Automotive Group Key Agreement and Client Authentication with DANCE

Mehmet Mueller
Dept. Computer Science, *Hamburg University of Applied Sciences*, Germany
mehmet.mueller@haw-hamburg.de

***Abstract*—Cars will be able to interact with their environment via V2X to improve driver experience and safety. Future cars are expected to operate a Service-Oriented Architecture (SOA) to become more adaptable and manageable. SOME/IP by AUTOSAR is an automotive middleware for SOA architectures and Automotive Ethernet that lacks security mechanisms. DNSSEC with DANE is a mature Internet standard that ensures the integrity and authenticity for service parameters and certificates bound to names. Traffic encryption in automotive networks require group key agreement protocols to prevent eavesdropping of personal data. In this paper, we augment the SOME/IP SD with client authenticity and a group key agreement protocol based on DNSSEC and DANE. Further, we evaluate a distributed and contributory group key agreement protocol, and implement a distributed approach into the SOME/IP SD. Next, we evaluate our prototype implementation together with service and client authentication, and a group key agreement protocol in the emulation framework Mininet with common group sizes in an automotive network of a production vehicle. We find practical performance results, especially when encryption key updates are considered while the car is not operating (e.g., parking, idling, reconfiguration).**

***Index Terms*—Automotive security, authentication, attestation, service orientation, SOME/IP, AUTOSAR, standards, encryption**

I. Introduction

Future in-vehicle networks will turn cars into feature-rich mobile transportation systems that communicate with their surroundings (Vehicle-to-X (V2X)) to benefit from internet connectivity, safe driving, efficient traffic management and driver comfort. This also provides adversaries a larger attack surface via communication interfaces [1]–[5] and in-car software [6], [7]. However, automotive protocols still lack security mechanisms [8], as they are designed for closed environments. With an active approach, industry standards like ISO/SAE 21434 [9] and legal mandates such as the European Cyber Resilience Act demand automotive security across the entire supply chain, encompassing both hardware and software.

In our previous work [10], we discussed the emergence of the Service-Oriented Architecture (SOA) paradigm and *Scalable service-Oriented MiddlewarE over IP* (SOME/IP) [11] as a widely deployed automotive middleware via IP and Automotive Ethernet [12] with real-time capabilities [13]. Here too, SOME/IP lacks security mechanisms [14], [15].

Electronic Control Units (ECUs) in automotive networks communicate collaboratively in groups on separate channels, which is done in Ethernet networks via multicast. Among security goals for group communication, there are confidentiality and authenticity [16], which we focus on in this work. Confidentiality means to protect traffic against eavesdropping, while authenticity ensures that a communication party is not an imitator who could pose a threat to vehicle safety.

Traffic encryption covers the confidentiality in networks, e.g., to prevent unauthorized tracking of vehicle location data. Challenges in group communication are the distribution of encryption keys and key updates to ensure that as little traffic as possible is exposed if an attacker breaks the encryption. Group encryption keys are distributed by Group Key Agreement (GKA) protocols, for which the three different approaches called *centralized*, *distributed* and *contributory* exist [17]. The difference between them is the instance that is selected as the sponsor for the group key. While the *centralized* approach uses a dedicated host as the sponsor, the other two involve group members and prevent a Single Point of Failure (SPOF). Prevalent approaches in automotive networks are *centralized* protocols [18]–[20] and *distributed* protocols [15], [21].

Perfect Forward Secrecy (PFS) uses encryption keys that are generated and exchanged independently of long-term keys to ensure data protection, even if long-term keys are compromised. It is even mandatory in the prominent Internet standard Transport Layer Security (TLS) 1.3 and is therefore preferred.

In our previous work [10], we covered service authenticity in SOME/IP with Domain Name System Security Extensions (DNSSEC) [22] and DNS-Based Authentication of Named Entities (DANE) [23]. Active Internet-Drafts describe client authenticity with DNSSEC and DANE called DANE Authentication for Network Clients Everywhere (DANCE) [24]–[26].

In this paper, we use DANCE for client authenticity and a *distributed* GKA protocol for confidentiality. We model a namespace for client certificates to retrieve them during the SOME/IP Service Discovery (SD) and verify remote clients. Our GKA protocol is performed during the SOME/IP SD and uses the Diffie-Hellman (DH) [27] key agreement for PFS. We compare the performance of GKA approaches separately and evaluate different security schemes in *vsomeip* [50].

The remainder of this paper is structured as follows. Section II recaps related work on SOME/IP authenticity and GKA protocols. Section III presents our DANCE-based client authenticity scheme, which expands our previous solution, and *distributed* GKA protocol in the SOME/IP SD. We evaluate our concept in Section IV and discuss performance results. Section V concludes with an outlook.

II. Background

Current vehicles include a broad spectrum of heterogeneous services [10], [28]. Miller and Valasek demonstrated successful network attacks on current vehicles via various interfaces including V2X communication [2], [3].

Automotive networks in Electrical/Electronic (E/E) architectures are becoming increasingly complex with the growth of services for which SOAs offer a more convenient handling [29]. SOME/IP by AUTOSAR is a widely deployed SOA automotive middleware, which lacks security mechanisms [14], [15].

We have evaluated our previous prototype implementation in the SOA solution SOME/IP with promising performance results, which achieves authenticity of services by using DNSSEC and DANE [10]. However, SOME/IP does not offer payload encryption between applications. Publishers transmit their application data via multicast, which requires a GKA protocol to achieve confidentiality.

In this work, we focus on expanding our implementation to client authenticity with DNSSEC and DANE and integrating a GKA protocol into the SOME/IP SD.

A. Authentication in SOME/IP and DANCE

Present authentication solutions for SOME/IP base on the public Certification Authority (CA) model with pre-deployed certificates [15], [21], on symmetric long-term keys [30] and on identity-key pairs involving symmetric long-term keys [19]. In our previous work [10], we used DNSSEC with DANE for secure service discovery. We designed a Domain Name System (DNS) namespace and bound service information and certificates to DNS names, which comply with the SOME/IP SD query possibilities, but we have omitted client authenticity for future work.

DANCE is an active Internet-Draft by the IETF and describes client authenticity for DNSSEC with DANE infrastructures. It prescribes that the client transmits its identity included in its certificate or in its *Hello* message during a TLS handshake, to allow servers to query the client's certificate from the DNS. Moreover, the identity should not have any relation to its network layer address for more flexibility, since clients may move around the network, which is likely in SOAs.

In this work, we expand our previous solution by client authenticity with DANCE. We design an additional namespace for clients to bind their certificates to their identity in the DNS. Thus, we get a solution based on a well-established and widely tested standard for robust and reliable security.

B. Group Key Agreement Schemes

Group key management schemes fall generally into three categories: (1) *centralized*, (2) *distributed* and (3) *contributory* [17]. (1) *Centralized* group key management uses a dedicated host to generate a group key and distribute it via a pair-wise secure channel to each group member (cf. Figure 1). This leads to less load on group members, but also means a SPOF possibly disabling encrypted communication entirely. (2) *Distributed* group key management chooses one group member to generate a group key and distribute it via a pair-wise secure channel to other group members (cf. Figure 1). This leads to more load on the specific group member but is more robust against host failures. (3) *Contributory* group key management requires every group member to contribute an equal share to achieve the same group key (cf. Figure 2). This allows for some key agreement protocols to be done without a secure channel but requires additional communication rounds to ensure every group member achieved the same group key [31], [32].

A preferable property for encryption keys is PFS which is mandatory in the TLS 1.3 [33] and Datagram Transport Layer Security (DTLS) 1.3 [34] Internet standards. DTLS is technically able to perform group communication via multicast as it is based on UDP, but its handshake is not designed for handling group keys according to any standard. Moreover, it is recommended to update session keys, also called re-keying, after a certain time period or after a certain amount of data is transmitted [35], [36]. In group communication, this requires to update the group session key using certain data structures every time a participant joins, leaves or the group changes by other events [17], [37]–[43]. However, updating the keys can be computationally intensive and depends on how long the data actually needs to be protected. Re-keying in vehicular networks is more difficult since their ECUs have tight hardware constraints. It is advisable to perform key updates when the car is parking, idling or being reconfigured (e.g., firmware/Over-the-Air (OTA) update) [19], [44], [45].

