Hardware/Software Co-design for Quantum-Resistant Authentication in V2V Communications

by

Eliú Moreno Ramírez

A Dissertation Submitted in partial
fulfillment of the requirements for the
degree of

Master in Computer Science

At

Instituto Nacional de Astrofísica, Óptica y Electrónica
November, 2024
Tonantzintla, Puebla, México.

Advisored by:

Dra. Claudia Feregrino Uribe, INAOE
CONAHCYT Professorship for young researcher

Dr. Miguel Morales Sandoval, INAOE
CONAHCYT Professorship for young researcher

Agradecimientos

Primeramente, me gustaría agradecer a mi familia, en especial a mis padres, Martha y Salomón, quienes siempre velaron por mí, brindándome todo el apoyo necesario para seguir adelante. También quiero expresar mi profundo agradecimiento a mis asesores, la Dra. Claudia Feregrino y el Dr. Miguel Morales, por su paciencia, apoyo y valiosos consejos durante la realización de esta tesis. Además, quiero agradecer al Consejo Nacional de Humanidades, Ciencias y Tecnologías por la beca que me otorgaron para llevar a cabo mis estudios. Y a quien estuvo conmigo al principio dándome su apoyo incondicional.

Resumen

La autenticación es un servicio de seguridad informática que verifica la identidad antes de otorgar acceso a un sistema. La autenticación puede aplicarse no sólo a usuarios sino también a mensajes. Este es el caso de las redes vehiculares ad hoc (VANETs por sus siglas en inglés), donde los vehículos están equipados con mencanismos que permiten la comunicación entre sí. Dado que estos mensajes son vitales para la seguridad vial, se debe asegurar que dichos mensajes son auténticos.

La autenticación de mensajes en comunicaciones V2V está regulada por el estándar IEEE 1609, que prescribe el uso de firmas digitales basadas en el esquema criptográfico ECDSA. Sin embargo, el avance en cómputo cuántico amenaza la seguridad de este esquema. Se estima que una computadora cuántica podría comprometer su integridad en la década de 2030, lo que hace necesario desarrollar soluciones criptográficas resistentes a ataques cuánticos, como los esquemas de criptografía post-cuántica (PQC por sus siglas en inglés).

En esta tesis se propone una arquitectura Hw/Sw para un esquema de autenticación V2V parcialmente híbrido que combina criptografía convencional y PQC. Este esquema parcialmente híbrido es necesario porque los esquemas puramente PQC, aunque seguros, no cumplen con las restricciones de tiempo y tamaño en comunicaciones V2V. La arquitectura propuesta implementa la verificación de firmas PQC en hardware para mitigar el sobrecosto en tiempo de ejecución que implican. El sistema se validó en un SoC, logrando verificar 81 vehículos en 100 ms, una aceleración de 2.19× en comparación cuando la verificación de dichas firmas se implementan únicamente en software.

Abstract

Authentication is a computer security service that verifies identity before granting access to a system. Authentication can be applied not only to users but also to messages. This is the case with vehicular ad hoc networks (VANETs), where vehicles are equipped with mechanisms that allow them to communicate with each other. Since these messages are vital for road safety, it is essential to ensure that these messages are authentic.

Message authentication in V2V communications is regulated by the IEEE 1609 standard, which prescribes the use of digital signatures based on the ECDSA cryptographic scheme. However, advancements in quantum computing threaten the security of this scheme. It is estimated that a quantum computer could compromise its integrity in the 2030s, making it necessary to develop cryptographic solutions resistant to quantum attacks, such as post-quantum cryptography (PQC) schemes.

This thesis proposes a Hw/Sw architecture for a partially hybrid V2V authentication scheme that combines conventional cryptography and PQC. This partially hybrid scheme is necessary because purely PQC schemes, while secure, do not meet the time and size constraints in V2V communications. The proposed architecture implements PQC signature verification in hardware to mitigate the runtime overhead they entail. The system was validated on a SoC, achieving the verification of 81 vehicles in 100 ms, a 2.19× speedup compared to when the verification of these signatures is implemented solely in software.

Contents

CHAPTER 1

Introduction

The Fourth Industrial Revolution, or Industry 4.0, is a concept that describes how new technologies (AI, IoT, robotics, and Big Data) have transformed and promoted industrial processes [68]. These innovations have enhanced productivity through automation and predictive maintenance of equipment. In this era of technological change, vehicular connectivity has become highly relevant for addressing issues such as increasing traffic accidents and congestion [12, 99]. This connectivity facilitates communication between vehicles and road infrastructure, enabling the implementation of advanced solutions in fleet management, logistics, and intelligent transportation [1, 90]. An effective way to connect vehicles with different elements is through the Vehicular Ad Hoc Network (VANET), which plays a crucial role in this new era of smart mobility [97].

VANETs facilitate wireless communication among moving vehicles, a crucial aspect of Intelligent Transportation Systems (ITS), which aims to enhance driver safety [11]. The information exchange in VANETs supports a range of applications, both security and non-security related [89]. For example, these communications can be used to coordinate the delivery of goods, optimize transportation routes, and improve safety in industrial environments [69]. Additionally, VANETs have applications in areas such as healthcare, public safety, and emergency management. In healthcare, it can be used to send alerts about traffic accidents or medical emergencies to the nearest first responders, allowing for a faster and more efficient response [85, 41]. VANET facilitates communication between vehicles (V2V) and between vehicles and infras-

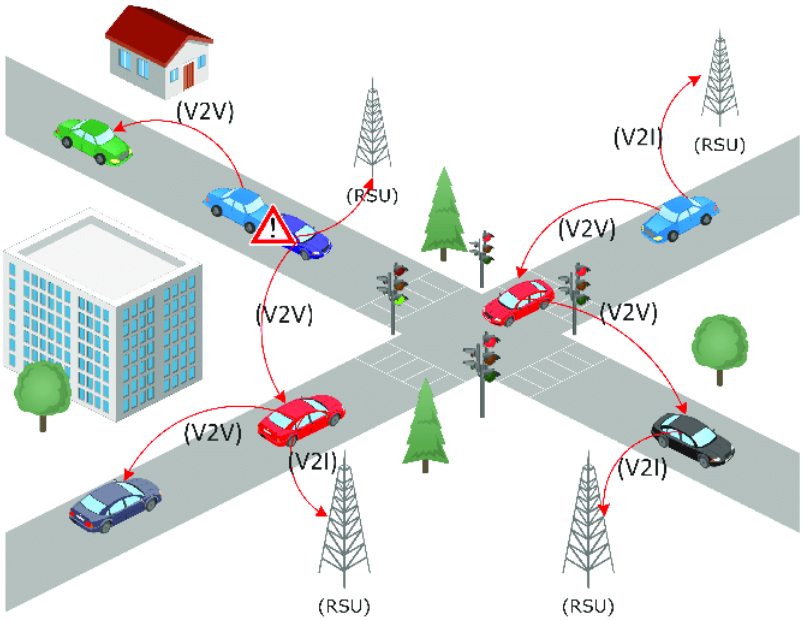


Figure 1.1: VANET diagram [93].

tructure (V2I). These communications are fundamental to the development of intelligent transportation systems and the improvement of traffic safety and efficiency. Figure 1.1 shows a VANET representation.

The vulnerabilities in V2V communications set out several risks for road security, for example, handling messages that provoke a dangerous situation on the highway, adding fake messages in the V2V communications to trick the receiving vehicles about the location, velocity, or accidents, the denial of service to some vehicles affects the coordination of traffic and highway security, or spoofing, which involves the creation of fake vehicles in the communication, which could generate misleading data about vehicle presence [57, 73]. Consider the scenario of Figure 1.2. Attacker's goal is to deny services to vehicle A, which involves blocking communication between vehicles B and C. Additionally, the attacker manipulates vehicles B and C to exchange false information with each other, causing an accident between them. The attacker then introduces a malicious vehicle, D, posing as a legitimate vehicle into the VANET network that sends false information to vehicles A and B [86].

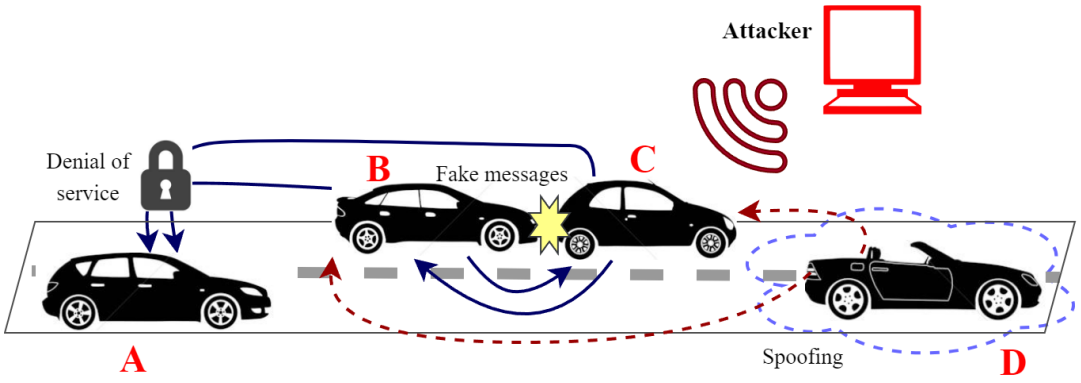


Figure 1.2: Attack models in V2V communications.

Implementing security measures and authentication in V2V communications is essential to mitigate risks such as the ones previously described and ensure road safety [82].

The authentication of messages in V2V communications using digital signatures is governed by the IEEE 1609.2 standard, which mandates the use of the elliptic curve digital signature algorithm (ECDSA) for message signing [47]. One of the main security mechanisms in V2V is the authentication of messages through the use of digital signatures, which is a type of public key cryptography.

Public key cryptography [79, 83], or asymmetric cryptography, is an encryption approach that uses two mathematically related keys: a public and a private key. Authentication from digital signatures is achieved as follows:

An entity A that wants to prove its identity to another entity B signs a piece of data using its private key. Then, the entity B verifies the A's signature using A's public key. The verification process fails if the public key used does not correspond to the signer's private key. More specifically, the digital signature process operates as follows:

The sender begins by generating a pair of cryptographic keys: a private key and a public key. To sign a message, the sender first applies a cryptographic hash function to the message, producing a hashed value. This hashed value is then encrypted with the sender's private key to create the digital signature. When the participant receives the signed message and the sender's public key, it starts by computing the hash value

of the received message. Next, the recipient decrypts the digital signature using the sender's public key to obtain the original hashed value. To verify the signature the recipient compares the decrypted hashed value with the computed hash of the message. The signature is valid only if both hashes match.

One of the widely employed digital signature algorithms is the ECDSA (Elliptic Curve Digital Signature Algorithm), which uses elliptic curve theory to generate cryptographic key pairs. It is recognized for its efficiency and strong security, making it suitable for diverse applications [40]. Furthermore, the IEEE 1609.2 standard mandates the use of digital certificates, an electronic document used to link unequivocally the identity of a vehicle to its public key. Digital certificates are issued by a trusted entity, such as a Certificate Authority (CA), as a way to transmit information about the vehicle and issue the public key securely and reliably as the CA signs it [2, 62]. One of the main uses of digital certificates is to avoid non-repudiation, that is, a vehicle denying the signature of a message by denying possession of a public key. To validate a digital certificate and then a signed message, the authenticity of the digital certificate must first be verified. This involves checking the digital signature of the certificate using the CA's public key. Once the certificate's authenticity is confirmed, the public key can be extracted from the certificate and used to verify the digital signature of the signed message. The IEEE 1609.2 standard also dictates the use of elliptic curve digital signatures for signing certificates.

In this protocol, a vehicle transmits its certificate along with a message containing information about its current status and the signature of the message. Upon receiving these, the recipient vehicle validates the certificate using the CA information and verifies the signature of the received message. Figure 1.3 provides a high-level overview of this process.

Elliptic curve digital signatures and other classical algorithms used in the current standard could face vulnerabilities due to the increase in computational power that quantum computing would offer [55, 96]. Quantum computing [32, 87] is progressing so fast that it provides new possibilities in the potency processing of information. Nevertheless, this development implies significant defiance in cybersecurity because quantum computers could solve the most usual problems based on public key cryptography (PKC) in polynomial time. For example, the last Google quantum

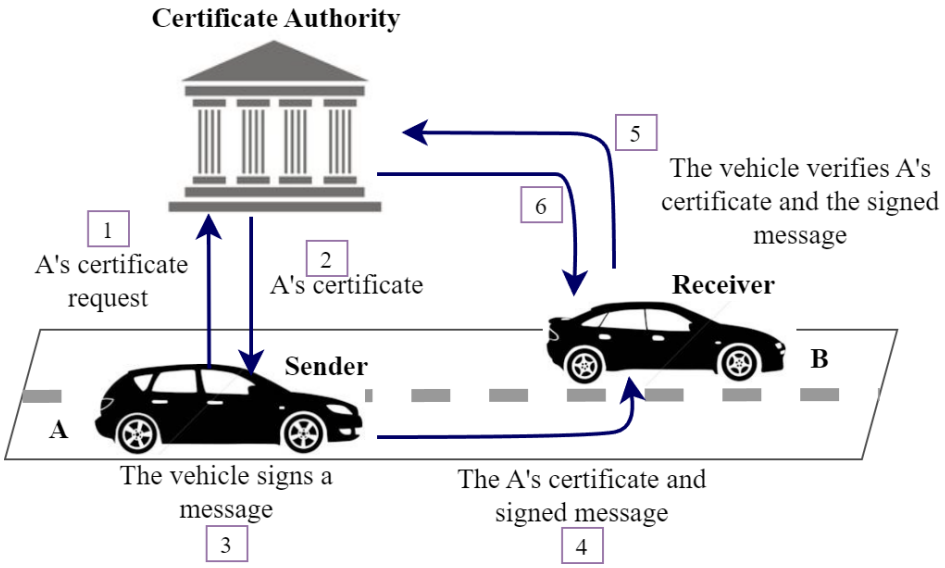


Figure 1.3: Model of V2V security protocol.

computer could solve a calculus in 200 seconds compared to a classic computer that could solve the same calculus in 10,000 years [74]. Also, companies such as IBM, Intel, and Microsoft work actively in standards for quantum computing [43]. As a result, the algorithms that have been fundamental for ensuring the security of information communication and authentication are vulnerable to quantum attacks. It is necessary to keep communication secure. Therefore, the National Institute of Standards and Technology (NIST) initiated a standardization process for post-quantum cryptography, aiming to ensure security against quantum computers [48].

