**Cryptography in an Automotive Zero Trust Architecture**

**Andrew Quick, Southwest Research Institute**

Introduction & Background

This paper provides an overview and research resources for cryptography in an automotive Zero Trust Architecture (ZTA). The automotive landscape consists of a network of embedded systems that have severe constraints in performance, timing, power, and cost. ZTA emphasizes authentication to ensure message traffic is only processed if the source is genuine and the message is untampered with. This paper assumes the reader has a general understanding of the goals associated with ZTA and the hardware constraints of an automotive system.

The primary cryptographic goal in ZTA is message authentication, this is ideally achieved through MACs and freshness values, such as SecOC, and symmetric encryption. To distribute the private keys associated with message authentication/encryption; public-key encryption and certificates are likely required. Additionally, public key exchange, digital signatures, and post quantum cryptography may prove useful within an automotive ZTA.

Message Authentication: Utilizing MACs and Hash Functions

A core component of ZTA is message authentication. Message authentication aims to ensure network traffic originates from an authentic source and has not been improperly modified. However, hash functions and Message Authentication Codes (MACs), utilized in SecOC, do not encrypt message data. Encryption can be added before or after a MAC but is an uncommon practice for safety critical functions due to the increased latency [1].

Diagram

Description automatically generated with medium confidenceHash functions and MAC algorithms create authentication tags, that act as digital fingerprints, unique to every message data combination. Small changes to the message data lead to significant changes to a newly calculated authentication tag. In practice, the authentication tag is sent with the message, the recipient then calculates a new authentication tag out of the message data and compares it to the sent authentication tag. Changes between the two authentication tags alert the receiver to potential tampering.

Figure 1: SecOC by AUTOSAR (graphic credit: SwRI)

However, if a malicious actor discovers the hash function in use, they can generate a fake message by calculating their own authentication tag to travel with their malicious data [1]. MACs mitigate this fraud by introducing a secret key into the authentication tag generation process [4]. As long as the key remains uncompromised, a 3rd party would be unable to calculate the correct authentication tag. Two commonly used MACs include Hash-based message authentication code (HMAC) standardized in FIPS 198-1 [2] and Cipher-based message authentication code (CMAC) standardized in NIST SP 800-38B [3]. A commonly used CMAC implementation mechanism, in the automotive industry, is the Secure Onboard Communication Protocol (SecOC) by AUTOSAR [4]. Shown in Figure 1, SecOC utilizes CMAC and a freshness value to ensure message authentication. Freshness values are commonly used along with MAC generation to prevent replay attacks.

While MACs are generally preferred over hash functions, there are three major disadvantages of MACs compared to a hash function [1].

1. Key management logistics are required.
2. The time to generate/verify the MAC increases the latency.
3. At higher data rates, the computation requirements can be overwhelming.

Message Encryption: Utilizing Ciphers

Unlike MACs, encryption provides confidentiality, but this comes with increased latency and strain on hardware. It should be emphasized ZTA can be achieved without encryption, as message authentication is the priority, not confidentiality. However, for message encryption, symmetric key cryptography is favorable over public key cryptography due to its superior speed and simplicity. Symmetric key cryptography is often broken into two groups:

1. Block ciphers: encrypts a group of bits or bytes at a time.
2. Stream ciphers: encrypts one bit or byte at a time.

Block ciphers, compared to stream ciphers, offer high diffusion and immunity to tampering; but are slower to encrypt/decrypt and an error in one symbol may corrupt the entire block [5]. High diffusion means if we change one bit of plaintext, then about half of the ciphertext changes, and vice versa. This greatly increases the computational requirements to crack the cipher, but decreases reliability, due to the error propagation of unintentional bit changes. A viable block cipher is the Advanced Encryption Standard (AES); it’s supported in most modern ECUs with additional cryptographic hardware accelerators [6].