Present GKA solutions for SOME/IP base on *centralized* [15], [19], [30] or *distributed* [15], [21] approaches. Their solutions introduce additional communication with group key distribution servers or custom protocols performed in addition to the SOME/IP SD. Our goal is to integrate a GKA protocol into the SOME/IP SD without additional messages. We do not consider a *centralized* approach, because having a dedicated group key distribution server means having a SPOF and additional communication with it anyway. A *contributory* approach also is not a suitable candidate, since it requires additional communication rounds with every group member, but we could not find related work in the automotive context, which is why we evaluate a *contributory* competitive approach.

In this work, we implement a *contributory* competitive approach inspired by [46], [47], in which the authors used the Spread Toolkit [48] for group communication based on TCP. Our approach is based on UDP via multicast, since UDP is preferred in automotive networks and is better suited for low latency. Our competitive approach just includes join operations without data structures for handling key updates by leave, merge or other group changing events, since key updates are computationally too expensive when the car is in use. Thus, we compare our competitive approach with our *distributed* scheme separately without the SOME/IP SD, to analyze the cost in latency for different group sizes. Finally, we integrate our *distributed* scheme including PFS into the SOME/IP SD without additional messages and compare it with our other security mechanisms in *vsomeip*.

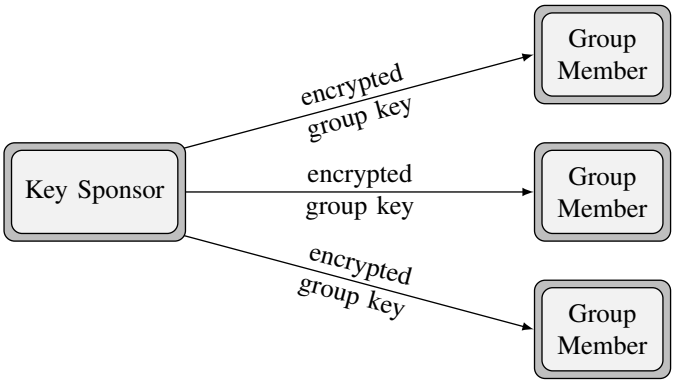

Fig. 1: Minimalistic example for *centralized* group key distribution concept with three group members. In a *distributed* approach, the key sponsor is also a group member.

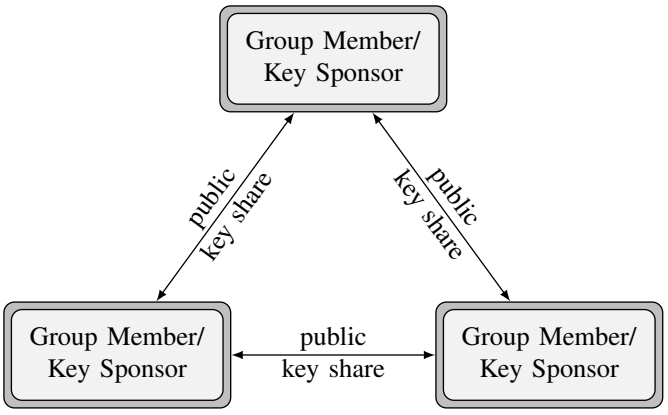

Fig. 2: Minimalistic example for *contributory* group key distribution concept with three group members.

III. Concept

In our previous work, we supplemented SOME/IP by the established Internet standards DNSSEC and DANE to achieve secure service discovery and service authenticity [10]. For our prototype implementation, we adapted the *vsomeip* reference implementation. In this work, we want to ensure that services and clients authenticate each other and establish symmetric session keys for group communication. We use DANCE to achieve additional client authenticity. For this, we bind DANE certificates to client IDs to integrate an authenticity scheme during the SOME/IP SD. For secure group communication, we integrate a *distributed* GKA protocol and perform a group key distribution mechanism during the SOME/IP SD, to encrypt multicast session communication based on the DH key exchange. Therefor, we additionally compare the suitability of our *distributed* approach with our *contributory* competitive approach for automotive networks.

A. Secure Communication Between Authenticated Services

A major difficulty is to prove the robustness and reliability of security solutions. We argue that mature and widely distributed and tested Internet standards provide a significant confidence in robustness and reliability, which is why we use and base on them for our automotive security concepts in terms of authentication and message encryption. The prominent Internet standards TLS 1.3 and DTLS 1.3 drive Internet security with authentication, based on the public CA model, and network traffic encryption.

Other than the public CA model, where certificates can be attested by multiple CAs and potentially cause attestation collisions, DNSSEC with DANE does not allow this by design [23]. We exploit DNSSEC with DANE for certificate and endpoint information management, whereby the authenticity and integrity of the certificates and endpoints are implicitly ensured.

In our previous work, we stored the publisher certificates in TLSA records and endpoint information as well as service parameters in SVCB records for service authentication in the DNS. We bound the records to query names based on the SOME/IP `find` parameters. Since endpoint information alone does not provide authenticity because of imitator attacks such as IP spoofing, a challenge-response scheme is required to verify remote endpoints. During the SOME/IP SD we queried endpoint information to validate it against the endpoint options in `offer` messages and integrated a challenge-response scheme to verify remote endpoints.

In this work, we use DANCE for client authentication. We store client certificates in TLSA records and bind them to their respective ID. Publishers then query client certificates from the DNS to verify a remote client during the SOME/IP SD. Unlike with publishers, we do not store network addresses in the DNS since it is common that clients often have dynamic or unpredictable addresses and may move around the network [25].

In addition to authentication, the encryption of network traffic ensures confidentiality to protect sensitive data from malicious attackers. Two security requirements in TLS 1.3 and DTLS 1.3 are the use of PFS and Authenticated Encryption with Associated Data (AEAD). PFS ensures data protection even if long-term keys are compromised, while AEAD guarantees the confidentiality and integrity of the data and ensures non-replayability.

As TLS 1.3 and DTLS 1.3 are not suitable for encrypting group communication, we discuss and evaluate GKA approaches. We identify a *distributed* GKA approach as suitable for a seamless integration into the SOME/IP SD and implement as well as evaluate a GKA protocol. In the GKA protocol we use the DH key agreement for PFS. The actual encrypted SOME/IP session is outside the scope of this work, in which AEAD encryption protocols would play a role.

B. Management of Group Keying

Familiar solutions for in-vehicle networks are *centralized* [18]–[20] and *distributed* approaches [15], [21]. Even having less load with the *centralized* approach, we think having a SPOF outweighs it, as long as the session establishment is within an acceptable time frame with *distributed* approaches. Therefore, we focus on examining the *distributed*

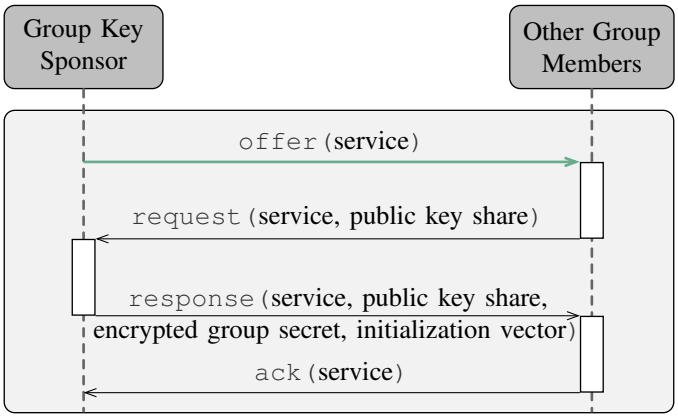

Fig. 3: Distributed Diffie-Hellman group key agreement.

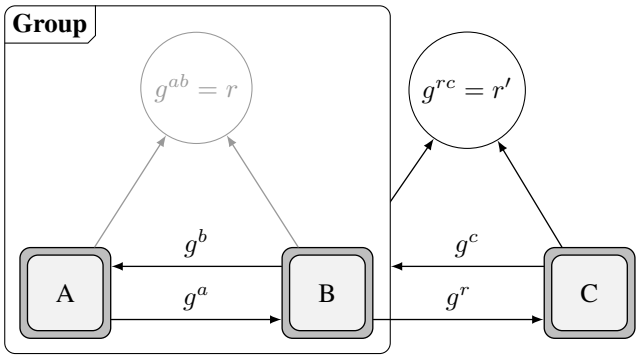

Fig. 4: Minimalistic example for *contributory* DH group key distribution concept with three group members.

and *contributory* approach. More precisely, we implement a *distributed* standalone approach inspired by [15], and a *contributory* standalone approach by [46]. In contrast to the authors of the *contributory* approach, we implement it like the *distributed* approach for the unreliable UDP protocol, which requires additional communication to ensure that group members achieve the same group key. We decide to follow these works, since both comply with the well-established and widely tested Internet standards TLS 1.3 and DTLS 1.3 in terms of key agreement and PFS, which makes these works convincing. We integrate a *distributed* approach into the SOME/IP SD in *vsomeip*, since *contributory* approaches are not seamlessly integrable due to additional group synchronization messages.