Post-quantum cryptography (PQC) consists of hard problems, even for quantum computers. PQC approaches are classified according to the security fundamentals they rely on: multivariate, isogeny, lattices, error-correcting code, and hash-based functions [14, 22]. However, replacing conventional schemes with PQC is not straightforward due to the higher spatial and temporal complexities of these algorithms compared to conventional ones [64]. Depending on the application, adjustments at the algorithmic, protocol, or implementation level are necessary.

1.1 Problem statement

Public-key cryptography (PKC) has provided robust services in different domains. In the case of V2V communications, the IEEE 1609.2 standard specifies the use of elliptic curve cryptography for the authentication of exchanged messages through the use of the Elliptic Curve Digital Signatures Algorithm (ECDSA). However, ECDSA and other PKC algorithms are threatened by quantum computing. For example, ECDSA is vulnerable to the Shor algorithm [38], which could be executed in quantum computers in polynomial time. According to data from IBM [45], there is currently a computer with approximately 1121 qubits. Based on current exponential growth and estimates of the number of qubits needed, a quantum computer with 317 million qubits is expected to be available by 2038, which could break ECDSA in one hour [17, 39]. Figure 1.4 shows an estimated timeline of the evolution of post-quantum transition. Due to future threats, it is essential to transition from the actual standard (IEEE 1609.2) to post-quantum standards. Hence, in 2017, NIST started the process of post-quantum standardization by defining three safe algorithms for digital signatures that will be standardized: Crystals-Dilithium, Falcon, and Sphincs+, each with different levels of security [4].

There are various stages in the evolution of ECDSA vulnerability. The first stage is when ECDSA is safe, as quantum computers are not a risk, and the vehicles accept the actual standard (IEEE 1609). At the same time, the second stage exists when vehicles have hardware capable of using post-quantum algorithms because the useful lifespan of a vehicle is, on average, between 10 and 15 years [95]. As seen in Figure 1.4, there will be a period since a quantum computer could break the ECDSA signatures at the end of the decade of 2030. So will be vehicles that need classic verification and quantum-safe verification [100]. Many organizations, including the NIST, recommend the use of hybrid schemes, i.e., use a post-quantum and the actual algorithms for the stage of transition of post-quantum algorithms cryptography [44]. The final stage is when ECDSA will break, and a new standard for safe communication will be necessary.

The transition to quantum-secure V2V communication is complex because vehicles need to be capable of accepting the current standard and also supporting

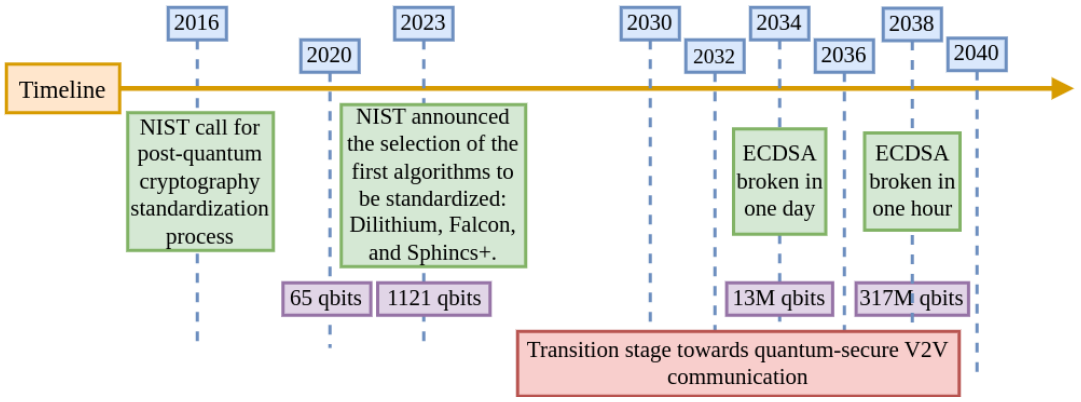


Figure 1.4: Estimated timeline of the post-quantum transition.

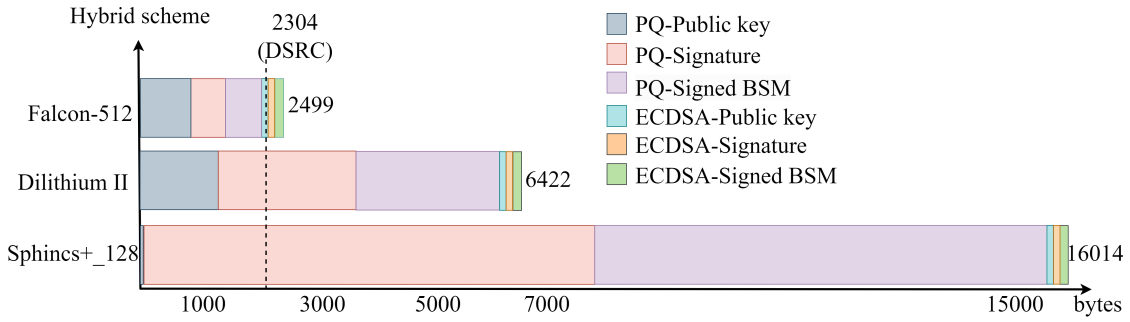


Figure 1.5: Hybrid schemes sizes.

PQC schemes. This is constrained by the protocol used for transmitting information data, as Dedicated Short Range Communications (DSRC) [59] has a payload limit of 2,304 bytes. For example, the key sizes and signature lengths of current PQC digital signature algorithms recommended by NIST [5] (such as Dilithium, Sphincs, and Falcon) are larger when compared to ECDSA at a 128-bit equivalent security level. If combined in a hybrid scheme for authentication, the aggregated size of post-quantum signatures and public keys exceeds the payload limit in current V2V communications, as depicted in Figure 1.5.

Furthermore, for viable V2V communication, it is suggested that a vehicle can sign ten basic messages (BM) per second and verify 100 vehicles in 100 ms [100], which could not be met by using a pure PQC approach.

1.2 Hypothesis

Under a transition scenario from classic to post-quantum security in V2V communication, it is necessary to 1) meet payload and verification time constrains and 2) support both current standard security specifications (ECDSA only) and PQC enabled crypto-engines as progressively available. A solution within the state of the art is a partially hybrid scheme, which could solve the payload problem by partitioning certificates and transmitting them over different messages, reducing and adjusting the payload. However, the problem of performance still exists since, in comparison to classical schemes such as ECDSA, post-quantum algorithms are more time consuming, causing a bottleneck due to inefficient verification, failing to achieve the requirement of 100 messages verified within 100 ms. There are hardware proposals to accelerate part or all of the verification of post-quantum algorithms. However, from the perspective of the end-to-end process of V2V authentication, it is necessary to define the specific operations to accelerate and perform the integration into software. Furthermore, optimizations are needed to achieve the required performance.

The hypothesis in this research is that a Hw/Sw co-design can enable the execution of V2V authentication in VANETs as recommended by current IEEE standards, meeting space requirements and time suggestions but considering the PQC transition.

Such a co-design takes advantage of hardware to increase the number of vehicles that can be verified in VANET environments at the time that the flexibility of the software allows implementing a hybrid approach to consider both classic and PQC cryptography to meet payload requirements. A co-design such as the one developed in this research has not yet been explored in the state of the art to address the problem of secure V2V communications during the transition to the quantum stage.

1.3 General objective

To develop a Hw/Sw co-design for efficient and quantum-resistant authentication in V2V communications in VANETs.

1.4 Specific objectives

- To determine the most suitable post-quantum algorithm, from those recommended by NIST, for the V2V communication authentication problem when implemented as a Hw/Sw co-design.
- To determine a Hw/Sw architecture for quantum-resistant message authentication in V2V communications.
- To create a functional prototype of the Hw/Sw co-design as a proof of concept to evaluate performance in terms of response time.

1.5 Methodology

The methodology in this thesis comprised the following general main parts:

1. Analysis of the verification protocol in V2V communications as recommended by the IEEE 1609.2.
2. Analysis of PQC digital signature schemes recommended by NIST.
3. Selection of the most suitable PQC-enabled approach for V2V authentication that meets constrains in terms of payload size and performance.
4. Design the hybrid authentication scheme and identify critical components for hardware acceleration.
5. Design of the Hw/Sw co-design architecture for a quantum-safe V2V communication protocol.
6. Validation and evaluation of proposed Hw/Sw architecture.

1.6 Document organization

The rest of this document is organized as follows: Chapter 2 addresses topics and concepts used in the following chapters: mathematical concepts, asymmetric cryptography, and computer security services; post-quantum cryptography along with its bases and standardization; vehicular communications and their standards used; and finally, the concept of Hw/Sw co-design. Chapter 3 reviews the works that propose

solutions as a replacement for the security standard in vehicular communications for the quantum era, followed by recommendations from the literature for the transition stage to the quantum era in V2V communications. Finally, the work related to the hardware acceleration of the post-quantum digital signature algorithms is presented. Chapter 4 describes the Hw/Sw co-design decisions, including the choice of the most appropriate post-quantum scheme, as well as the division of tasks executed in hardware and software in the proposed co-design. Chapter 5 presents the details of the prototype that demonstrated the viability of the proposed Hw/Sw co-design and its experimental evaluation, including the evaluation platform, the tools used, as well as the experimentation and results obtained. Chapter 6 summarizes the contributions of this thesis and outlines future research directions.

CHAPTER 2

Background

This chapter presents the concepts closely related to this research and useful for a correct interpretation of the subsequent chapters. First, mathematical concepts about PQC are presented. Then, the concepts of asymmetric cryptography, particularly the ones related to digital signatures for authentication are discussed. Next, the chapter discusses concepts related to PQC and its standardization process for digital signatures. Subsequently, the concept of Vehicular Ad-Hoc Networks (VANET), vehicle-to-vehicle communication, and the standard IEEE 1609.2 are presented, with special emphasis on the way the messages in V2V communications are authenticated. Also, the chapter discusses the importance of hardware/software co-designs (Hw/Sw), with a particular focus on those based on field-programmable gate arrays (FPGA) and system-on-chip (SoC). Finally, the chapter includes the material and method used for the co-design.

2.1 Mathematical fundamentals

Asymmetric cryptography, also referred to as public key cryptography, operates using a system that employs two different keys: a public key and a private key. It was designed to tackle the challenge of key exchange, which involves agreeing upon and calculating a single key used between two parties over an insecure channel. By utilizing two distinct keys, users no longer need to decide on a single key for an encrypted communication.

The functioning of public key cryptography is based on algebraic structures where one-way functions f are defined, i.e., functions for which there are no polynomial algorithms that compute f^{-1}. These functions become the foundation of security for public key cryptographic algorithms. This section presents the algebraic structures that underpin public key cryptography, highlighting the importance of these structures in the algorithms used for V2V communication authentication as well as those intended to replace the current standard.

Definition 2.1.1 (Group) *A group is a non-empty set G with a binary operation $*$, denoted as $(G, *)$, that satisfies the following properties:*

- *Closure:* $\forall a, b \in G : a * b \in G$.
- *Associativity:* $\forall a, b, c \in G : (a * b) * c = a * (b * c)$.
- *Identity element:* $\exists e \in G : \forall a \in G : a * e = e * a = a$.
- *Inverse:* $\forall a \in G : \exists c \in G : a * c = c * a = e$.