In contrast, stream ciphers offer a higher encryption/decryption speed and lower error propagation, while being more susceptible to modifications and decreased diffusion [5]. While considered less secure than block ciphers, stream ciphers are popular in embedded applications due to their superior speed and reliability.

When possible, combing MACs for authentication and encryption for confidentially, a practice called authentication encryption, is desired. This can be achieved by manually combining a MAC with cipher or by using a single authentication encryption mode of operation, such as the Galois/Counter Mode (GCM) [1]. GCM is a popular message authentication mode of operation due to its high-speed with relatively low hardware requirements. GCM first encrypts the data, commonly through an AES-128 block cipher, then calculates a MAC out of the cyphertext. This allows the data to be authenticated before decryption, preventing malicious code from being read.

Key Distribution: Utilizing Public-Key Encryption

In the cases above, the symmetric keys for MAC generation or message encryption, will need to be distributed over an unsecure communication bus to authentic devices. In this scenario asymmetric encryption, often referred to as public-key encryption, is ideal. Unlike symmetric encryption, public-key encryption conceals all private keys permanently in hardware, offering enhanced security. However, due to the use of public and private keys, there is substantial increases in latency, compared to symmetric encryption. Public-key encryption is shown in Figure 2, the receiver transmits a public key to all potential senders. When the sender transmits a message, it uses the receiver’s public-key to encrypt the message. Then, the receiver uses their private key to decrypt the message.

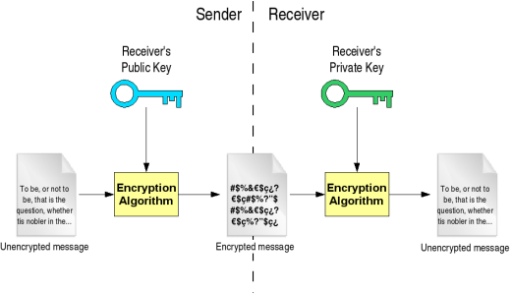


Figure 2: Public-Key (Asymmetric) Cryptography [7]

Rivest-Shamir-Adleman (RSA) is one of the oldest and most widely used public-key cryptosystems [1]. Its security guarantees rely on the factorization of large prime numbers, which is computationally expensive to implement, but infeasible to crack with conventional computing. RSA comes in varying key lengths, usually 2048 to 4096 bits, as 1024 bits and below is widely considered breakable.

Elliptic-Curve Cryptography (ECC) utilizes the discrete logarithm problem in the form of finite elliptic curve groups [8]. This provides similar security to RSA but with a dramatically smaller key-size. Key size is inversely proportionate to efficiency, meaning ECC is ideal for lower power environments such as automotive microcontrollers. The key size comparison in Table 1 compares various symmetric encryption algorithms with ECC and RSA based on equivalent levels of security. The difference in key sizes provides insight into the respective efficiency tradeoffs.

|  |  |  |
| --- | --- | --- |
| **Security level (bits)** | **ECC** | **RSA** |
| SKIPJACK (80) | 160 | 1024 |
| Triple-DES (112) | 224 | 2048 |
| AES-Small (128) | 256 | 3072 |
| AES-Medium (192) | 384 | 8192 |
| AES-Large (256) | 512 | 15360 |

*Table 1: Key Size Comparison in bits* [8]

As shown in Table 1, RSA requires 12 to 60 times the bits than symmetric keys, while ECC only doubles the key size. ECC is currently the most efficient mainstream public-key encryption schemes [8].

Key Management: Certificates

The integrity of public-key cryptography is completely reliant on the chain of trust, meaning messages must be encrypted with the public key of a trusted source. Without chain of trust, a 3rd party can manipulate a sender to use the 3rd party’s public-key, allowing the 3rd party fully decrypt messages via their private key. To create this chain of trust, a certificate-based scheme is likely required. A commonly used certificate format is the X.509 International Telecommunication Union standard, which utilizes a public-key hashing algorithm to ensure source authentication [1]. Creating this chain of trust is essential to ZTA, as failure to secure the public-key infrastructure, collapses the entire security apparatus.

