT
  - On memory saving
  - On scaling up network size (larger finite field)

Discussion & Future Directions (in detail)

- On Performance Bottleneck and Room for Improvement
  - Compared to computation workload, communication overhead is less flexible and left with limited room for improvement, since it is largely determined by the underlying bus
  - Thus we highlight SSKT's advantage in communication efficiency
- On Storage and Memory Cost
  - SSKT achieves superior computation efficiency using pre-computed intermediate results, which needs SRAM to store
  - Though SRAM is affordable nowadays, the tradeoff deserves more attention
- On Scalability and Future Work
  - $GF(256)$-arithmetic caps the network size by 128
  - To support larger network size, larger Larger dissections (eg. two-bytes) and larger fields, eg. $GF(2^{16})$, can be used
  - → Need more powerful ECUs
  - → Need more efficient implementation of finite field arithmetic & polynomial operation



Thanks!