Definition 2.1.2 (Subgroup) *Let G be a group with a binary operation $*$. A non-empty subset $H \subseteq G$ is said to be a subgroup of G if $(H, *)$ forms a group.*

Definition 2.1.3 (Semigroup) *A semigroup is a non-empty set G with a binary operation $*$, denoted as $(G, *)$, that satisfies the properties of closure and associativity.*

Definition 2.1.4 (Ring) *$(A, *, \star)$ is a ring if $(A, *)$ is a group and $(A, \star)$ is a semigroup.*

Definition 2.1.5 (Field) *$(F, *, \star)$ is a field if $(F, *)$ is a group and $(F, \star) - \{e\}$ is a group, be e the identity element in $(F, \star)$.*

Definition 2.1.6 (Finite Field) *A finite field consists of a finite number of elements. It is defined by its cardinality, denoted as $|F|$, and by two binary operations $(*, \star)$ defined on F that satisfy all the properties of a field.*

Definition 2.1.7 (Quotient Ring of Integers) *It is a ring denoted as $\mathbb{Z}_q$ formed by dividing the ring of integers $\mathbb{Z}$ by an ideal $q\mathbb{Z}$ (the set of all integer multiples of an integer q).*

Definition 2.1.8 (Polynomial Ring) *A polynomial ring is constructed from polynomials that involve one or more variables, where the coefficients are drawn from another ring, typically a field.*

Definition 2.1.9 (Irreducible Polynomial) *If F is a field, a non-constant polynomial $f(x) \in F[x]$ is irreducible over F if its coefficients are in F, and it cannot be factored into the product of two non-constant polynomials in $F[x]$.*

Definition 2.1.10 (Quotient Polynomial Ring) *The quotient ring $K[X]/(p)$, where p is a polynomial of degree d, can be understood as the vector space of polynomials of degree less than d, with multiplication defined modulo p.*

Definition 2.1.11 (Affine Plane) *An affine plane P is a geometric structure with a set of points and lines that satisfies: 1) each pair of distinct points determines a unique line, 2) each line contains at least two points, and 3) for any point not on a given line, there is exactly one line parallel to the given line passing through that point.*

Definition 2.1.12 (Elliptic Curve) *Let F be a field with $car(F) \neq 2, 3$, where the function $car(F)$ expresses the characteristic of the field F, which is a number $n \in \mathbb{Z}$ such that $1 + \overbrace{\ldots}^{n} + 1 = 0$. The elliptic curve EC defined over F in the affine plane P is defined as $EC(F) = \{(x, y) \in A_K^2 \mid y^2 = x^3 + Ax + B\}$ where $A, B \in F$.*

Definition 2.1.13 (Lattice) *Lattices are discrete n-dimensional subgroups in $\mathbb{R}^n$. Specifically, a lattice in $\mathbb{R}^n$ is generated by a basis $\mathbf{B} = \{b_1, b_2, \ldots, b_n : b_i \in \mathbb{R}^m\}$ and is defined as $\boldsymbol{\mathcal{L}}(\boldsymbol{B}) = \{a_1 b_1 + \ldots + a_n b_n : a_i \in \mathbb{Z}\}$. Here, n denotes the lattice rank, and m denotes the lattice dimension.*

Definition 2.1.14 (Ring lattices) *For the rings $\mathbb{Q} = \mathbb{Q}[x]/(\phi)$ and $\mathbb{Z} = \mathbb{Z}[x]/(\phi)$, where $m, n \in \mathbb{Z} \setminus \{0\}$ with $n \leq m$, and for a full-rank matrix $\mathbf{B} \in \mathbb{Q}^{n \times m}$, the lattice $\Lambda(\mathbf{B})$ generated by $\mathbf{B}$ is the set $\mathbb{Z}^n \cdot \mathbf{B} = \{\mathbf{zB} \mid \mathbf{z} \in \mathbb{Z}^n\}$.*

Definition 2.1.15 (FFT) *Fast Fourier Transform is defined as following: let $f \in \mathbb{Q}[x]/(\phi)$, Ω_ϕ the set of the complex roots of ϕ, ϕ is monic such as $\phi = \prod_{\zeta \in \Omega_\phi}(x - \zeta)$. FTT is defined as:*

$$FFT_\phi(f) = (f(\zeta))_{\zeta \in \Omega_\phi} \tag{2.1}$$

Definition 2.1.16 (NTT) *The Number Theoretic Transform (NTT) operates within a finite field $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$. It is constructed by substituting complex arithmetic used in the Fast Fourier Transform (FFT) with modular arithmetic.*

Definition 2.1.17 (Isogeny) *Given two commutative groups G and H, a group homomorphism $f : G \to H$ (i.e., a mapping that preserves the group operation) is called an isogeny if it satisfies two conditions: First, the homomorphism f is surjective, meaning that for every element $h \in H$, there exists at least one element $g \in G$ such that $f(g) = h$. Second, each fiber $f^{-1}(h)$ for $h \in H$ contains a finite number of elements.*

Definition 2.1.18 (Hash) *A cryptographic hash function $\mathcal{H}$ is a computable function defined by an algorithm such that $\mathcal{H} : \{0,1\}^* \to \{0,1\}^M$. The hash function takes as input an element from $\{0,1\}^*$ (a binary string of variable length) and maps it to an element in $\{0,1\}^M$ (a binary string of fixed length M).*

The key properties of hash functions include being 1) deterministic, meaning they consistently produce the same output for the same input; 2) fixed output size, independent of the input size; 3) preimage resistance, making it computationally impractical to find any input x for a given hash value $\mathcal{H}(x)$; 4) second preimage resistance, where given an input x and its hash $\mathcal{H}(x)$, it is computationally infeasible to find a different input x' such that $\mathcal{H}(x') = \mathcal{H}(x)$; and 5) collision resistance, which makes it computationally infeasible to find two distinct inputs x and x' that produce the same hash value.

2.2 Digital signatures and digital certificates

Asymmetric cryptography can provide authentication services through a digital signature scheme. It provides the security to the recipient that the message came from a verified sender. The digital signature scheme consists of a tuple of algorithms $\mathcal{S} = (KGen, Sign, Vrfy)$ executed by A, which is defined as follows:

- $KGen$ returns A's key pair, the private key sk, and its corresponding public key pk.

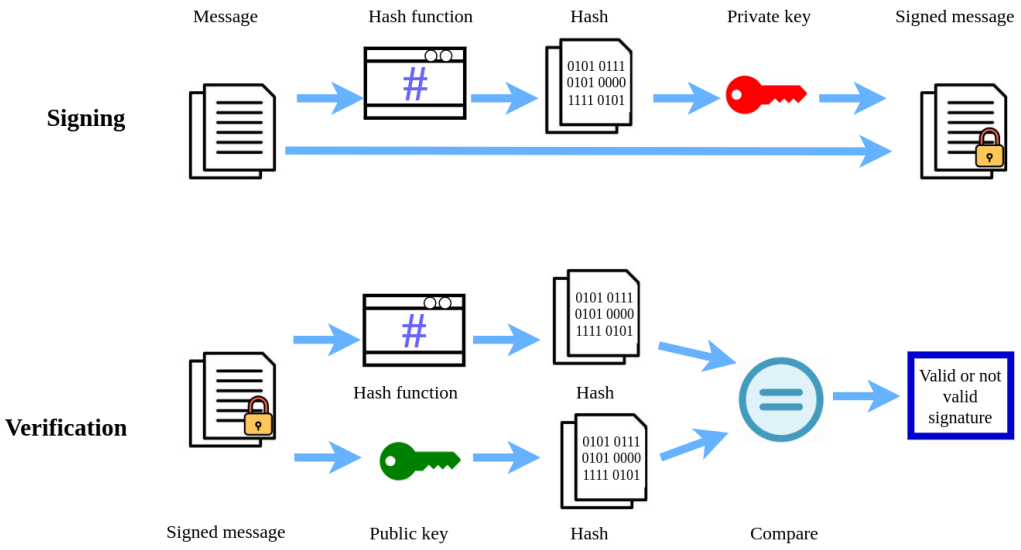


Figure 2.1: Digital signature scheme. Above is the procedure for signing a message. Below is the process for verifying the signature.

- $Sign$ returns a signature sig for a message m using A's private key sk.
  - Compute the hash of the message: $y = H(m)$.
  - Encrypt y using sk.
- $Vrfy$ returns a boolean value depending on whether A's signature sig is successfully verified or not, respectively, using as input a message m, a signature sig, and a public key pk.
  - Compute the hash of the received message: $y' = H(m)$.
  - Verify the signature using the public key and the message hash:
    - Decrypt the signature to obtain the signed hash y.
    - Return true only if $y = y'$, indicating that the signature is valid.

This procedure is illustrated in Figure 2.1. Here on, the terms *digital signature* and *signature* will be used interchangeably.

A digital certificate scheme relies on digital signature scheme services and plays a crucial role in securely transmitting public keys. A digital certificate is essentially the digital signature of a certificate authority on an individual's public key, coupled with the identity of the public key's owner. Throughout this document, the terms

digital certificate and *certificate* will be used interchangeably.

In this document, the notation *Entity.Operation()*, means that *Entity* executes the *Operation()*. With this notation, the functioning of a digital certificate is outlined as follows:

Let S_U and S_{CA} be the signature schemes for a user U and a certification authority CA, respectively. The user U generates its keys with $(sk_U, pk_U) \leftarrow S_U.KGen()$, and the certification authority CA generates its keys with $(sk_{CA}, pk_{CA}) \leftarrow S_{CA}.KGen()$. The user's certificate, generated by CA, is $C_U \leftarrow S_{CA}.Sign(sk_{CA}, I_U)$, where $I_U = \{U_{identity}, pk_U\}$. The certificate of CA must also be created, and for simplicity, a self-signature is assumed; however, it could be generated by another certification authority, producing an authentication path as $C_{CA} \leftarrow S_{CA}.Sign(sk_{CA}, I_{CA})$, where $I_{CA} = \{CA_{identity}, pk_{CA}\}$. From this, a function is defined to obtain the public key endorsed by the certificate: $pk_{CA} \leftarrow C_{CA}.getPK()$ and $pk_U \leftarrow C_U.getPK()$. Additionally, a function to obtain the certificate's signature is necessary: $sig_{C_U} \leftarrow C_U.getSignature()$.

1. With the above, a message signature from U is as follows:
   (a) U sign the message: $sig \leftarrow S_U.Sign(m, sk_U)$.
   (b) The user transmits (m, sig, C_U, C_{CA}) over a communication channel, from which a potential receiver R can later verify the message.
2. The receiver R verifies the received message by doing the following:
   (a) Obtains both public keys: $pk_{CA} \leftarrow C_{CA}.getPK()$ and $pk_U \leftarrow C_U.getPK()$.
   (b) Verifies C_U using $b_1 \leftarrow S_{CA}.Vrfy(C_U, C_U.getSignature(), pk_{CA})$.
   (c) If b_1 is false, then m is rejected; otherwise, R proceeds with the verification of sig using $b_2 \leftarrow S_U.Vrfy(m, sig, pk_U)$.
   (d) The receiving user accepts m only if b_2 is true.

2.3 ECDSA

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a public key algorithm proposed by Scott Vanstone in 1992 [53]. ECDSA has been proven more effective than the Digital Signature Algorithm (DSA), the actual standard for digital signatures,

providing the same security with a smaller key size [70].

2.3.1 Algorithm

The functioning of ECDSA is as follows:

(a) Public parameters
- EC is the elliptic curve used.
- G serves as the base point of the elliptic curve and acts as its generator, representing a point on the curve that generates a subgroup of large prime order n.
- n is the order of G, such that $n \times G = O$, where O denotes the identity element.
- d_A is the private key associated with the elliptic curve cryptography.
- Q_A is the public key related from d_A.
- m represents the message intended to be transmitted or signed.
- *Hash( )* is a hash function that returns an integer.

(b) Key pair generation: The private key d_A is selected as an element within the interval $[1, n-1]$ and, the public key is calculated by $Q_A = d_A \times G$, where G is a predefined point on the elliptic curve.

(c) Signature generation: Calculate $e = \text{Hash}(m)$, where m is the message to be signed. Next, choose a random element $k \in [1, n-1]$. Compute the curve point $(x_1, y_1) = k \times G$, and then determine $r = x_1 \mod n$. If $r = 0$, return to the previous step and choose a new k. Let z be the leftmost L_n bits of e, where L_n denotes the bit length of the group order n. Next, calculate $s = k^{-1}(e + rd_A) \mod n$, and if $s = 0$, return to the step of choosing a new k. The resulting signature is the pair (r, s).

(d) Verification: To verify a signature, begin by checking if the following conditions are met; otherwise, reject the signature. Ensure that $Q_A \neq 0$, $Q_A \in EC$, and $n \times Q_A = 0$. Next, verify that $r, s \in [1, n-1]$. Compute the hash value $e = \text{Hash}(m)$ for the message m, and let z represent the leftmost L_n bits of e. Then, calculate $u_1 = zs^{-1} \mod n$ and $u_2 = rs^{-1} \mod n$. Compute the

elliptic curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$. If $(x_1, y_1) = 0$, the signature is invalid. The signature is considered valid if $r \equiv x_1 \mod n$; otherwise, it is invalid.

2.3.2 Security of ECDSA

The security of ECDSA is based on the Elliptic Curve Discrete Logarithm Problem (ECDLP) defined as follows:
ECDLP: Given a generator P of an elliptic curve E over finite field $\mathbb{F}_p$, and a point Q in E, the objective is to find an integer k such that $Q = kP$.

There are two primary types of attacks on digital signatures: key-only attacks, where the adversary has only the public key, and message attacks, where the adversary gains access to some signatures prior to compromising the function.

Several methods can be used to breach a digital signature, including obtaining the secret key, developing an alternative signing algorithm with an equivalent secret key, generating a forged signature for a specific message, or producing a fraudulent signature for one or more messages. The ECDSA algorithm relies on several security assumptions to ensure its effectiveness. Firstly, the security of ECDSA is based on the discrete logarithm problem in the subgroup generated by G, which is computationally difficult to solve. This difficulty ensures that retrieving the secret key from the public one is not feasible. Secondly, the algorithm uses a hash function that is both collision-resistant and unidirectional, further strengthening the security of the digital signature.

A quantum computer running Shor's algorithm can be generalized to discrete logarithms; it can be used to efficiently solve the discrete logarithm problem on elliptic curves in approximately time of $\mathcal{O}(n^3)$ [35].

ECDSA's security is based on its mathematical complexity and the robustness of the hash function used. However, the advent of quantum computers introduces a potential threat, as Shor's algorithm can efficiently solve the discrete logarithm problem on elliptic curves in polynomial time over a quantum computer [42]. Thus, as quantum technology advances, ECDSA could face significant security challenges.

2.4 Post-Quantum Cryptography

Due to the progress of quantum computing, as mentioned above, PQC is a possible solution. It is divided into five principal categories [10], according to the problems used as security basis for which no efficient algorithm in the quantum are found:

1. Isogeny-based: This approach relies on elliptic curve theory and isogenies, which involve the challenge of computing an isogeny of a specific degree between two isogenous supersingular elliptic curves [30].
2. Code-based: It is based on error-correcting codes for use in public key cryptography. The difficulty is based on decoding a message that contains an aleatory error and recovering the code structure, "coding theory", for example, syndrome decoding (SD) and learning parity with noise (LPN) [20].
3. Laticce-based: It is based on problem lattices like the Shortest Vector Problem (SVP). The difficulty is based on handling vectors in a large number of dimensions [21].
4. Multivariate Polynomial: The "multivariable" is referred to a polynomial with more than only one variable. When it is used in public key cryptography, the public keys are represented by polynomials, not linear [29].
5. Hash-based: It is based on hash functions; as Merkle signature, these schemes are quantum computing resistant [77].