Figure 3 shows our standalone *distributed* GKA protocol. The group key sponsor offers a service whereupon other group members join with sending a `request` message that specifies the desired service and their respective public key share. Due to the unreliability of UDP, the sponsor must send the `offer` cyclically, but for our evaluation we set the receive buffer size large enough so that no cyclical messages are required. In addition, each participant knows the group size in order to determine that each participant can take part in the encrypted communication or that the group key agreement has been concluded to determine the end of the evaluation. Then, the group key sponsor performs a DH key agreement involving its own DH secret together with the public key share of the group member, whereupon the resulting symmetric key is used to encrypt the group key. After that, the group key sponsor sends a `response` message that specifies the service, its own public key share, the encrypted group secret and the initialization vector involved in the encryption. Subsequently, the group member performs a DH key agreement involving its own DH secret together with the public key share of the sponsor to decrypt the group secret with the resulting symmetric key. Finally, the group key agreement for that particular service is confirmed by the group member with an `ack` message. In *vsomeip* we choose the publisher as the group key sponsor which transmits an encrypted group key during the SOME/IP SD.

Figure 4 shows a minimalistic *contributory* DH GKA example on which the standalone implementation we evaluate is based. More precisely, we use Elliptic Curve Cryptography (ECC) for our DH GKA, to which the presented approach is applied analogously. We omitted the modulo operation in Figure 4 for the sake of clarity. The *contributory* approach chooses an initial group key sponsor, which is initially *A*, whereby the last host to join becomes the group key sponsor. For the group size of two with *A* and *B*, the *contributory* approach does not differ from a regular two party DH key exchange, as the resulting shared secret r becomes the shared group secret. Group member *B* is now the group key sponsor. If another group member *C* joins a group with a size of two or more, the group key sponsor computes the public group share g^r, using the shared group secret r from the previous agreement and sends it to the joining party *C*. *C* sends its public share g^c to all group members, whereupon all group members *A* and *B* and the joining party *C* are able to compute the new shared group secret r'. Finally, group member *C* is appointed as the group key sponsor.

For our standalone *contributory* implementation, certain messages must be sent cyclically to maintain the protocol in the event of message loss, due to the unreliability of UDP. However, for our evaluation we set the receive buffer size large enough so that no cyclical messages are required. In addition, each participant knows the group size in order to determine that each participant can take part in the encrypted communication or that the group key agreement has been concluded to determine the end of the evaluation. For that, we implement a *sponsor election* and *successor synching* phase. In addition, all messages are sent via multicast to keep all other group members up to date about new members.

Figure 5 shows the *sponsor election* of our standalone implementation. The *sponsor selection* phase ensures that each participant becomes a sponsor once and agrees on the symmetric group key, which contains its own secret and all public key shares of the previous sponsors. We assume the service provider as the initial group key sponsor. This phase is divided into the three further phases *predecessors request*, *offer* and *response*.

In the *predecessors request* phase, the current sponsor sends a `request` message that specifies the service in question and the member IDs below its own ID. This identifies out of order `request` messages in the *offer* phase to prevent group members from being appointed as the group key sponsor more than once. The first sponsor is an exception, as it has the lowest possible member ID and therefore no predecessors. Group members respond with a *response* message that specifies the desired service, the public key share and the respective member ID. Both, the sponsor and unassigned members update invalid member entries, which still specify falsely unassigned members. Other group members include the public key share of members with a higher ID in their group key agreement, if they missed `response` messages in the *response* phase, which is finally ensured in the *successor synching* phase. Unassigned members add entries of assigned members, to exclude them as a next sponsor.

The current sponsor then enters the *offer* phase if it has no member entry which specifies an unassigned member, otherwise it skips the *offer* phase entirely. The current sponsor sends an `offer` message that specifies the relevant service, to which unassigned members respond with a `request` message that specifies the desired service and the respective public key share. Both, the current sponsor and unassigned members update their entries about unassigned members by receiving `request` messages.

Finally, the current sponsor enters the *response* phase to determine the future sponsor. It agrees on the future symmetric group key by including the current secret group key and the public key share of the unassigned member it is about to appoint as the future sponsor. Then it sends a `response` message that specifies the relevant service, the current public group share and the ID, the IP and the public key share of the future sponsor. Like the current sponsor the group members agree on the future symmetric group key by including the current secret group key and the public key share of the future sponsor. The future sponsor then agrees on the future symmetric group key by including its own secret and the current public group share. All the three phases are repeated until no unassigned member is left. However, the *sponsor election* phase does not guarantee that all group members achieve the same symmetric group key for which the *successor synching* is used.

Figure 6 shows the token-based *successor synching* phase of our standalone implementation. This phase ensures that every group member includes the public key shares of the following sponsors, after the *sponsor election* phase concludes. This phase is divided into the three further phases *successors request*, *synch token* and *finish*. The initial token owner is the member with the highest member ID and as it has no successor it sends it directly to the member with the lowest member ID or service provider.

The current token owner then enters the *successors request* phase and sends a `request` message that specifies the relevant service and the member IDs whose public key shares it is missing. Concerned group members then send a `response`

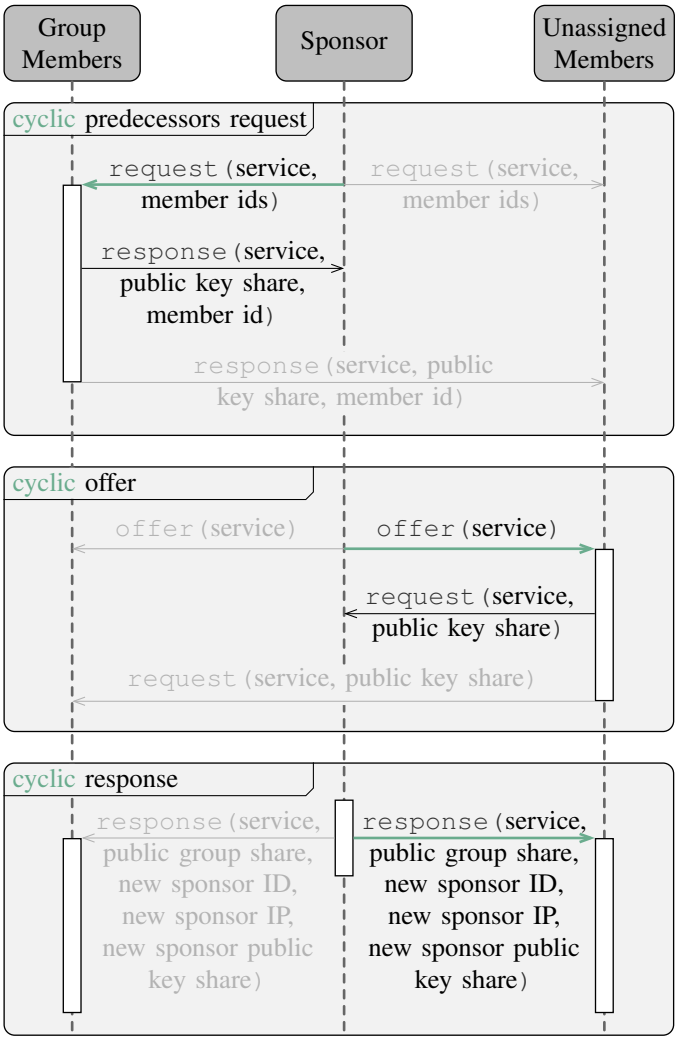

Fig. 5: Contributory Diffie-Hellman group key agreement in *sponsor election* phase.

message that specifies the relevant service, the respective member ID and the respective public key share. After the token owner receives the subsequent public key shares it includes them to the group key agreement to achieve the complete symmetric group key. As the `response` message is sent via multicast, other group members may receive missing public key shares. Subsequently, the token owner enters the *synch token* phase and sends a `synch token` message to appoint the next member as the token owner. This scheme keeps alternating between the *successors request* and *synch token* phase, until the member with the third to last ID receives the token, since the last two joined group members already agreed on the complete group key. The member with the third to last ID then enters the *finish* phase to signalize to all the other members that the group key agreement is finished. Finally, the member with the lowest member ID acknowledges the finish message.