Public-Key Authentication: Utilizing Digital Signatures

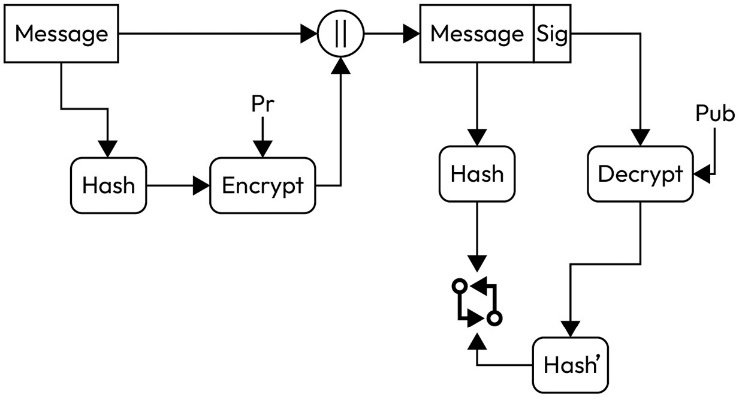
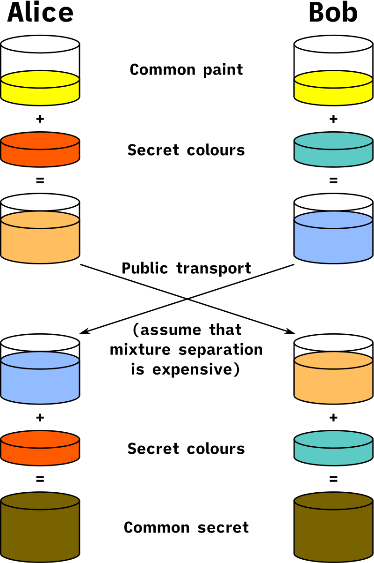
A digital signature is used to ensure integrity, authenticity, and non-repudiation (sender can’t falsely claim they didn’t sign the message). Digital signatures may provide uses in areas where a MAC does not meet security needs. Shown in Figure 3, digital signatures work like message encryption in reverse, encrypting a hash with the sender's private key to be decrypted with the corresponding public key [1]. This offers more security than MACs, as the private key does not have to travel the medium but comes with similar resource restraints as public key encryption.

Figure 3: Digital Signatures [1]

DSA (Digital Signature Algorithm) is a variant of ElGamal encryption that takes advantage of the discrete logarithm problem [9]. Standardized in FIPS 186-4, it is the most used digital signature and offers similar levels of security with similar key-sizes as RSA [10].

Elliptic Curve Digital Signature Algorithm (ECDSA) is an Elliptic-Curve variant of DSA. It offers smaller key sizes for similar levels of security as DSA making it more ideal for areas of energy/time restraints.

Public Key Exchange: Shared Symmetric Keys

Public key exchange may be useful in areas where secure distribution of a symmetric encryption key is needed. Public key exchange allows 2 or more parties to establish a shared symmetric key over an unsecure communication medium; a hacker would need to identify the private keys of each device to be able to calculate their shared symmetric key.

Diffie-Hellman key exchange (DHKE) is considered the base of most modern key exchanges [9]. The paint analogy in Figure 4 explains the DHKE well. The common paint equates to an agreed upon public key, secret colors is their private keys, public transport is the communications channel, and common secret is the finalized shared symmetric key.

Figure 4: Diffie-Hellman key exchange [14]

A common key exchange is a combination of ECC and DHKE called Elliptic Curve Diffie Hellman Key Exchange (ECDH). ECDH works just like the analogy above but uses ECC for their keys and domain parameters [10].