Nevertheless, PQC faces a challenge due to the computational overhead associated with implementing these algorithms compared to actual standards. Therefore, it is crucial to develop solutions that mitigate the computational cost of these algorithms, especially if performance is a concern, as in the V2V communication scenario.

2.4.1 Standardization

The NIST has advocated for the adoption of PQC algorithms as early as the end of the decade 2030, foreseeing the potential breach of current cryptography standards by that time [37]. NIST started a process of standardization for PQC in 2016; the objective was to find better algorithms for public key cryptography, exchange keys, and digital signatures. All proposals were received and evaluated. In February 2019,

NIST announced 26 candidates for a second round, considering cost, performance, and implementation techniques. In July 2022, the NIST published four algorithms to be standardized: Crystals-Kiber for public key encrypts, Crystals-Dilithium, Falcon (both lattice-based), and Sphincs+ (based-hash) for digital signature.

2.4.2 Security fundamentals

Next, the problems used by the three standardized PQC algorithms are discussed.

2.4.2.1 Lattice-based theory

With the lattice definition presented above, the hardness of a cryptographic system lattice-based consists of solving four NP problems.

1. **Learning With Errors (LWE)**: Consider the quotient ring $\mathbb{Z}_q^m$, let $A_{n\times m}$ be a matrix where each matrix element $a_{ij} \in \mathbb{Z}_q$, let $s \in \mathbb{Z}_q^n$, let $\mathcal{D}$ be an error distribution in $\mathbb{Z}_q^m$ taking an error vector $e \in \mathcal{D}$, define a function $\mathcal{F} : \mathbb{Z}_q^n \to \mathbb{Z}_q : \mathcal{F}(s,e) = sA + e = b$ is also called LWE such that $\mathcal{F}(s,e) = LWE(s,e) : sA + e = b$, the problem is to find the pair (s,e) from (A,b).
2. **Ring Learning With Errors (RLWE):** RLWE is a variant of the Learning With Errors (LWE) problem. Let $a_i(X)$ be a set of random but known polynomials from $K_q[X]/p$ with coefficients in K_q, $e_i(X)$ be a set of small random and unknown polynomials from $K_q[X]/p$, $s(X)$ be an unknown small polynomial relative to the ring $F_q[X]/p$, and $b_i(X) = (a_i(X) \cdot s(X)) + e_i(X)$. This search version involves finding the unknown polynomial $s(X)$ given the list of pairs of polynomials $(a_i(X), b_i(X))$.
3. **NTRU**: Given a parameter $n = 2^k$, $q \in \mathbb{N}$ and a polynomial $\phi(x) = x^n + 1$, the problem consist of determining four polynomial $(f, g, F, G \in R)$, such as f is invertible modulus q, satisfying the next equation:

$$fG - gF = q \mod \phi$$

4. **SelfTargetMSIS**: Problem that consist of finding a vector $\begin{pmatrix} z & c & v \end{pmatrix}^T$ with

small coefficients and a message μ satisfying

$$\mathcal{H}\left(\mu \parallel k[A \mid t \mid I] \cdot \begin{pmatrix} z & c & v \end{pmatrix}^T\right) = c$$

where A and t are uniformly random and I is the identity matrix.

2.4.2.2 Hash-based theory

With the hash definition presented above, the security of a hash-based cryptography system is evaluated based on the following principles related to the hardness of hash functions or structures derived from hashes:

1. **Winternitz One-Time Signature (WOTS)**: Digital signature scheme proposed by Ralph Winternitz in 1986. Each private key can only sign one message in that scheme: "on-time signature".
2. **A hash tree**: Also known as Merkle tree proposed in 1979 by Ralph Merkle [76], is a structure tree in which each node "leaf" is the hash value of a data block and each node "branch" is the hash value of its child nodes. This action is repeatedly used until the tree root is calculated. A tree hash allows an efficient verification for a large data structure.
3. **Extended Merkle Signature Scheme (XMSS)**: Cryptography scheme that allows the signing of many but limited messages. It uses four components: WOTS, two functions hash, and a pseudorandom function. Each key pair (private/public) is associated with a binary tree, with each node having a hash value: the leaf contains an especially hash associated with a public key WOTS, and each node no-leaf contains the hash value of its children.

2.4.3 Falcon

The security of Fast-Fourier Lattice-based Compact Signatures over NTRU (Falcon) [33] is based on lattices for digital signatures over NTRU lattices. The functioning of Falcon is as follows:

(a) Public Parameters

- Falcon works with elements in $\mathbb{Q}[x]/(\phi)$, with $\phi = x^n + 1$ for $n = 2^k$.
- A modulus $q \in \mathbb{N}^*$.
- $\beta \in \mathbb{N}$ is a boundary such as $\lfloor \beta^2 \rfloor > 0$..
- σ and $\sigma_{min} < \sigma_{max}$ are standard deviations.
- $H()$ represent a hash function.

(b) Key pair generation: Compute $f = \sum_{i=0}^{n-1} f_i x_i$ and $g = \sum_{i=0}^{n-1} g_i x_i$, where f_i and g_i are generated from a Gaussian distribution. Verify that f is invertible modulo q. Next, compute the NTRU equation given f and g. Construct the matrix $\hat{\mathbf{B}}$ using the fast Fourier transform, calculated for g, G, f, and G as follows:

$$\hat{\mathbf{B}} = \left[\begin{array}{c|c} \mathrm{FFT}(g) & -\mathrm{FFT}(f) \\ \hline \mathrm{FFT}(G) & -\mathrm{FFT}(F) \end{array}\right].$$

Then, compute the Falcon tree. Starting with $\mathcal{G} \in M_{2,2}(\mathbb{Q}[x]/(\phi))$, construct the Falcon tree T. The root of this tree is $\tilde{L}$, and its two child nodes, G_0 and $G_1 \in M_{2,2}(\mathbb{Q}[x]/(x^{n/2}+1))$, are derived from the LDL decomposition of D_{11} and D_{22}, which allows expressing a matrix as the product of a lower triangular matrix L, a diagonal matrix D with positive entries, and the transpose of L (L^T). By iterating on G_0 and G_1, the entire tree T is formed, where each leaf $l \in \mathbb{Q}$. Compute $h = gf^{-1} \mod q$, where the private key is $sk = (\hat{\mathbf{B}}, T)$ and the public key is $pk = h$.

(c) Signature generation: Given a private key sk and a message m, the signer uses sk to sign m. First, generate a random nonce r uniformly from $\{0,1\}^{320}$. Next, hash the concatenated string $(r \| \| m)$ to produce a polynomial $c \in \mathbb{Z}_q[x]/(\phi)$, where $c = H(r \| \| m, q, n)$. Compute t such that $\tilde{B}t = c$. Then, compute z using Fast Fourier Sampling (ffSampling), which samples signature coefficients from a Gaussian distribution. The bitstrings s' are obtained by compressing s_2, resulting in a transformation to $(8 \cdot b_l - 328)$ bits. Finally, the signature is represented by the pair (r, s').

(d) Verification: Given a public key $pk = h$, a message m, a signature $sig = (r, s')$, and a threshold $\lfloor \beta^2 \rfloor$, the verifier uses pk to verify the signature sig of the

message m as follows: First, concatenate the nonce r and the message m into a string $(r\|\|m)$, which is then hashed to produce a polynomial $c \in \mathbb{Z}_q[x]/(\phi)$. Next, decompress s' to obtain the polynomial $s_2 \in \mathbb{Z}_q[x]/(\phi)$. Compute the value $s_1 = c - s_2 h \mod q$. The signature is considered valid if $\|(s_1, s_2)\|\|^2 \leq \lfloor \beta^2 \rfloor$; otherwise, it is rejected.

2.4.4 Dilithium

The security of CRYSTALS-Dilithium [31] relies on the difficulty of two lattice problems: SelfTargetMSIS and LWE problems. The functioning of Dilithium is as follows:

(a) Public parameters
- A polynomial ring $R_q = \mathbb{Z}_q[x]/(x^{256} + 1)$, where q is a prime satisfying $q \equiv 1 \mod 512$.
- Let $k, l \in \mathbb{N}$.
- $G()$ and $H()$ represent hash functions.
- Parameters γ_1, $\gamma_2 \in \mathbb{Z}$.
- $d \in \mathbb{Z}_q$, η, Ω and $\beta \in \mathbb{Z}$ are boundary.

(b) Key pair generation: Select two random strings ρ and θ from $\{0,1\}^{256}$. Next, use ρ to construct a matrix $\mathbf{A}^{k \times l}$, where each entry is a polynomial in $R_q = \mathbb{Z}_q[X]/(X^n + 1)$. Then, choose two random secrets s_1 and s_2 from R_q^l, ensuring that each element of s_1 and s_2 is less than or equal to η. Compute t as $\mathbf{A}s_1 + s_2$. Next, calculate $t_0 = t \mod 2^d$ and $t_1 = \frac{t - t_0}{2^d}$. The public key is then $pk = (\rho, t_1)$, and the private key is $sk = (\rho, \theta, G(p\|\|t_1), s_1, s_2, t_0)$.

(c) Signature generation: Given the keys (sk, pk) and a message M, the process to compute the signature involves several steps. First, compute A from ρ. Then, calculate μ as $G(G(\rho\|t_1)\|M)$, and subsequently compute $p' = G(\theta\|\mu)$. Select a short vector $y \in \mathbb{R}_q^l$ with coefficients less than γ_1, using the NTT representation from ρ. The signer then computes $\omega = Ay$. The hash value c is determined as $H(\texttt{High(w)}, M)$, where `High(w)` represents the high bits of ω. Next, calculate z as $y + cs_1$. Compute both high and low bits, $r_1 = \texttt{High}(\omega - cs_2)$ and $r_0 = \texttt{Low}(\omega - cs_2)$, and ensure they satisfy the conditions; if not, repeat the selection of y. Specifically, check that $\|z\|_\infty < \gamma_1 - \beta$, $\|r_0\|_\infty < \gamma_2 - \beta$, and

$r_1 = \omega_1$. Compute h as $r_1 \oplus \texttt{High}(\omega - cs_2 + ct_0)$. Finally, compute $\Omega' = w(h)$ and if $\Omega < \Omega'$, repeat the process from step 3. The resulting signature is $sig = (z, h, c)$.

(d) Verification: Given a signature sig on a message m, follow these steps: First, calculate A and μ. Then, compute $\omega' = \texttt{High}(\omega - cs_2) = \texttt{High}(Az - ct_1 \cdot 2^d) = \texttt{High}(\omega - cs_2 + ct_0)$. Next, verify that the following conditions are met for the recipient to accept the signature: $\|z\|_\infty < \gamma_1 - \beta$, $c = H(\mu \| \omega_1')$, and $\Omega' = w(h)$ should be such that $\Omega \leq \Omega'$.

2.4.5 Sphincs$^+$

The security of Sphincs$^+$ [15] is based on a combination of digital signature schemes: WOTS and XMSS. The functioning of Sphincs$^+$ is as follows:

(a) Public parameters
- A hypertree, witch comprises d Merkle trees, where each tree has a height of h'.
- A FORS tree, that consists of k parallel trees with a height of a.
- A length n for each hash value, determining the security level.
- The Winternitz parameter w represents a trade-off parameter, with larger values resulting in shorter signatures but slower signing processes.

(b) Key generation: The key generation process assumes the existence of a function `secRand` that, given n as input, provides n bytes of cryptographically secure randomness. First, compute SK.seed = secRand(n), which generates all the WOTS+ and FORS private key elements. Next, compute SK.prf = secRand(n) to produce a randomization value for the hashed message. Then, calculate PK.seed = secRand(n), which serves as the public seed. Subsequently, calculate PK.root, representing the hypertree root, specifically the XMSS root of the top-level tree. The private key is $sk = (SK.\text{seed}, SK.\text{prf}, PK.\text{seed}, PK.\text{root})$, while the public key is $pk = (PK.\text{seed}, PK.\text{root})$.

(c) Signature generation: Given the private key sk and a message m, the signature is calculated using the following steps: First, compute R, an n-byte pseudo-random string generated from SK.prf and m. Next, calculate the digest of the

message m. Then, compute $SIGFORS$, which represents a FORS signature applied to the initial ka bits of the digest. Subsequently, derive $PKFORS$ from $SIGFORS$, which corresponds to the public key associated with the FORS signature. Finally, calculate $HTSIG$, a hypertree signature applied to $PKFORS$. The resulting signature is $sig = (R, SIGFORS, HTSIG)$.

(d) Verification: Given a signature sig on a message m, follow these steps: First, retrieve R, which corresponds to the initial n bytes of sig. Next, obtain $SIGFORS$, which consists of the subsequent $k(a+1) \cdot n$ bytes of sig. Then, calculate the digest of the message m. Use $SIGFORS$ and the initial ka bits of the digest to derive $PKFORS$. Finally, utilize $PKFORS$ to verify the hypertree.

2.5 Security in VANETs

A vehicular ad-hoc network (VANET) is a communication network where vehicles are the main nodes. These networks offer extensive services since they are key part of intelligent transportation systems, allowing vehicles to efficiently acquire information to enable applications such as traffic information in real time, ease of driving, and road safety [34]. In addition, these networks can provide driver assistance and even allow autonomous driving at some point [63]. VANETs have two main components: vehicles and infrastructure. Vehicles are equipped with onboard communication units (OBUs), while infrastructure can communicate via satellites or via pole-mounted units called Roadside Units (RSUs). This network offers various types of communication [52], such as vehicle-to-infrastructure (V2I) communication, which facilitates the transmission of crucial information about traffic status and other road conditions. Vehicle-to-pedestrian (V2P) communication is also promoted, which promotes safety and coordination between vehicles and pedestrians. Vehicle-to-Vehicle (V2V) communication allows cars to exchange messages directly with each other.