Our goal is to integrate a group key distribution protocol complying with the SOME/IP SD, which is not possible

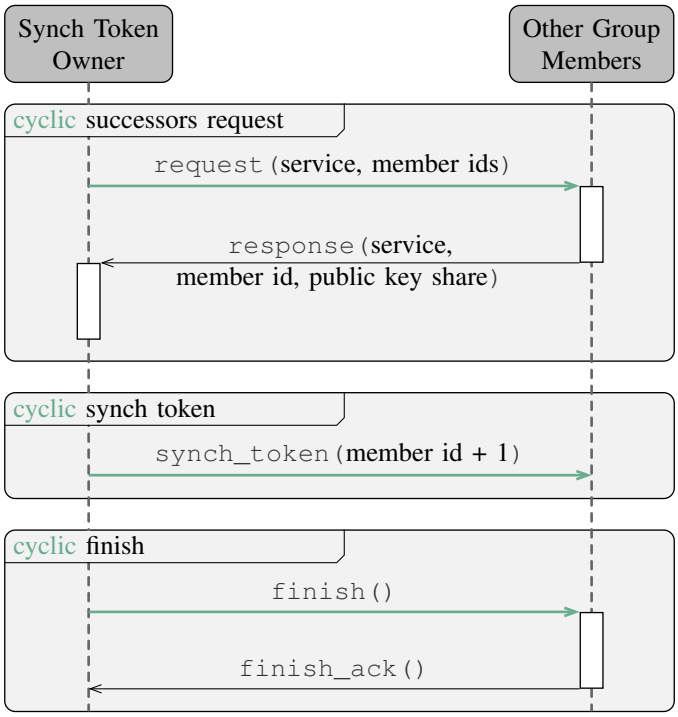

Fig. 6: Contributory Diffie-Hellman group key agreement in *successor synching* phase.

without additional communication rounds with the *contributory* approach. Therefore, we integrate a *distributed* approach during the SOME/IP SD as it can be done seamlessly. Nevertheless, we consider the *contributory* standalone approach as a competitive approach and analyze its total latency for different group sizes and compare it with the *distributed* standalone approach with regard to suitability for vehicle networks.

C. Integration of DANCE and GKA in SOME/IP

DNSSEC verifies DANE TLSA records or certificate authenticity. However, this does not verify an endpoint and still allows impersonation attacks like IP spoofing. We counter this issue by using a challenge-response scheme to verify endpoint authenticity. Figure 7 shows the conceptual architecture of *vsomeip* including our added DNSSEC components. Both the client and server share a unified stack consisting of application, routing manager, and service discovery components. The routing manager deals with application specific transport endpoints used for payload transfer and receipt.

Our modifications, also in Figure 7, are the addition of a DNSSEC resolver from which clients and servers can resolve authenticated certificates and endpoint information to verify clients and servers during the SOME/IP SD. More specifically, we add the client authenticity according to DANCE in this work, the service authenticity is done in our previous work [10]. We also add our *distributed* GKA protocol, where the server and client use their respective public key share *PKS* to establish a symmetric key *SK* according to the DH key agreement during the subscription. The server uses the

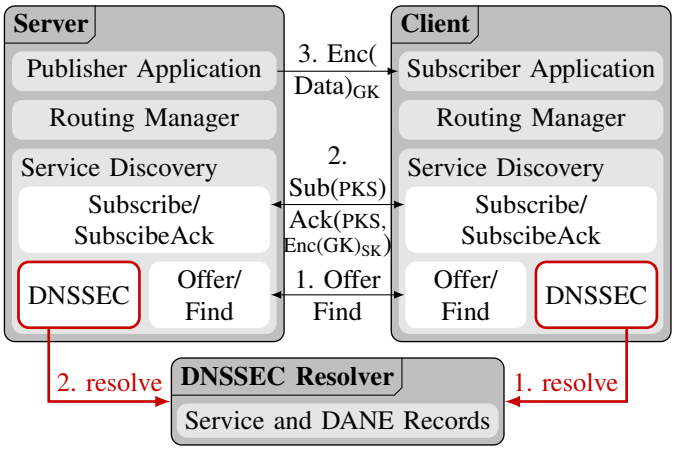

Fig. 7: SOME/IP SD modification for using DNSSEC and GKA.

SK to transmit the encrypted group key *GK*, whereupon it is used as a symmetric key for session data. Furthermore, we disabled `offer` and `find` messages in our previous work, since their authenticity is not verified by default. However, this also removes the service awareness capability which the SOME/IP SD can only fulfill through the aforementioned messages. Therefore, we now use the intended SOME/IP SD to retain its service awareness capability, but still validate it against the DNSSEC and DANE information.

We design a namespace for clients according to the similar scheme as for our service authentication (cf. [10]) following the service specific client identity in DANCE [25]. We use *client* as the parent domain and exploit the 16 bit application ID in *vsomeip* as the client ID, which we prepend to the parent domain. Again, we make use of the attribute leaf name pattern and prepend *_someip* to the client ID. A client certificate for the client ID 0x0001 has this concrete query name:

_someip.id0x0001.client.

Figure 8 shows our service and client authenticity scheme with our *distributed* GKA protocol split into a *consumer-triggered discovery* and *publish-subscribe* phase. The publisher includes a nonce challenge as a SOME/IP configuration option in the `offer` message in addition to the usual service properties. The subscriber initially queries the SVCB (cf. [10]) record of the publisher and uses it to validate the information in the `offer` message after their receipt. Subsequently, the subscriber signs the publisher's nonce challenge from the `offer` message and sends the signature along with its client id, nonce challenge and DH public share. The publisher constructs the DNS name using the client ID from the `subscribe` message and queries the subscriber's TLSA record. Then, it verifies the subscriber's signature and signs its nonce challenge. Further, it computes the shared secret using the subscriber's DH public share and encrypts the shared group secret with it. So, it packs the signature, the initialization vector used during the encryption, its own DH public share and the encrypted group secret as configuration options into the

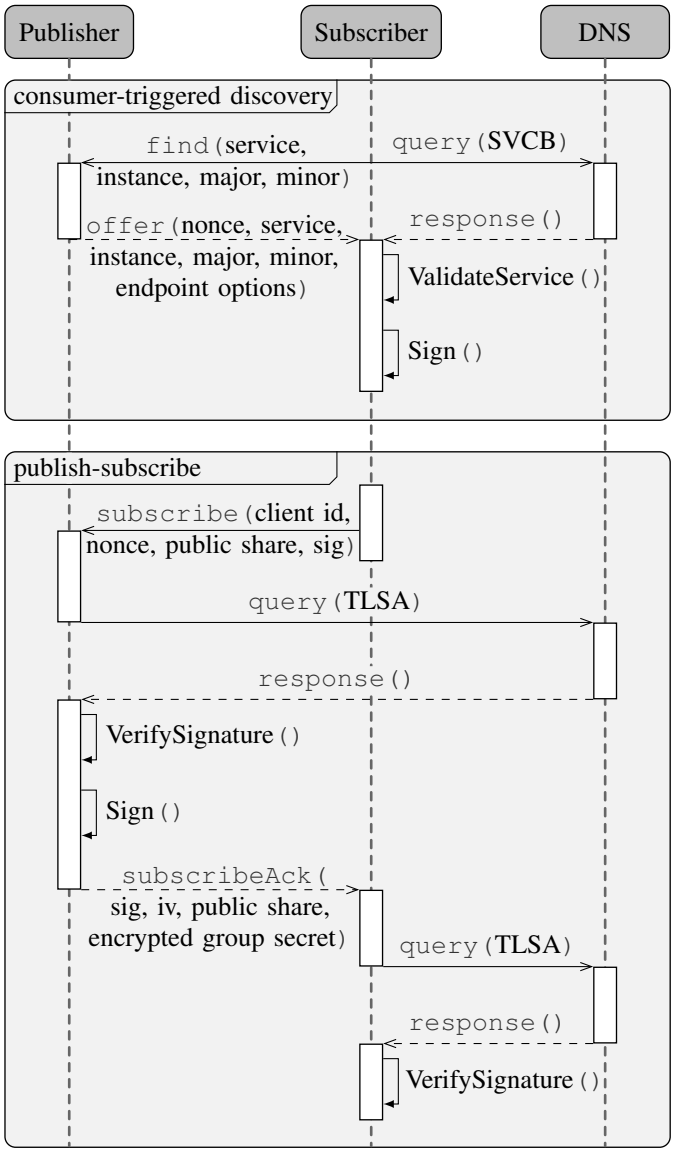

Fig. 8: Augmented SOME/IP SD with DNSSEC and DANE for secure publisher and subscriber service discovery and authentication.