Most key exchanges, including ECDH, utilize ephemeral keys. Ephemeral keys are temporary keys that are destroyed when a session is terminated. Ephemeral keys are preferred over long-term key storage as stored keys are at risk to side channel attacks [1]. Ideally a vehicle would destroy all its keys when turned off and distribute new keys on startup, but this would extend vehicle startup time significantly.

Post-Quantum Cryptography (PQC)

Quantum computing can exploit quantum mechanics to solve mathematical problems that would be considered infeasible for classical computing [11]. As the scale and availability of quantum computing increases, there must be a proportional response in cryptography to develop mathematical problems infeasible to quantum computing. For most symmetric keys, it is a simple as increasing key sizes. But for reasons outside the scope of this paper, asymmetric encryption will likely need a complete overhaul of standards [11].

There are currently two National Institute Standards and Technology (NIST) standardized algorithms used to generate quantum resistant digital signature schemes: Leighton-Micali Signature (LMS) and eXtended Merkle Signature Scheme (XMSS), which have the variants Hierarchical Signature System (HSS) and multi-tree XMSS (XMSSMT) [12]. Additionally, NIST’s PQC project has selected three digital signature algorithms and one public-key encryption/key-establishment algorithm, called CRYSTALS-KYBER, to begin the standardization process [11]. These four developing standards are generally considered more promising than the current, computationally expensive standards.

Conclusion

Maximizing security in automotive case presents unique cryptographic challenges. Due to resource limitations of hardware and communication networks; prioritization of message authentication, efficient public-key infrastructure, and secure chain of trusts is required to ensure automotive safety and performance requirements within a ZTA.

# References

|  |  |
| --- | --- |
| [1] | A. M. Nasser, Automotive Cybersecurity Engineering Handbook, Packt Publishing, 2023. |
| [2] | National Institute of Standards and Technology, "FIPS PUB 198-1," 2008. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.198-1.pdf. |
| [3] | M. Dworkin, "NIST Special Publication 800-38B," 5 2005. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf. |
| [4] | AUTOSAR, "Specification of Secure Onboard Communication Protocol," 30 11 2020. [Online]. Available: https://www.autosar.org/fileadmin/standards/R20-11/FO/AUTOSAR\_PRS\_SecOcProtocol.pdf. |
| [5] | D. B. Young, "Foundations of Computer Security, Lecture 45: Stream and Block Encryption," University of Texas at Austin, [Online]. Available: https://www.cs.utexas.edu/~byoung/cs361/lecture45.pdf. |
| [6] | National Institute of Standards and Technology (2001), Advanced Encryption Standard (AES). (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) NIST FIPS 197-upd1, updated May 9, 2023., [Online]. Available: https://doi.org/10.6028/NIST.FIPS.197-upd1. |
| [7] | A. Ahamed, "Public Key Algorithms in Cryptography," July 2020. [Online]. Available: https://aashikahamed.medium.com/public-key-algorithms-in-cryptography-5bb6877c5685. |
| [8] | D. Hankerson, A. Menezes and S. Vanstone, "Guide to Elliptic Curve Cryptography," Springer. [Online]. |
| [9] | B. Schneier, Applied Cryptography, vol. 2, Wiley. |
| [10] | Siddiqui, Mehrotra, Rahat and Suresh, "A Review on Elliptic Curve Cryptography for Embedded Systems," *International Journal of Computer Science & Information Technology (IJCSIT),* vol. 3, pp. 84-103, 2011. |
| [11] | "Post-Quantum Cryptography," National Institute Standards and Technology, 2023. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography. |
| [12] | D. Cooper, D. Apon, Q. Dang, M. Davidson, M. Dworkin and C. Miller, "Special Publication 800-208," National Institute of Standards and Technolgy, [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-208.pdf. |
| [13] | National Institute of Standards and Technology, Federal Information Processing Standards Publication, July 2013. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.186-4.pdf. |
| [14] | A. Han Vinck, "Introduction to Public Key Cryptology," University of Duisburg-Essen SVG. |