V2V communications imply exchanging information, such as position, velocity, direction, information about traffic, accidents, works on the road, traffic marks, and emergencies. This information is processed and analyzed in a platform and then

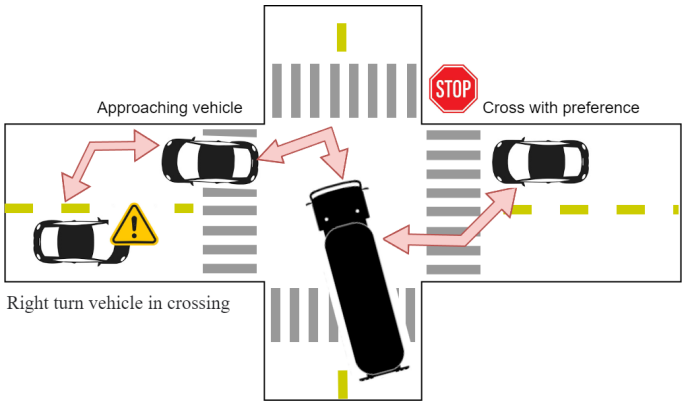


Figure 2.2: Applications of V2V communications.

distributed to the rest of the vehicles that form part of VANET [49]. For the V2V implementation to be effective, the road must have many vehicles capable of communicating with each other. In addition, communication must meet the requirements in this domain, such as the delay in the exchange of information and the potency for validating and processing it. Figure 2.2 shows a model of V2V communication for road safety.

To implement VANETs, communication technologies have been defined, such as Dedicated Short Range Communications (DSRC) and Cellular Vehicle to Everything (C-V2X). These technologies are the principal protocols for vehicle-to-vehicle (V2V) communication worldwide and are specified within the Media Access Control (MAC) layer. DSRC is a protocol based on IEEE 802.11 and designed for high-mobility V2V environments with a range of approximately 1000 m. It is the most commonly used protocol for V2V communication in the United States, Europe, and Japan [100]. In addition, DSRC indicates the datum package limit of 2,304 bytes [46]. This technology is subject to multiple standards, one of which is a standard for exchanging messages between network nodes, SAE J2945 [24], which mentions the use of a Basic Safety Message (BSM) that has all the information that a vehicle could need. Another is the IEEE 1609 family of standards, which dictates, among other things, mechanisms and algorithms of security for the exchange of BSMs. Every BSM comes with its signature and a certificate. A CA must transmit certificates to the sender

to append one of these in a BSM; the suggestion is 20 certificates with a validity of a week [19]. Also, this certificate must be a pseudonym, giving the sender anonymity. A Secure Protocol Data Unit (SPDU) is used for transmitting data in vehicular networks. Each certificate rotates every five minutes, and according to industry standards, a certificate is attached to every fifth message [24, 50]. Specifically, a vehicle includes its complete certificate in one out of every five SPDU transmissions. In all other transmissions, only a hash value of the certificate is included. This approach is referred to as a five-message cycle. In addition, IEEE 1609 stipulates that the generation of signatures for the messages and certificates must use ECDSA, with curves that provide 128-bit level security.

2.6 Hw/Sw co-design

The Hw/Sw co-design concept emerged early in the 1990s for systems that combine hardware and software devices for a specific task using a tool to compile software and synthesize hardware [103]. These tasks can be costly in software and can be accelerated by hardware. The accelerators are designed to execute specific operations faster and more efficiently, enjoying the hardware's parallel processability. At the same time, the co-design looks to keep the flexibility provided by the software, which is a tool more powerful to resolve an amplifier range of problems than does hardware, which modification requires physical changes in its components by the other hand, the software can be changed, updated, and reconfigured. [27]. Most embedded systems are formed by hardware and software components that operate simultaneously and cooperatively [23]. The systems are directly involved with the environment or other devices, unlike other devices that interact primarily with the end user, such as medical devices, aviation systems, communication systems, and telephony. In the evolution of embedded systems, FPGAs (Field-Programmable Gate Arrays) and SoCs (System-on-Chip) have an essential role because they provide flexibility and reconfigurability to adapt to several applications [98].

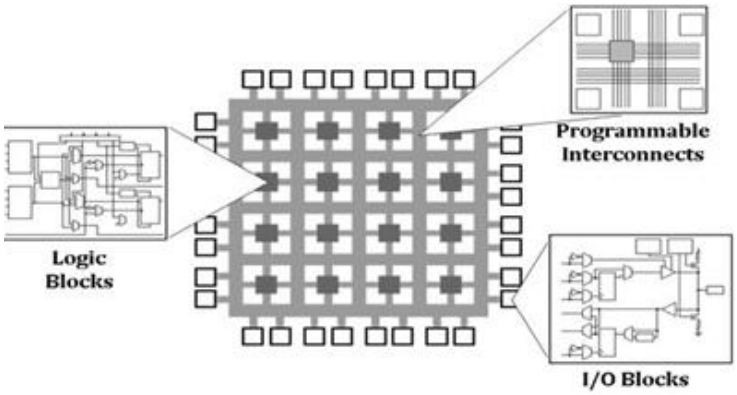


Figure 2.3: Basic Architecture of an FPGA [51].

2.6.1 Programmable Reconfigurable Devices

A Field Programmable Gate Array (FPGA) contains a series of resources such as logic blocks, which include logic gates and elements such as LookUp Tables (LUTs) and Flip-Flops, allowing the implementation of complex logic functions. Additionally, FPGAs include Block RAM (BRAM) to store data temporarily, as well as Digital Signal Processors (DSPs) intended to perform advanced mathematical operations, such as multiplication and addition in parallel, making them particularly useful for signal processing and digital filtering applications. Another essential resource in an FPGA is the input/output (I/O) pins, which are configured to interact with external devices and other system components. Its connection and functionality can be programmed using a hardware description language. The main feature of these devices is their ability to be reprogrammed to perform a specific function, which helps in the development of electronic devices because the developer can design, refine, and implement solutions in hardware without the need to use an application-specific integrated circuit (ASIC). A basic schematic is shown in Figure 2.3.

2.6.2 System-on-Chip (SoC)

With the impact caused by FPGA, the actual trend is to add components of FPGA architecture with a microprocessor CPU in a unique device called System-on-Chip

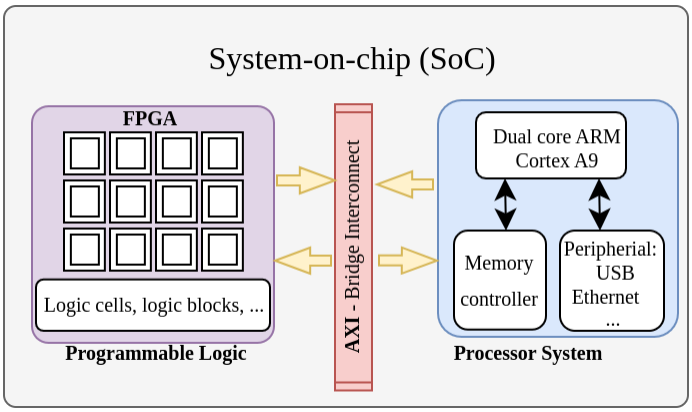


Figure 2.4: Basic architecture of Xilinx Zynq-7000 SoC.

(SoC). In cooperation with separate devices, a SoC provides less energy consumption, better time communication, and space-saving on the printed circuit board [3, 80]. That trend has motivated Xilinx to develop the family of SoC Zynq-7000, which combines an ARM Cortex-A9 processor with an FPGA from the families Artix or Kintex [25].

The architecture of these devices is complemented by the standard interface AXI, which provides ample bandwidth and a low-latency connection between the two principal parts of the SoC. This connection type implies that the designer can use the processor and the programmable logic from FPGA without creating different interfaces for the two devices physically separated. Figure 2.4 shows a basic architecture of the SoC.

2.6.3 Hardware description language

A Hardware Description Language (HDL) is a type of programming language specifically used for hardware design and development. This language enables a formal and precise description of an electronic circuit, allowing for automated analysis and simulation. It also facilitates the synthesis of an HDL description into a netlist, specifying how physical electronic components are interconnected. This netlist can be placed and routed to generate masks used to create an integrated circuit.

One of the prominent languages is VHDL, which is able to describe various circuit

types such as ASICs, program FPGAs, or PLDs, among others. Another relevant language is Verilog, which has a syntax inspired by the C programming language, making it familiar to engineers. Xilinx Vitis HLS, a High-Level Synthesis (HLS) tool developed by Xilinx, is another form to describe hardware. With HLS, users can create complex algorithms based on FPGAs using C/C++ code, which supports complex data types and mathematical functions. It also integrates with AXI4-Stream, facilitating data exchange with other IPs.

2.7 Materials and methods

The Hw/Sw co-design was implemented using Vitis HLS [106], Vivado [8], and PYNQ [88] tools and frameworks.

2.7.1 Vitis HLS

Vitis HLS is a high-level synthesis tool that converts functions written in a high-level programming language into Register-Transfer Level (RTL) designs. These RTL designs represent the corresponding hardware implementation and are used to develop Intellectual Property (IP) modules. These RTL IP modules encapsulate a desired function in a hardware design. Once synthesized, the RTL IP can be used in conjunction with other hardware and software components within the design flow to develop complex and complete systems.

The Vitis HLS tool automates many of the modifications necessary to optimize and implement C/C++ code in programmable logic, thus generating low latency and high performance in the algorithm. Some of the fundamental tasks of this tool include inferring the pragmas necessary to implement the correct type of interface according to the arguments of the function to be synthesized and creating pipelines between loops and functions. In addition, it allows the code to be customized and implemented according to different interface standards (such as AXI4) or other specifications.

The Vitis HLS design flow involves several key steps: First, the desired algorithm is compiled, simulated, and debugged in C or C++, generating a main function for

synthesis and a TestBench for operation verification. Next, the generated reports are reviewed to analyze and optimize the design using pragmas and parallelization directives. The algorithm is then synthesized into an RTL design or an HDL file using the Vitis HLS tool. After synthesis, the RTL design's implementation is verified through RTL co-simulation with the TestBench. Following verification, the clock frequency for the created IP is configured. Finally, the IP implementation is packaged into a compiled object file and exported as an RTL IP.

2.7.2 Vivado

Vivado is a set of software tools for the design of hardware solutions on Xilinx FPGAs under different hardware description standards such as VHDL, Verilog, and SystemVerilog, whose main components are:

- **IP integrator**: Allows the creation of complex hardware systems by instantiating and interconnecting IP cores and modules from the Vivado catalog within a design panel.
- **IP packager**: Tool for creating plug-and-play IPs to extend the Vivado IP catalog.
- **Vivado Block Design**: Graphic design tool that allows users to build hardware systems visually by interconnecting functional blocks. Among its functions are:
  - Includes IP and documentation required to design systems that utilize the full capabilities of Zynq-7000 SoC devices.
  - Users can customize and configure the functional blocks according to their specific requirements
  - To analyze and optimize system performance, such as estimating latency and resource consumption.
  - Connect AXI ports between components.

Once the system design is complete, the synthesis process is performed. During synthesis, Vivado converts the design into a hardware implementation. Then, Vivado can generate the files (*.bit, *tcl) that are essential to program the FPGA and define its functionality and internal configuration, including the layout of logic gates and

interconnects.

- "*.bit": contains the bitstream, which is the specific configuration of the FPGA generated by the hardware synthesis and design using Vivado.
- "*.tcl": contains a script that performs various tasks, such as setting up the design environment, generating output files, running simulations, and automating design workflows.

2.7.3 PYNQ

The PYNQ framework is an open-source Linux-based system using the Petalinux operating system designed specifically for embedded systems or SoCs. It provides a development environment that allows users to use both the CPU and FPGA easily. PYNQ runs on an integrated circuit board (FPGA) from Xilinx's Zynq family, which integrates ARM processors with programmable logic.

The PYNQ framework allows access to the programmable logic using different languages (such as C/C++ or Python), which simplifies the design and implementation of high-performance embedded systems. Under this environment, the flexibility and rapid prototyping of FPGA can be exploited to accelerate computationally intensive algorithms and signal processing in real time while taking advantage of the familiarity and wide range of libraries available for software development. This makes PYNQ especially suitable for various applications, ranging from learning computer science to rapidly prototyping complex embedded systems.

The workflow in PYNQ is as follows: First, load the Linux image through the SD card. Next, place the necessary files generated by Vivado (.bit and .tcl). PYNQ makes it possible to create scripts to upload the IP files. These files are used for programming the FPGA through the "*.bit" file, and the write and read memory addresses are accessed through the "*.tcl" file. This functionality allows the creation of a control function for the operation of the IP.

2.8 Summary

This chapter reviewed the fundamental concepts employed throughout this thesis. The structures used for security in V2V communications were shown, such as digital signatures and digital certificates. However, due to the capability of quantum computing, the security of current digital signature schemes in V2V communications will become vulnerable. Therefore, the fundamentals of post-quantum cryptography were presented based on problems for which there is no efficient algorithm to solve them on quantum computers. The NIST initiated a standardization process that led to the adoption of three digital signature algorithms: Dilithium, Falcon, and Sphincs.

The chapter also included a discussion on the security in VANETs, where concepts such as vehicle-to-vehicle (V2V) communication, the DSRC standard responsible for the maximum payload that can be used in V2V, as well as the IEEE 1609.2 standard, which mentions ECDSA based on the discrete logarithm problem over elliptic curves, as the cryptographic mechanism to be used for V2V message authentication.

Finally, the concept of Hw/Sw co-design was discussed as a way to accelerate the most demanding tasks in various applications by taking advantage of the flexibility provided by the software and the high performance provided by the hardware. Additionally, the rise of using SoC (System on Chip) to implement and validate such co-designs was discussed, as these devices have programmable logic and a processor system on a single device, reducing costs, energy, and delays. Finally, the chapter described the frameworks used for the Hw/Sw co-design, such as Vitis HLS, Vivado, and PYNQ.