`subscribeAck` message. After the subscriber receives the `subscribeAck` message, it queries the publisher's TLSA record containing the certificate of the publisher from the DNS to verify the received signature. Actually, the subscriber queries the TLSA record earlier, immediately after receiving the SVCB record, but we left it in the figure like that for the sake of clarity. It computes then the shared secret using the publisher's DH public share. After that, the subscriber decrypts the shared group secret with the computed shared secret and initialization vector.

IV. Evaluation

We evaluate a *distributed* approach inspired by Zelle et al. [15] and a *contributory* approach by Kim et al. [46]. We consider only the join operation for group members, since leave or other group changing operations are not practicable for in-car networks [19], [44], [45]. First, we compare our both protocols in a separate implementation. The *contributory* protocol can not be integrated into the SOME/IP SD handshake without additional communication rounds and overhead which is why we do not evaluate it in *vsomeip*. In contrast, the *distributed* protocol is suitable for integration into the SOME/IP SD which we implement into *vsomeip* and evaluate. We compare our prototype implementation in terms of discovery and total latency with different security mechanisms.

A. Evaluation Setup

We run our evaluations on the same host system (CPU: i7-1260P, RAM: 32GB LPDDR5-6400) with cryptography libraries (*Crypto++* [49]: 8.9, *Crypto++ PEM Pack*: 8.2). The first evaluation compares the *distributed* and *contributory* protocol and runs all group members in separate processes on the loopback interface with larger buffers to prevent packet loss. Here we measure the latency for the key agreement that results when all group members receive the group key. The second evaluation compares our prototype implementation which customizes *vsomeip* with a DNS resolver library, and emulates all group members and the authoritative DNS name server with their own network interface with 1 Gbit bandwidth and no transmission delay on a Linux Kernel-based Virtual Machine (KVM) (*Mininet*: 2.3.0, *Ubuntu*: 22.04.3 LTS, *NSD*: 4.8.0, *vsomeip* [50]: 3.4.10, *C-ARES* [51]: 1.81.1). Here, we measure the service discovery and total latency. The total latency includes the service discovery and subscription and thus the costs for all cryptographic operations. We evaluate four solutions from our previous work (cf. [10]) together with four additional solutions implemented in *vsomeip* [50]:

1) **PUB AUTH, SUB AUTH**: Performs publisher and subscriber authentication with the original SOME/IP SD without any DNS operations. It uses pre-deployed certificates and our challenge-response mechanism.
2) **PUB AUTH, SUB AUTH, GKA**: Performs our *distributed* GKA in addition to the previously mentioned solution.
3) **PUB AUTH, SUB AUTH, DNSSEC, DANE**: Performs the same operations as the first solution, but instead of pre-deployed certificates, certificates are queried using DNSSEC and DANE.
4) **PUB AUTH, SUB AUTH, DNSSEC, DANE, GKA**: Performs our *distributed* GKA in addition to the previously mentioned solution and thus contains all security mechanisms.

First, we compare all solutions with one publisher and one subscriber. We then compare solution number four and three, with a publisher and a selected number of subscribers. Moreover, SOME/IP uses a random initial delay to scatter `find` and `offer` messages to prevent hosts from flooding the network all at once. We turn off the random initial delay for the first comparison in order to have comparability with solutions not using the SOME/IP SD.

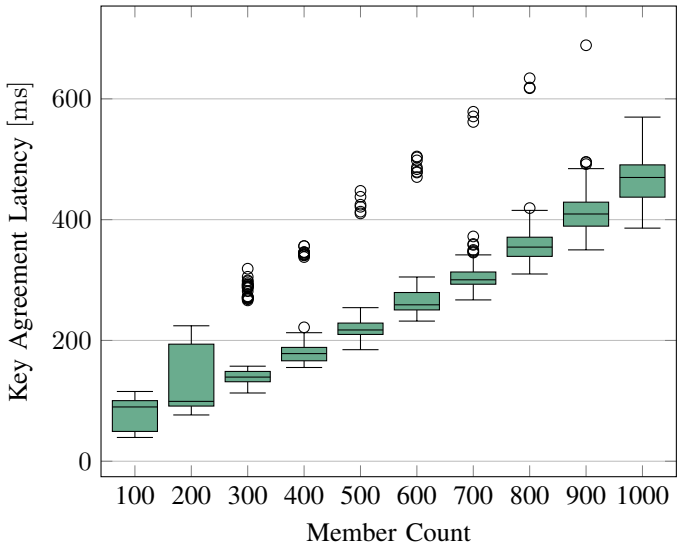

Fig. 9: Latency of *distributed* key agreement protocol.

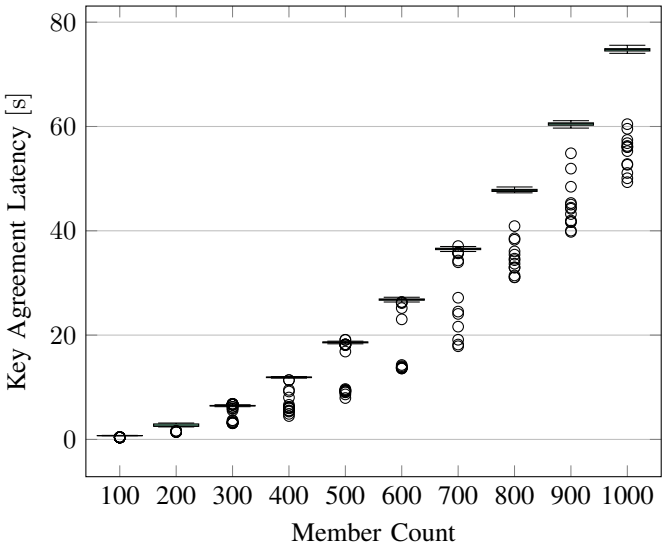

Fig. 10: Latency of *contributory* key agreement protocol.

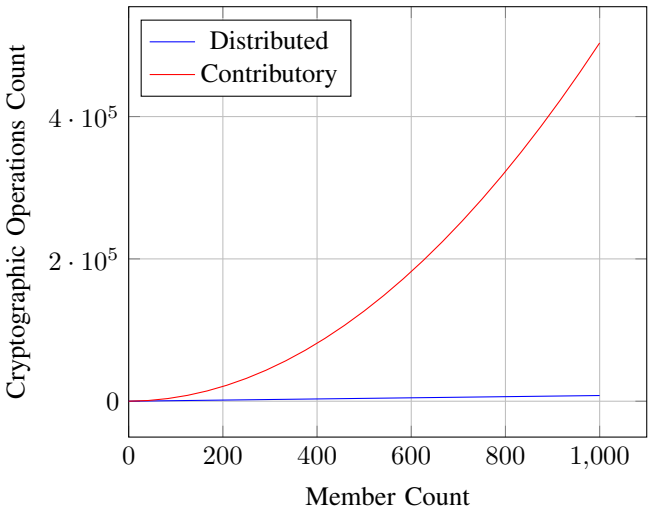

Fig. 11: *Distributed* vs. *contributory* cryptographic operations count.

B. Distributed vs. Contributory Key Agreement

Figure 9 and 10 show the key agreement latency for our *distributed* and *contributory* key agreement implementations for group sizes from 100 to 1000 in hundreds of steps. For each group member count we collect 133 samples with timestamps indicating the start of the agreement and the end. It is noticeable that the *distributed* protocol remains below 600 ms with 1000 group members while the *contributory* protocol performs on a scale of seconds. We measure that the *contributory* protocol already hits a latency above 2 s with 200 group members. This is due to involving every group member during a join operation while the *distributed* protocol only involves the group key sponsor and the joining group member, resulting in a different number of cryptographic operations.

This becomes clearer in Figure 11, which shows the number of cryptographic operations for group sizes from 100 to 1000 in hundreds of steps. We evaluate the number of cryptographic operations including the generation of the DH key pair, agreement on the new shared group secret and the public group share computation for the *contributory* protocol. For the *distributed* protocol, we include the generation of the DH key pair and group secret, agreement on the individual shared secret, hash calculation over the shared secret, and the encryption and decryption of the group secret. The *contributory* protocol scales not as good as the *distributed* one but seems still applicable for smaller group sizes like 26 for which we measure latencies between 90 ms and 109 ms. In comparison, we measure latencies between 10 ms and 35 ms for the *distributed* protocol.