CHAPTER 3

State-of-the-art

This chapter presents a review of related works to the problem of providing quantum resistance to V2V communications. First, works aiming at replacing the actual standard in communications V2V are discussed. Then, a discussion is provided about works that propose hybrid schemes to overcome the transition stage to post-quantum cryptography. Finally, the chapter reviews works that pursue to speed up the three finalists in the NIST PQC standardization process.

Table 3.1 shows the notation used in this chapter and for the rest of this thesis.

Name	**Meaning**
U	A user
S_U	U's signature scheme
pk_U^c	U's classic public key
sk_U^c	U's classic private key
C_U	U's hybrid certificate
C_U^C	U's classic certificate
C_U^{pq}	U's post-quantum certificate
$S_U^c.Sign()$	Function that signs using classic algorithm by U
$S_U^c.Vrfy()$	Function that verifies a classic signature generated by U
I_U	Metadata of U
$C_U^C.getSignature()$	Function that obtains the signature of C_U^c
$C_U^{pq}.getSignature()$	Function that obtains the signature of C_U^{pq}
CA	Authority certificate

S_{CA}	CA's signature scheme
pk^c_{CA}	CA's classic public key
pk^{pq}_{CA}	CA's post-quantum public key
sk^c_{CA}	CA's classic private key
sk^{pq}_{CA}	CA's post-quantum private key
$S^c_{CA}.Sign()$	Function that signs using classic algorithm by CA
$S^{pq}_{CA}.Sign()$	Function that signs using PQC algorithm by CA
$S^c_{CA}.Vrfy()$	Function that verifies a classic signature generated by CA
$S^{pq}_{CA}.Vrfy()$	Function that verifies a PQC signature generated by CA
α	Parameter that indicates the number of partitions
BSM	Basic safety message information
$CCons_\alpha()$	Function that rebuilds the α's parts of the certificate
$CFrag_\alpha()$	Function that partitions a certificate into i parts
C_i	i-th partition of the certificate
$H()$	Hash function
sig^c	Classic signature
sig^c_i	Classic signature of the i-th BSM_i
$SPDU$	Safe Protocol Data Unit

Table 3.1: Notation Table.

3.1 A Substitute for IEEE 1609.2

Mukherjee et al. [81] present LB-CPPA, a Conditional Privacy-Preserving Authentication (CPPA) lattice scheme based on the SIS problem. This scheme offers security and privacy, such as V2V as V2I communications. LB-CPPA consists of three principal stages: *system initialization*, in which the Trust Authority (TA) generates and shares the parameters between the entities; *signature generation*, in which the anonymity nodes (for example, vehicles) sign messages; $verification$, in which the RSU and the vehicles verify the input messages. The security of LB-CPPA underwent analysis within a random oracle model to demonstrate its resistance against quantum computers.

Salman and Blankinship [91] describe an implementation aimed at two key goals: ensuring the resilience of Public Key Infrastructure (PKI) against quantum computing threats and minimizing energy consumption. This approach utilizes cryptographic keys specifically designed to enhance security and efficiency in V2V and V2I communications within VANETs, developed by Crash Avoidance Metrics Partners (CAMP) in collaboration with the United States Department of Transportation (USDOT). The system architecture consists of three primary components: the end device, the Registration Authority (RA), and the Certification Authority (CA). The process begins with vehicles sending a Certificate Signing Request (CSR) containing seed keys to the RA, which then forwards the request to the CA for signing. The implementation employs the Integrated Elliptic Curve Encryption Scheme (ECIES). As a quantum-resistant alternative, the NTRU problem is also considered, although it has been observed to be slower than ECIES in experiments conducted on a Raspberry Pi.

Li et al. [71] propose a lattice digital signature scheme based on the RSIS problem with revocation dynamic for VANETs, which they probed in a random oracle model that in their proposal is unforgettable in a quantum computer. That scheme allows sign aggregation from multiple signers across different messages into a single signature and reduces the signature length and verification efficiency. It involves four entities: the Key Generation Center (PKG), responsible for key management service; the signer, who signs messages using their key; the aggregator, who collects signatures and messages to aggregate the signatures generated by signers; and the verifier, who only needs to verify the aggregate signature to validate all messages.

Dharminder and Mishra [28] present a practice scheme for Conditional Privacy-Preserving Authentication (LCCPA) based on lattices in the SIS problem for vehicular security. LCPPA improves the efficiency of communication and reduces the time of verification, guaranteeing identification and user authenticity. The network model involves four entities: trusted third party (TT), RSU verifies the authenticity of messages, Authentication Server (ATS) provides the resources, and vehicle detects tampers of the messages. LCCPA is similar to Diffie-Hellman (DH), a key establishment protocol between two parties, with a slight storage cost but a similar communication cost.

Kim and Seo [61] propose a method to apply post-quantum cryptography algorithms, specifically Crystals-Dilithium, in the V2V communication in the application layer. It introduced a "simple split path" and a method split package to apply the big signatures in V2V communication. The experimental results show a smaller latency than traditional methods.

In the state-of-the-art, there are solutions to the long periods for the substitution or modification of the actual standard to add post-quantum cryptography either with NIST recommendation for standard or another scheme; some proposals only show that it is quantum resistant; nevertheless, some leave aside metrics such as time and storage cost.

Twardokus et al. [100] discuss the viability of the post-quantum algorithms: Falcon, Dilithium, and Sphincs$^+$ in V2V communications. They show the use of a hybrid scheme, i.e., the use of the actual standard ECDSA and a post-quantum algorithm, for a transition stage, nevertheless is not feasible because the big size of payload caused by the use of both signatures from that scheme and does not stick to requirements of the V2V communication protocols. Considering this, they propose a partially hybrid scheme; its operation is described in the following subsection. However, it even shows a disadvantage in the number of cars that can send and verify messages.

3.2 Transition to post-quantum cryptography

The transition to post-quantum cryptography has been recommended in several forums and works [6, 54, 58, 107]. Suppose the time remaining until large-scale quantum computers are available for cryptanalysis is less than the sum of the lifetime of the data needing protection and the time required to transition to post-quantum cryptography. In that case, the data will be at risk. A clear example of this risk is the store-now-decrypt-later (SNDL) [78] attack against conventional encryption algorithms, where an adversary stores text encryption to decrypt it later using a quantum computer.

However, while it is vital to advance the transition to post-quantum cryptographic

algorithms, it is also crucial to adopt a conservative and cautious stance. A hybrid strategy has been suggested to address this issue; this scheme is recommended to facilitate the transition to the post-quantum stage [75]. It allows security to be maintained during the implementation of new algorithms since the security of the hybrid scheme depends on at least one of the underlying components remaining secure [65]. From this perspective, the hybrid strategy offers an effective solution when the security of a new primitive is not yet established, but that of an old primitive is already in doubt.

3.2.1 Protocol for verification in V2V communications

Explaining the current security protocol used in V2V communications is necessary for a clear interpretation of the hybrid scheme.

Let $\mathcal{M}$ denote the message space corresponding to BSMs and $\mathcal{U}$ denote the set of pseudonyms, a user $U \in \mathcal{U}$, the certification authority CA, and $BSM \in \mathcal{M}$. Let S_U^c and S_{CA}^c be the signature schemes using classic algorithms for a user U and a certification authority CA, respectively, as defined in subsection 2.2. The protocol for verification in V2V $\mathcal{P}$ communications [100] is defined as follows:

The process begins with the "Certificate request phase", where the user requests a certificate from the Certificate Authority (CA). The CA loads its keys (pk_{CA}^c, sk_{CA}^c) and generates the user's certificate C_U^c, which is then transmitted to the user. In the "SPDU generation phase", the sender loads their own keys (pk_U^c, sk_U^c), generates a Basic Safety Message (BSM), and signs it to obtain the signature sig^c. The resulting SPDU consists of the user's certificate C_U^c, the BSM, and the signature sig^c, which is transmitted to the receiver. Finally, in the "SPDU verification phase", the receiver verifies the user's certificate C_U^c and the signature sig^c. If both verifications are successful, the receiver proceeds to process the BSM.

Figure 3.1 illustrates a model of the current V2V security protocol. Figure 3.2 presents description in the form of an interaction diagram 2.5.

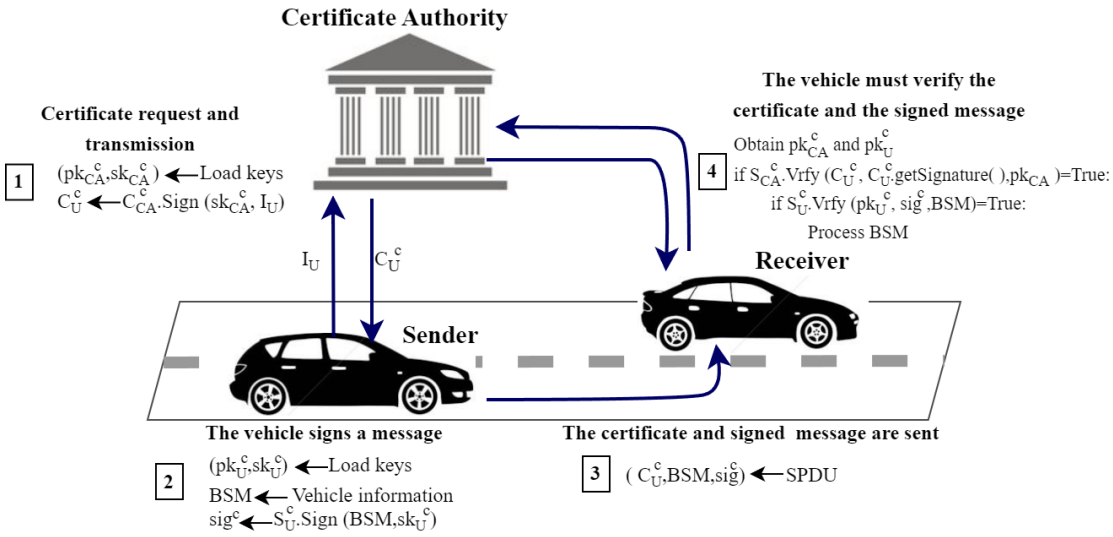


Figure 3.1: Current IEEE 1609.2 V2V security protocol.

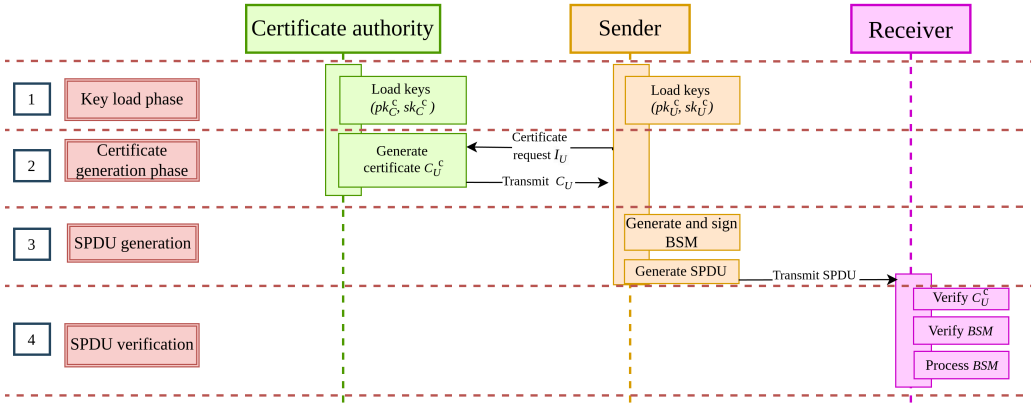


Figure 3.2: Interaction diagram of the current V2V protocol.

3.2.2 Hybrid scheme in V2V communication

A hybrid cryptographic scheme combines different cryptography algorithms in a scheme [7]. In this context, the secret key is a concatenation of the secret keys of the algorithms that make up the hybrid scheme, and the public key is a combination of the public keys corresponding to those secret keys. In this thesis, a hybrid scheme is the combination of a traditional scheme with a post-quantum scheme.

3.2.2.1 Post-Quantum hybrid scheme for V2V communications

The hybrid scheme in the context of V2V communication is shown below. Here, the ECDSA scheme, as specified by the IEEE 1609.2 standard, is combined with a post-quantum digital signature scheme, such as one of the three NIST finalists. The process begins with a certificate authority issuing certificates for the sender user, which are signed using both ECDSA and a post-quantum signature scheme. Each BSM is then signed with both ECDSA and post-quantum signatures. Upon receipt, the receiver verifies the hybrid certificate using the public keys of the authority (one for ECDSA and one for the post-quantum scheme). After verifying the certificates, the BSM is authenticated using the public keys of the sending user, with one key for ECDSA and one for the post-quantum scheme. Nevertheless, using the hybrid scheme is impractical in the transition to the post-quantum stage due to the size of the signatures that overtake the requirements of the V2V domain. A viable solution, as demonstrated by Twardokus et al. [100], is to implement a partially post-quantum hybrid scheme for V2V authentication.

3.2.2.2 Partially post-quantum hybrid scheme for V2V communicaions

This design is efficient, has a small payload, and maintains backward compatibility. The scheme consists of a hybrid certificate that combines a quantum-resistant digital signature scheme and the current scheme (ECDSA) within the certificate, maintaining the basic security message (BSM) signature solely with ECDSA. This dual-signature approach guarantees that the ECDSA verification key remains unaltered and originates from a legitimate certificate authority. However, the partially

hybrid scheme addresses two main attack vectors:

1. A quantum adversary could potentially forge a signature on a BSM. This would require creating the forgery within the certificate's validity period, which is shortened as ECDSA becomes vulnerable over time [100]. The period of vulnerability can be managed through updates, and during the transition phase, quantum adversaries are no more powerful than classical counterparts in this type of attack.
2. A quantum adversary might attempt to produce a fraudulent certificate issued by a certificate authority (CA). To mitigate this risk, the system incorporates a second signature using a post-quantum algorithm, resulting in a hybrid certificate.

To provide a detailed understanding of this partially hybrid design, the following functions are defined:

- $C_U = (C_U^c || C_U^{pq})$, the hybrid certificate is defined differently in each design, where C_U^c is the classical certificate, C_U^{pq} is the post-quantum certificate, and $\|$ is the concatenation operator of these two components.
- $CFrag_\alpha : C_U \rightarrow \{C_1, \ldots, C_\alpha\}$ is the set of the hybrid certificate partitions into α equal parts (i.e., $C_1 = (C_U^c || C_{frac})$, where C_{frac} is a fraction of C_U^{pq}). The choice of α depends on computing resources demanded by the PQC algorithm that meet the IEEE 1609.2 requirements in terms of time and space.
- $CCons_\alpha : \{C_1, \ldots, C_\alpha\} \rightarrow C_U$ reconstructs a hybrid certificate.
- $H : C_U \rightarrow h^c$, where H is a hash function, $h^c \in \{0,1\}^{256}$.

The parameter α is optimized based on the design and the post-quantum algorithm to ensure that all BSMs can be signed at least with ECDSA. The goal is to minimize α while maintaining post-quantum security for the maximum number of BSMs possible. Additionally, all frames must have a maximum size of 2,304 bytes to comply with current DSRC protocols, and all frames transmitting C_U are required to be of equal size.

Another aspect to consider for the proposed partially hybrid post-quantum scheme is the five-message cycle, which helps with payload reduction.

Similar to the current protocol, let $\mathcal{M}$ denote the message space corresponding to

BSMs, and let $\mathcal{U}$ denote the set of pseudonyms, with a user $U \in \mathcal{U}$, the certification authority (CA), and $BSM \in \mathcal{M}$. Let S_U^c be the classical digital signature scheme for a user U, and S_{CA}^c, S_{CA}^{pq} be the classical and PQC digital signature schemes for the CA, respectively. The partial hybrid scheme works as follows:

First, during the "Key load phase", the CA loads its keys $(pk_{CA}^c, sk_{CA}^c, pk_{CA}^{pq}, sk_{CA}^{pq})$. The sender does the same. The sender only uses conventional cryptography using its keys (pk_U^c, sk_U^c). Next, in the "Certificate generation phase", the sender requests the certificate from the CA, which then generates a hybrid certificate (C_U) consisting of the concatenation of two certificates: one signed with ECDSA (C_U^c) and the other with a PQC (C_U^{pq}) algorithm. The CA then transmits the certificate to the sender to transmit the SPDUs. During the "Pre-α phase", for each BSM_i where $i = 1, ..., \alpha - 1$, the $SPDU_i$ consists of a partition of the hybrid certificate (C_i), the BSM_i, and its signature (sig_i^c). In this phase, the receiving vehicle delays processing of BSM_i until BSM_α is sent, which includes the final part of the post-quantum certificate, the BSM_α, and the signature sig_α. The "In-α phase", the receiving vehicle reconstructs the certificate (C_U). As this is a scheme designed for the transition phase, it must remain compatible with the current protocol. For vehicles capable of verifying PQC signatures, the post-quantum signature of the certificate should be verified. If the verification is successful, or if the vehicle does not support post-quantum verification, the receiving vehicle verifies the ECDSA-signed certificate and the BSM signature. Once verified, it proceeds with the processing of BSM_i. During the "Post-α phase", for each BSM_i where $i = \alpha + 1, ..., 5$, the $SPDU_i$ consists of the corresponding signature sig_i and the hash of the complete certificate. Upon receiving $SPDU_i$, the receiving vehicle verifies the hash of the complete certificate and the signature, then proceeds to process the BSM_i. Figure 3.3 shows an interaction diagram of the partially hybrid post-quantum scheme previously described.

The cycle described in the previous paragraph repeats every five $BSMs$, while certificates are periodically updated; for example, certificates are updated every five minutes with a validity period of one week.

This scheme ensures that the payload size requirement for each SPDU is satisfied, as shown in Table 3.3. Both the choice of α and the SPDU size calculations for each scheme were based on the recommendations in [100, 60]. The ECDSA certificate

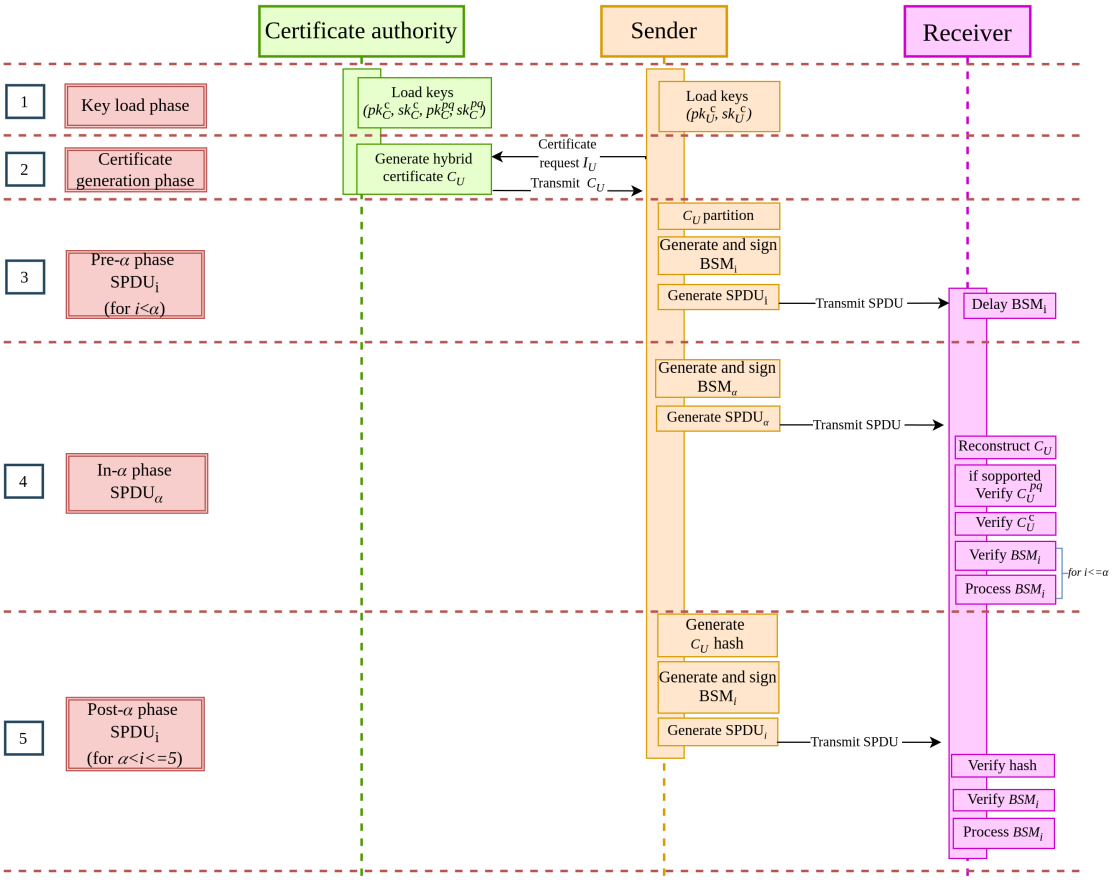


Figure 3.3: Interaction diagram for a partially hybrid post-quantum scheme for quantum-resistant authentication in V2V communications.

Scheme	**NIST Security Level**	**Signature Size**	**Public Key Size**
Falcon 512	1	666	897
Dilithium II	1	2420	1312
Sphincs$^+$ 128	1	7856	32

Table 3.2: Comparison of cryptographic post-quantum algorithms.

Scheme	$\|C_U\|$	$\|C_i\|$	α	$\|SPDU_1\|$	$\|SPDU_2\|$	$\|SPDU_3\|$	$\|SPDU_4\|$	$\|SPDU_5\|$
Falcon 512	828	828	1	955	159	159	159	159
Dilithium II	2582	1291	2	1418	1418	159	159	159
Sphincs$^+$ 128	8018	2005	4	2132	2132	2132	2132	159

Table 3.3: Size (bytes) for certificates and SPDUs in PQC schemes for the partial hybrid scheme.

size is 162 bytes, while the total certificate size, denoted by $|C_U|$, comprises both the ECDSA certificate and a post-quantum signature. The size of a partition of C_U, represented by $|C_i|$, is based on α. The size of the ECDSA signature generated by the transmitting user is denoted as $|sig|$. The *BSM* size is approximately 39 bytes, while the hash size, using the SHA-256 algorithm, is 32 bytes. $|SPDU_i|$ corresponds to the size of $SPDU_i$ within the five-message cycle of the partial hybrid PQC scheme. Specifically, $|SPDU_i| = |BSM| + |C_i| + |sig| + 24$ when $i \in 1, ..., \alpha$, and $|SPDU_i| = |BSM| + |hash| + |sig| + 24$ when $i \in \alpha + 1, ..., 5$. Furthermore, the sizes of the three post-quantum algorithms are considered, with a security level of 128-bits corresponding to NIST level 1, as used in the partially hybrid scheme. In security terms, level 1 is equivalent to the protection provided by 128-bit AES (Advanced Encryption Standard), which means there are 2^{128} possible key combinations for encryption and decryption [104]; level 3 corresponds to the security of 192-bit AES, and level 5 corresponds to the security of 256-bit AES.

In the comparison of the presented sizes, it is also necessary to consider the additional size of the headers for practical application within the certificates. These headers include data such as the expiration date, metadata of the transmitting user, and crucial information for verification and decoding, totaling between 40 and 30 bytes [100].

3.3 Acceleration of PQC Digital Signature Algorithms

As mentioned, post-quantum digital algorithms are slower than the actual algorithms. Implementing these algorithms in hardware can accelerate the execution time. Schemes based on lattices have shown be more efficient in devices with limited resources than other reliable math to quantum computers (compared with the hash function used in Sphincs+) because they are fast and have small signature sizes [72].

Wang et al. [102] mention that Dilithium has favorable characteristics for full implementation in hardware. Nevertheless, Falcon has greater difficulty in hardware implementation due to float-point arithmetic. These operations require floating-point FFT operations, which are costly to implement in hardware. On the other hand, verification is much simpler, involving integer polynomial operations and using the Shake 256 hash function based on Keccak, a part of SHA-3. Authors present an implementation of a Hw/Sw co-design for the Dilithium scheme. They use an SoC where the most costly part is calculated by hardware, which includes polynomial multiplication and the computing of the hash. They do design a hardware implementation of NTT to accelerate the polynomial multiplication, which is coherent with Zhou et al. [108] that mention that the polynomial multiplication and SHA-3 function is the most cost computationally intensive function of Dilithium. They implement a co-design of Dilithium to accelerate the polynomial multiplication. Gaoyu et al. [36] present a co-design implementation to accelerate the polynomial operation and hash function SHAKE for the key generation, signature generation, and verification. In addition, full implementations for Dilithium have been reported, for example, Beckwith et al. [13] and Land et al. [66].

Schmid et al. [92] describe a full hardware implementation of Falcon using High-Level Synthesis (HLS). Their code is based on the NIST implementation written in C and modified to accomplish with HLS. Karabulut and Aysu [56] presented a co-design for Falcon that accelerates the discrete Gaussian sampling used during key generation. They propose a co-design of this sampling to handle flexible calculations enabling the adjustment of variable parameters in software and the execution of fun-

damental operations with low latency using parameterized and customized hardware.

Amiet et al. [9] present a hardware accelerator for Sphincs$^+$ based in SHAKE256. They mainly focus on fast signing with moderate use of FPGA resources. Their accelerator is capable of signing, key generation, and signature verification. Authors mentioned that their accelerator, compared with other hardware implementations of the classical signature scheme RSA and ECDSA, is highly competitive in signature and verification time terms. Dai et al. [26] propose four hardware architectures for Haraka in SPHINCS+. The different cases combine several optimization methods for different application cases. Berthet et al. [16] proposed a full area-efficient FPGA implementation of Sphincs$^+$ based SHA-256. This optimization of resources allows use on low-power IoT devices.

Soni et al. [94] present a methodology to implement and evaluate post-quantum cryptography (PQC) algorithms in hardware, including the three finalists of the NIST post-quantum algorithm standardization. Given the complexity of PQC algorithms, it is challenging to develop optimized RTL designs in Verilog/VHDL in a short time. Authors present a full implementation for Sphincs$^+$, Dilithium, and an implementation of verification of Falcon. A HLS methodology is adopted from the available C specifications.

Table 3.4 presents a summary of the state-of-the-art hardware acceleration of PQC digital algorithms. This table shows the accelerated algorithm, the accelerated part, the platform used for its evaluation, the hardware description language employed (if mentioned), and the acceleration obtained. In the case of a SoC, the comparison is made with the processor within the SoC, and in the case of an FPGA, the comparison is as mentioned by the authors.

Reference	**Algorithm**	**Aceleration part**	**Platform**	**Lenguaje description**	**Speed - up**
[102]	Dilithium	Polynomial multiplication, hashing, and sampling for signing and verification	SoC: Xilinx Zynq-7000	Verilog	17× for signing and 40x for verification
[108]	Dilithium	NTT implementation for polynomial addition and multiplication for signing and verification	SoC: ARM Cortex-A9 of Zynq-7020	Verilog	11.2× for signing and 7.4x for verification
[13]	Dilithium	Full implementation	Virtex Ultrascale+	Verilog	2961× for key generation, 1071× for signing, and 2042× for verification. Compared to Cortex-M4.
[36]	Dilithium	NTT/INTT for polynomial operation and SHAKE.	SoC: ARM Cortex-A9 of Zynq-7000	Verilog	8.7x for key generation, 6.3x for signature, and 9.1 for verification.
[66]	Dilithium	Full implementation	Artix-7		592× for key generation, 300× for signing, and 510× for verification. Compared to Cortex-M4.
[9]	Sphincs$^+$	Full implementation	Artix-7	Verilog-VHDL	10X for signing, 6.3x for verification. Compared to desktop computer CPU.