Considering only the start-up of a car after which as many services as possible should be available, the *distributed* protocol is more suitable. The *contributory* approach becomes more computationally intensive as the group size increases, which requires consideration of factors such as group size and the number of group changes.

C. Comparison of All Solutions

Figure 12 and 13 show the discovery and total latency of all our solutions without the random initial delay and request-response delay in *vsomeip*. Here, we evaluate our solutions with one publisher and one subscriber. For each solution we collect fifty samples with timestamps to calculate latencies based on the difference between them.

The solutions in Figure 12 have not significant differences in their latency. The solutions without the SOME/IP SD seem to be faster, which could be due to the maturity of the DNS and its optimization in response time. Moreover, this is reinforced by the fact that the SOME/IP SD packets are smaller than the SVCB packets and the transmission delay in Mininet is turned off anyway, which rules out the transmission delay.

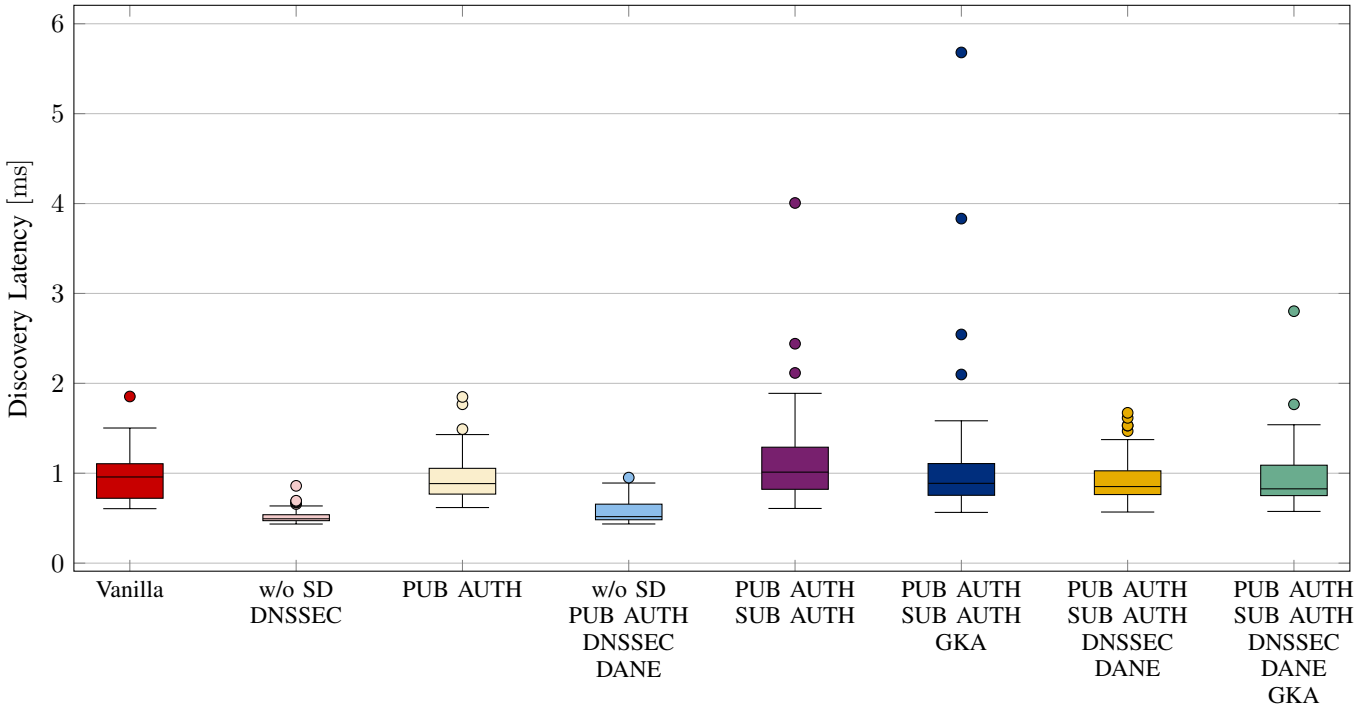

Fig. 12: Discovery latency of all solutions without random initial delay.

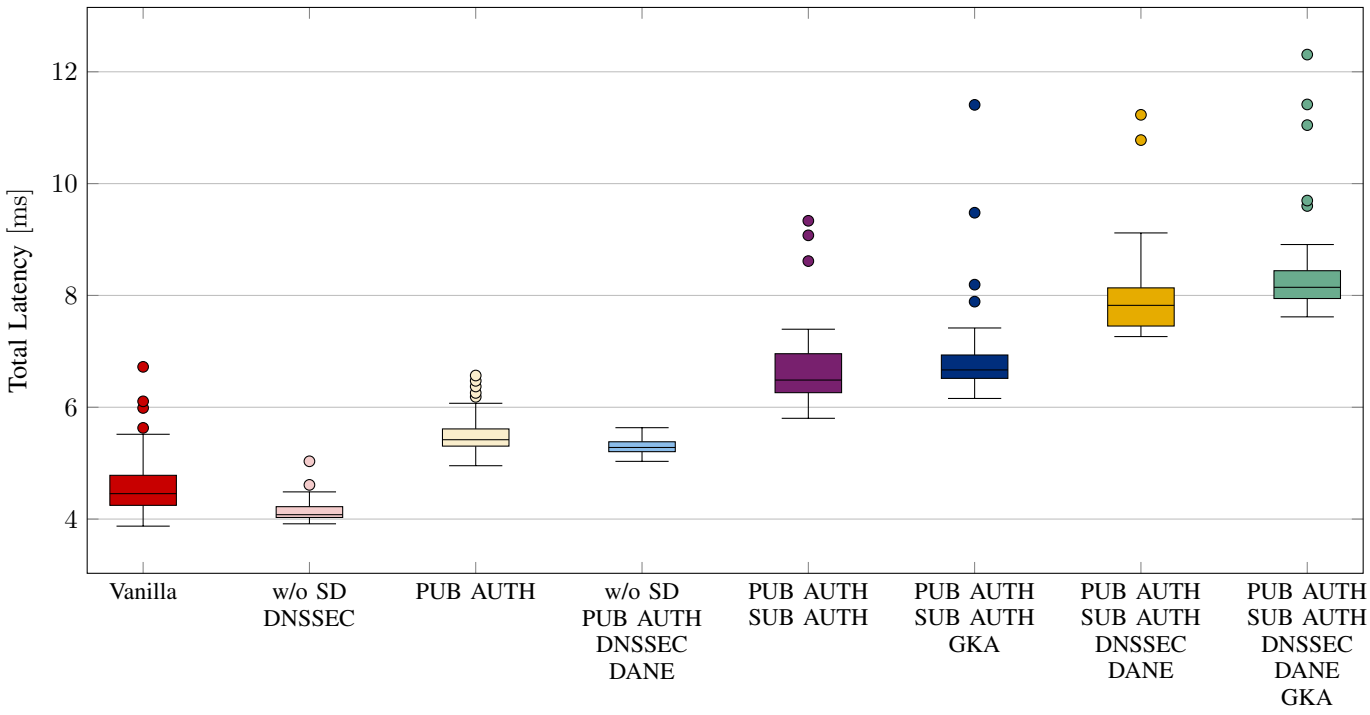

Fig. 13: Total latency of all solutions without random initial delay.

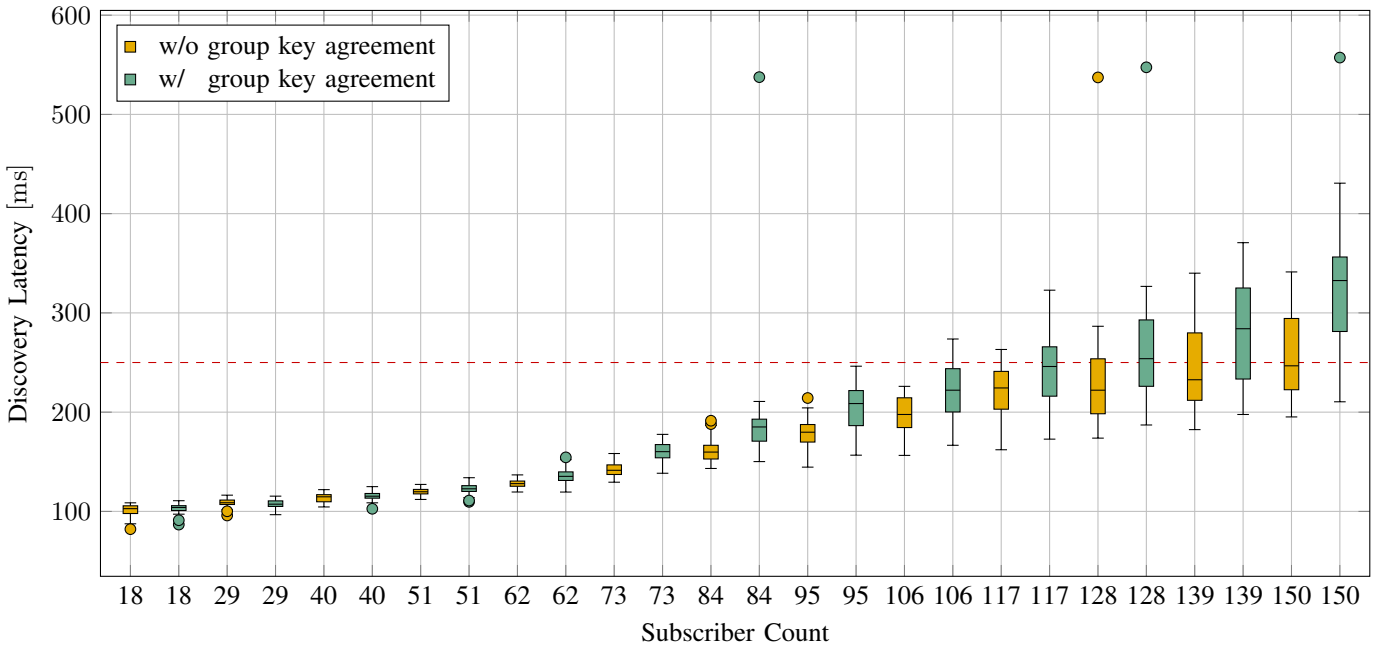

Fig. 14: Discovery latency without vs. with *distributed* group key agreement.

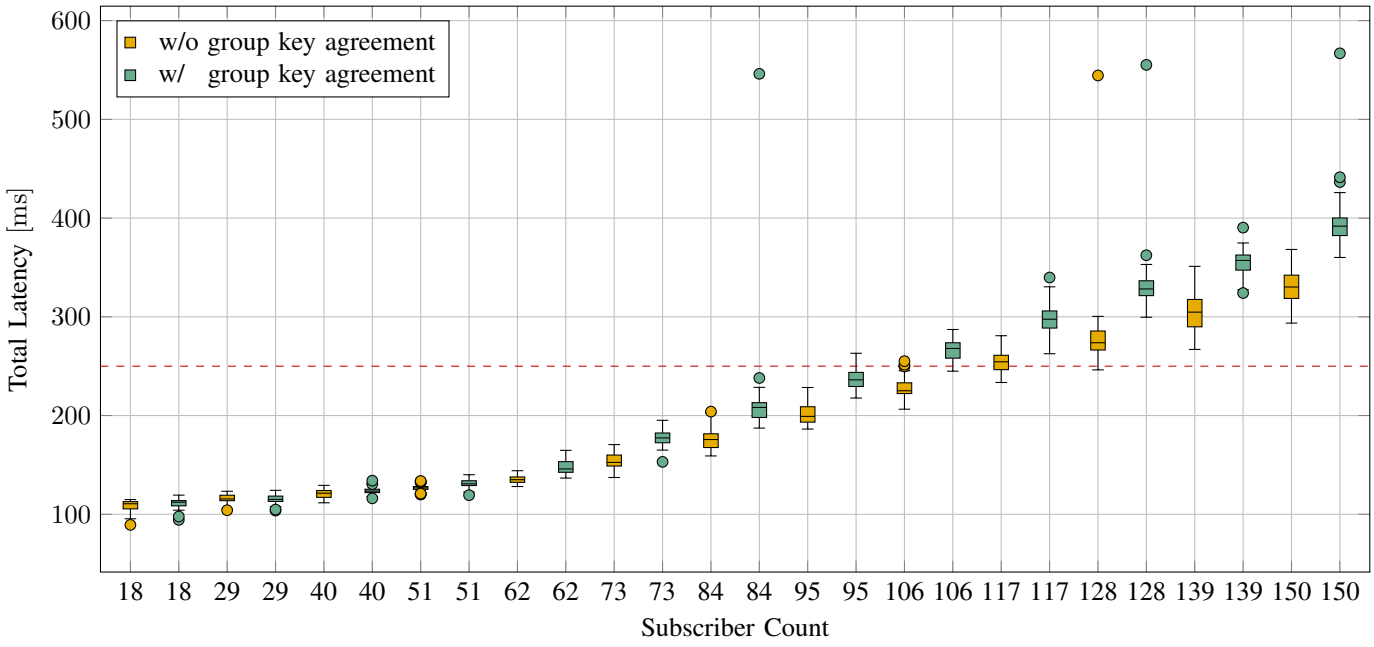

Fig. 15: Total latency without vs. with *distributed* group key agreement.

The solutions in Figure 13 show an expected increase in latency with increasing security mechanisms. The last two solutions from right with DNSSEC and DANE start to show a larger performance penalty. This is due to the fact that the publisher has to query the certificate of the subscriber after it receives its `subscribe` message including the certificate's DNS name and wait for its receipt. In addition, the certificate exceeds an MTU of $1{,}500\,\mathrm{B}$ which is why it is sent in multiple TCP segments. That is not noticeable only having publisher authenticity since the subscriber queries the publisher's certificate in parallel to the SOME/IP SD and has to wait for the `subscribeAck` message anyway.

The certificate management with DNSSEC and DANE shows only a slight performance penalty with additional client authenticity in contrast to pre-deployed certificates. We compare both client authentication solutions with and without GKA to obtain practical group sizes in terms of latency for the vehicle commissioning.

D. Client Authentication with vs. without Encryption

Figure 14 and 15 show the discovery and total latency of our solution with all security mechanisms. Both figures show the impact of our *distributed* GKA for the solution with service and client authenticity using DNSSEC with DANE. In other words, we assess our solution with all security mechanisms versus without its *distributed* GKA. Here, we evaluate our solutions with one publisher and subscribers from 1 to 150, of which we present selected group sizes from 18 to 150 in steps of eleven. Our group sizes are based on the data of a production vehicle in which a device is involved in a maximum of 146 services. For each solution we collect fifty samples with timestamps to calculate latencies based on the difference between them. Automotive engineers recommend not exceeding a delay of $250\,\mathrm{ms}$ before the car can be entered in order to ensure a satisfactory user experience. We therefore consider group sizes that exceed a latency of $250\,\mathrm{ms}$ to be rather impractical if a GKA has to be concluded for the entire group, especially if the car can only be entered afterwards.

In Figure 14, a subscriber count of 106 without GKA is still below the $250\,\mathrm{ms}$ mark, whereas a subscriber count of 95 with GKA almost reaches it during the discovery phase. In Figure 15, 106 subscribers without GKA generally remain below a latency of $250\,\mathrm{ms}$ after the subscription is finished for all of them, but there are still some outliers. A subscriber count of 95 with GKA exceeds the $250\,\mathrm{ms}$ mark in contrast to the discovery phase, whereas it looks more practical with 84 subscribers.

Looking at the total latency and the $250\,\mathrm{ms}$ latency mark, a group size of 106 is the maximum without GKA, while with GKA it is already reached with around 84 group members. Since our evaluation is done on a single host machine in an emulation framework, the performance results are likely better on actual nodes, where the load is actually distributed. In addition, the use of hardware acceleration for cryptographic operations would also contribute to an improvement.

V. Conclusion and Outlook

In this paper, we supplemented our previous service authenticity scheme by client authenticity and a Group Key Agreement (GKA) protocol required for encryption. Our work is aimed at automotive Ethernet networks considering SOAs, which enable a higher flexibility and maintainability in future cars. In addition to authenticity, we also drew attention to the need for confidentiality and message integrity in automotive networks, as sensible data such as GPS can enable attackers to track the vehicle's location.

Our work is based on established and widely used Internet standards and thus offers robust and reliable security with a high level of trust. With DANCE and the mature, and well-established Internet standards DNSSEC and DANE, we secured the widespread automotive middleware SOME/IP by AUTOSAR with additional client authenticity, which complements our previous work [10]. Further, we integrated seamlessly a GKA protocol into the SOME/IP Service Discovery (SD) without additional communication flow, avoiding additional network latency. Since there is no Internet standard supporting GKA, we discussed and evaluated the suitability of GKA approaches, considering industry requirements for the start-up time and group sizes used in a production vehicle. Here, we employed basic security mechanisms which meet current security requirements such as authenticated session key agreement and Perfect Forward Secrecy (PFS), which are mandatory in the well-established and widely tested Internet standards TLS 1.3 and DTLS 1.3. Our findings showed that our solution is practical for a group size of around 84 nodes, in which all ECUs have to complete their session establishment, before the car can be entered.

Our solution leads to the following future research directions. A performance evaluation in a full-featured production grade vehicle with hardware acceleration for cryptography will likely improve benchmark results, since the load then is distributed on individual ECUs. Since key updates in automotive networks are done in certain operating phases, a risk assessment is required for storing the session keys for encryption in memory without hardware protection. In addition, an authorization mechanism that determines which services, clients may access or servers may offer, would lead to a more fine-granular access control.

References

[1] S. Checkoway *et al.*, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," in *Proceedings of the 20th USENIX Security Symposium*, vol. 4. USENIX Association, Aug. 2011, pp. 77–92.

[2] C. Miller and C. Valasek, "A Survey of Remote Automotive Attack Surfaces," *Black Hat USA*, vol. 2014, 2014.

[3] ——, "Remote Exploitation of an Unaltered Passenger Vehicle," *Black Hat USA*, vol. 2015, p. 91, 2015.

[4] Z. Cai *et al.*, "0-days & Mitigations: Roadways to Exploit and Secure Connected BMW Cars," *Black Hat USA*, vol. 2019, p. 6, 2019.

[5] S. Nie *et al.*, "Free-fall: Hacking Tesla from Wireless to Can Bus," *Briefing, Black Hat USA*, vol. 25, p. 16, 2017.

[6] F. Kohnhäuser *et al.*, "Ensuring the Safe and Secure Operation of Electronic Control Units in Road Vehicles," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 126–131.

[7] L. Xue *et al.*, "SAID: State-aware Defense Against Injection Attacks on In-vehicle Network," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 1921–1938.

[8] A. Martínez-Cruz *et al.*, "Security on in-vehicle communication protocols: Issues, challenges, and future research directions," *Computer Communications*, vol. 180, pp. 1–20, Dec. 2021.

[9] International Organization for Standardization, "Road vehicles – Cybersecurity engineering," ISO, Geneva, CH, Standard ISO/SAE DIS 21434, 2020.

[10] M. Mueller *et al.*, "Authenticated and Secure Automotive Service Discovery with DNSSEC and DANE," in *Proc. of the 14th IEEE Vehicular Networking Conference (VNC)*. IEEE, Apr. 2023.

[11] AUTomotive Open System ARchitecture (AUTOSAR) Consortium, "SOME/IP Service Discovery Protocol Specification," AUTOSAR, Tech. Rep. 802, Nov. 2022.

[12] K. Matheus and T. Königseder, *Automotive Ethernet*. Cambridge, United Kingdom: Cambridge University Press, Jan. 2015.

[13] IEEE 802.1 Working Group, "IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks," IEEE, Standard Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014), Jul. 2018.

[14] R. B. Tzvi, "Hijacking SOME/IP Protocol with Man in the Middle Attack,". Website. [Online]. Available: https://argus-sec.com/blog/cyber-security-blog/some-ip-protocol-man-in-the-middle-attack/

[15] D. Zelle *et al.*, "Analyzing and Securing SOME/IP Automotive Services with Formal and Practical Methods," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ser. ARES 2021. ACM, Aug. 2021.

[16] R. Canetti *et al.*, "Multicast security: A Taxonomy and Some Efficient Constructions," in *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*. IEEE, 1999.

[17] Y. Amir *et al.*, "On the Performance of Group Key Agreement Protocols," *ACM Transactions on Information and System Security*, vol. 7, pp. 457–488, Aug. 2004.

[18] H. Khemissa and P. Urien, "Centralized architecture for ECU security management in connected and autonomous vehicles," in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, Oct. 2022.

[19] B. Ma *et al.*, "An Authentication and Secure Communication Scheme for In-Vehicle Networks Based on SOME/IP," *Sensors*, vol. 22, p. 647, Jan. 2022.

[20] H. Tan, "An Efficient Key Management Scheme For In-Vehicle Network," 2023.

[21] M. Iorio *et al.*, "Securing SOME/IP for In-Vehicle Service Protection," *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 13 450–13 466, Nov. 2020.

[22] D. E. 3rd, "Domain Name System Security Extensions," IETF, RFC 2535, March 1999.

[23] P. Hoffman and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA," IETF, RFC 6698, August 2012.

[24] S. Huque and V. Dukhovni, "TLS Extension for DANE Client Identity," Internet Engineering Task Force, Internet-Draft draft-ietf-dance-tls-clientid-03, Jan. 2024, work in Progress.

[25] ——, "TLS Client Authentication via DANE TLSA records," Internet Engineering Task Force, Internet-Draft draft-ietf-dance-client-auth-05, Jan. 2024, work in Progress.

[26] A. Wilson *et al.*, "An Architecture for DNS-Bound Client and Sender Identities," Internet Engineering Task Force, Internet-Draft draft-ietf-dance-architecture-03, Jan. 2024, work in Progress.

[27] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. Inf. Theor.*, vol. 22, pp. 644–654, Nov. 1976.

[28] M. Cakir *et al.*, "A QoS Aware Approach to Service-Oriented Communication in Future Automotive Networks," in *2019 IEEE Vehicular Networking Conference (VNC)*. Piscataway, NJ, USA: IEEE Press, Dec. 2019.

[29] A. Kampmann *et al.*, "A Dynamic Service-Oriented Software Architecture for Highly Automated Vehicles," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Piscataway, NJ, USA: IEEE Press, 2019, pp. 2101–2108.

[30] S. Lee *et al.*, "Protecting SOME/IP Communication via Authentication Ticket," *Sensors*, vol. 23, p. 6293, Jul. 2023.

[31] K. P. Birman and T. A. Joseph, "Reliable communication in the presence of failures," *ACM Transactions on Computer Systems*, vol. 5, pp. 47–76, Jan. 1987.

[32] Y. Amir *et al.*, "The Spread Toolkit Architecture and Performance," Johns Hopkins University, CNDS-2004-1, 2004.

[33] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," IETF, RFC 8446, August 2018.

[34] E. Rescorla *et al.*, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3," IETF, RFC 9147, April 2022.

[35] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol," IETF, RFC 4253, January 2006.

[36] E. Barker *et al.*, "Guide to IPsec VPNs," National Institute of Standards and Technology, Gaithersburg, MD, US, Tech. Rep. SP 800-77r1, June 2020.

[37] A. Ballardie, "Scalable Multicast Key Distribution," IETF, RFC 1949, May 1996.

[38] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification," IETF, RFC 2093, July 1997.

[39] D. Wallner *et al.*, "Key Management for Multicast: Issues and Architectures," IETF, RFC 2627, June 1999.

[40] M. Baugher *et al.*, "Multicast Security (MSEC) Group Key Management Architecture," IETF, RFC 4046, April 2005.

[41] B. Weis *et al.*, "The Group Domain of Interpretation," IETF, RFC 6407, October 2011.

[42] S. Jarecki *et al.*, "Flexible Robust Group Key Agreement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 879–886, May 2011.

[43] M. Burmester and Y. Desmedt, *A Secure and Efficient Conference Key Distribution System (Extended Abstract)*. Springer Berlin Heidelberg, 1995, pp. 275–286.

[44] P. Mundhenk *et al.*, "Lightweight Authentication for Secure Automotive Networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, ser. DATE 2015. IEEE Conference Publications, 2015.

[45] D. Zelle *et al.*, "On Using TLS to Secure In-Vehicle Networks," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ser. ARES '17. ACM, Aug. 2017.

[46] Y. Kim *et al.*, "Group Key Agreement Efficient in Communication," *IEEE Transactions on Computers*, vol. 53, pp. 905–921, Jul. 2004.

[47] ——, "Tree-based group key agreement," *ACM Transactions on Information and System Security*, vol. 7, pp. 60–96, Feb. 2004.

[48] Spread Concepts LLC, "The Spread Toolkit,". Website. [Online]. Available: https://www.spread.org/

[49] W. Dai, "Crypto++ library 8.9,". Website. [Online]. Available: https://www.cryptopp.com/

[50] BMW AG, "vsomeip v3.4.10," GitHub repository. Available: https://github.com/COVESA/vsomeip/

[51] Contributors, "C-ARES 1.18.1,". Website. [Online]. Available: https://c-ares.org/