[26]	Sphincs$^+$	Haraka	ASIC		7.79×. Compared to CortexA72
[16]	Sphincs$^+$	Full implementation	Xilinx XZU3EG FPGA		The FPGA implementation is in the same order of magnitude in terms of speed as the desktop computer CPU.
[56]	Falcon	Discrete Gaussian sampling for key generation	SoC: ARM Cortex-A9 of Zynq-7000		2.7x for Verification
[92]	Falcon	Full implementation	Zynq UltraScale+ (ZCU104) FPGA	HLS	9.8x for key generation, 30x signing, and 4.8x for verification. Compared to Cortex-M4

Table 3.4: Hardware acceleration of PQC digital algorithms.

3.4 Summary

This chapter reviewed the state of the art in verification in V2V communications. There are proposals for replacing the current standard for V2V communications in the quantum stage. Additionally, for a transition to this stage, hybrid schemes are suggested, i.e., using the current cryptographic standard (ECDSA) and a post-quantum scheme. However, this approach is impractical due to the amount of payload used by both schemes, which violates the payload requested by the DSRC standard. Therefore, partially hybrid schemes have been suggested. The main difference from the current protocol is that the BSM signatures are maintained with ECDSA, and the certificate is signed with both ECDSA and a post-quantum algorithm. Additionally, by using industry standards such as a five-message cycle, it is possible to divide the certificate into partitions and transmit each chunk to the receiver, who reconstructs the certificate. The receiver can then proceeds to verify both signatures on the

certificate and continue with the current protocol using the five-message cycle, i.e., verifying a hash associated with the certificate and the BSM. Therefore, the use of this type of scheme is the best suggestion to address the issue of a quantum-safe protocol in the transition stage.

Given that this work aims to use hardware, potential implementations must be feasible. Only the verification of the post-quantum scheme is used within the state-of-the-art in partially hybrid schemes. Thus, in an Hw/Sw co-design, the verifications of these schemes must be feasible to implement in hardware. In the state of the art, works related to hardware implementations of signature verification for the three post-quantum schemes were discussed. This suggests that, at the implementation level, any algorithm can be chosen for this partially hybrid scheme protocol in V2V communications.

CHAPTER 4

Hw/Sw co-design for quantum-safe authentication in V2V communications

As mentioned in Chapter 1, two challenges make it difficult to adopt a pure post-quantum approach in the transition stage to post-quantum security due to the constrains implied by the actual standards. The first one is to meet the payload limit imposed by the communication protocol (DSRC). The second is to meet performance requirements, specifically to verify 100 vehicles in a 100 ms time frame [100].

This chapter presents the details of a novel Hw/Sw co-design well suited for enabling authentication in V2V communications during the transition stage to post-quantum resistance. This co-design supports the execution of a partial hybrid scheme that uses both classic and PQC digital signatures in the authentication process of messages exchanged in V2V communications according to the IEEE standard. While most of the partial hybrid scheme is executed in a general-purpose processor, hardware is used to accelerate PQC signature verification, which results in the most time consuming operation in the whole protocol.

4.1 Partially hybrid scheme

In the transitional stage, it is recommended to use a hybrid scheme in the V2V communications domain, involving both ECDSA and a post-quantum cryptography digital signature scheme. However, due to payload constraints, a hybrid scheme is

not feasible because a post-quantum signature, which is considerably larger than classical schemes, is added to an ECDSA signature, which significantly increases the payload.

In section 3.2.2, a partially hybrid solution to this problem, documented in the literature, was presented. Since this work focuses on a hardware and software co-design, and it is essential to identify which tasks require acceleration, the protocol is described at the operational level in Figure 4.1, using the notation defined in Table 3.1.

First, the "Pre-α phase" (lines 1-9) is outlined. During this phase, the transmitting vehicle U uploads the C_U, performs the partitioning of C_U, generates BSM_i, and signs to create the corresponding SPDU_i (line 8), which is transmitted to the receiving vehicle R. The latter maintains the delay (line 9) due to the impossibility of reconstructing C_U. This process repeats for $i = 1, ..., \alpha - 1$.

The "$In - \alpha$ phase" (lines 10-24) allows U to generate SPDU_α, which includes a partition C_i, BSM_i, and the respective signature (lines 10-12), which is then transmitted to R. In this phase, C_U can be verified, allowing for its reconstruction (line 13). Next, it is checked whether the vehicle can verify PQC signatures (line 15); if the verification is successful or if the vehicle lacks this capability, the ECDSA-signed certificate is verified (line 19). Subsequently, R can verify the signatures on BSM_i for $i = 1, ..., \alpha$, that is, all previous BSMs that were delayed due to the inability to verify them. Then, R can process each BSM (lines 20-23).

Finally, in the "Post-α phase" (lines 25-34), U creates SPDU_i generating and signing BSM_i for $i = \alpha + 1, ..., 5$, along with the hash of C_U. Upon transmitting this SPDU, the hash is verified (line 29), and if it is correct, BSM_i is verified (line 32) and then processed.

For the Hw/Sw co-design, the three finalist schemes for post-quantum standardization were considered. A main task for designing the Hw/Sw architecture was determining which tasks can be executed efficiently in software and which create bottlenecks and need to be accelerated in hardware. Starting from the previous hybrid scheme, the design of the proposed Hw/Sw architecture was done, considering the following stages:

- The selection of the most convenient post-quantum scheme according to the

	Sender Vehicle U		Receiver Vehicle R
1	$(C_U^{pq} \| C_U^C) \leftarrow C_U$		
2	$Pk_U \leftarrow$ Public key		
3	$Sk_U \leftarrow$ Private key		
4	$\{C\}_{i=1,...,\alpha} \leftarrow CFrag_\alpha(C_U)$		
5	if $i = 1, ..., \alpha - 1$		
6	Create BSM_i		
7	$sig_i \leftarrow Sign^c(BSM_i, Sk_U)$		
8	$SPDU_i \leftarrow (BSM_i, sig_i, C_i)$	$\xrightarrow{\text{Transmit } SPDU_i}$	$(BSM_i, sig_i, C_i) \leftarrow SPDU_i$
9			Delay BSM$_i$
10	Create BSM_α		
11	$sig_\alpha \leftarrow Sign^c(BSM_\alpha, Sk_U)$		
12	$SPDU\alpha \leftarrow (BSM_\alpha, sig_\alpha, C_\alpha)$	$\xrightarrow{\text{Transmit } SPDU_\alpha}$	$(BSM_\alpha, sig_\alpha, C_\alpha) \leftarrow SPDU_\alpha$
13			$C_U \leftarrow CCons_\alpha(C_1, ..., C_\alpha)$
14			$(C_U^{pq}, C_U^c) \leftarrow C_U$
15			if PQC sopported:
16			if $CVrfy^{pq}(Pk_C^{pq}, C_U^{pq})$=True:
17			continue
18			else: Abort
19			if $CVrfy^c(Pk_C^c, C_U^c)$=True:
20			for i=1,...,α
21			if $Vrfy^c(Pk_U^c, sig_i, BSM_i)$=True :
22			Process BSM$_i$
23			else: Abort
24			else: Abort
25	Create BSM_i		
26	$sig_i \leftarrow Sign^c(BSM_i, Sk_U)$		
27	$h \leftarrow H(C_U)$		
28	$SPDU_i \leftarrow (BSM_i, sig_i, h)$	$\xrightarrow{\text{Transmit } SPDU_i}$	$(BSM_i, sig_i, h) \leftarrow SPDU_i$
29			if $h == H(C_U)$:
32			if $Vrfy^c(Pk_U^c, sig_i, BSM_i)$=True:
31			Process BSM_i
32			else: Abort
33			else: Abort

Figure 4.1: Vehicle-to-Vehicle communication protocol using a partially hybrid scheme.

Scheme	Time	vs. ECDSA
ECDSA	1.26	1.00
Falcon 512	69.19	54.78
Dilithium II	14.59	11.55
Sphincs$^+$ 128	12331.91	9764.36

Table 4.1: Comparative time (ms) of PQC signature generation algorithms vs. ECDSA on Cortex A9.

payload size restrictions.

- The partitioning of tasks to be most conveniently implemented in hardware or software based on the target performance.

4.2 Post-quantum signature for the Hw/Sw co-design

First, the most suitable PQC digital signature algorithm for the hybrid V2V authentication scheme was selected. To achieve this, the focus was on analyzing the three PQC digital signature algorithms recommended by NIST.

There are three security levels among Dilithium implementations: Dilithium2 (level 1), Dilithium3 (level 3), and Dilithium5 (level 5). Falcon has two security levels: Falcon 512 (level 1) and Falcon 1024 (level 5). Sphincs$^+$ has three security levels: Sphincs$^+$ 128 (level 1), Sphincs$^+$_192(level 3), and Sphincs$^+$_256 (level 5). The IEEE 1609.2 standard recommends a security level of 128 bits, so the study only considers level 1 in the three algorithms [100].

Two metrics were primarily considered for an analysis of the three algorithms: performance and signature size.

Firstly, as a performance test, three PQC algorithms were implemented and evaluated under the same conditions. The libraries used were Botan [18] for ECDSA, liboqs [84] for Dilithium and Sphincs$^+$, and NIST implementation [92] for Falcon. Tables 4.1 and 4.2 show a performance comparison of all these schemes for signature and verification operations.

The lattice-based schemes, Dilithium and Falcon, offer advantages compared to

Scheme	**Time**	**vs. ECDSA**
ECDSA	0.10	1.00
Falcon 512	2.50	25.00
Dilithium II	3.41	34.10
Sphincs$^+$ 128	12.32	123.20

Table 4.2: Comparative time (ms) of PQC signature verification vs. ECDSA on Cortex A9.

PQC signature verification	$\nu_{\max}$
Falcon 512	37
Dilithium II	26
Sphincs$^+$ 128	7

Table 4.3: Performance comparison of PQC signature verification in the partially hybrid approach for V2V message authentication.

the hash-based scheme (Sphincs$^+$). Firstly, their efficiency is better in terms of time and resources, as they require fewer computational resources. So, they seem more suitable for applications where performance is a priority [67].

Table 4.3 shows a performance comparison with the maximum capacity of vehicles to be verified, for the three post-quantum digital signature schemes under the partially hybrid scheme. The column $\nu_{\max}$ shows the number of vehicles to be verified with each of the post-quantum schemes recommended by NIST. This number is dependent on the verification of the SPDU$_\alpha$, which is when the receiving vehicle acquires the complete certificate. In this SPDU, the verification of the hybrid certificate is carried out together with the verification of the $\alpha - 1$ BSMs previously received.

In terms of the signature size, lattice-based schemes have smaller public keys and signatures than hash-based schemes, as it can be seen in Table 4.4, where Sphincs$^+$ has a signature size approximately 122 times larger than ECDSA, six times larger than Dilithium, and nine times larger than Falcon. Falcon has the shortest signatures among the three PQC digital signature algorithms. This aspect is crucial in the context of V2V communications using a partially hybrid scheme. Table 3.3 shows that Falcon only requires one SPDU, so latency reduces for the receiver to acquire the entire hybrid certificate. In contrast, Dilithium requires two SPDUs, which causes a

Algorithm	**Public Key**	**Signature**
ECDSA	64	64
Falcon-512	897	666
Dilithium-II	1312	2420
Sphincs$^+$ 128	32	7856

Table 4.4: Comparison of cryptographic sizes (bytes) of digital signature schemes.

delay in the verification of a BSM. On the other hand, Sphincs$^+$ needs four SPDUs, which results in a delay of three BSMs before they can be verified. This consideration is crucial in domains where decisions must be made within milliseconds, such as in vehicular contexts. In terms of the length of the digital signature, Falcon is the best choice due to the absence of delay for BSM verification.

The above shows that Sphincs$^+$ has the lowest performance in both response time and delay time before being able to verify the hybrid certificate, as well as needing the largest payload to transmit the hybrid certificate, not taking advantage of the five-cycle messages feature, which aims to reduce the amount of payload to transmit. On the other hand, Falcon and Dilithium show the best performances in this domain. However, Falcon resulting being better than Dilithium due to the lack of delay since one SPDU is sufficient for the complete transmission of the hybrid certificate.

On the other hand, Dilithium stands out for its simplicity of full hardware implementation compared to Falcon; as discussed in the previous chapter, Falcon verification is also favorable for hardware implementation due to the absence of elements that cause difficulties in the entire scheme, such as Gaussian sampling, which is used in key and signature generation.

Table 4.5 provides a comparison between Falcon and Dilithium using the parameters originally proposed by their creators. Signature verification in Falcon and Dilithium use the same hash function; however, while Falcon uses a single polynomial, Dilithium uses arrays of polynomials, making Dilithium's operations more complex. That is why signature verification in Falcon has better running times. Due to its feasible hardware implementation, lower payload usage, and lack of certificate partitioning, Falcon provides a better combination of efficiency and simplicity

	Dilithium	**Falcon**
Security Basis	M-LWE, SIS	NTRU-SIS
Hash Function	SHA-3	SHA-3
Elements	Vectors/matrices of polynomials $(k, l) \in [(4, 4), (6, 5), (8, 7)]$	Polynomials
Polynomial Degree	256	$[512, 1024]$
Coefficient Modulus	$8,380,417$ (23-bit)	$12,289$ (14-bit)

Table 4.5: Comparison of Dilithium and Falcon parameters.

in the verification algorithm than the other two schemes. Therefore, Falcon represents the best option for verifying hybrid certificate signatures, and consequently, it results being the most suitable post-quantum algorithm for authentication in V2V communication when implemented as a Hw/Sw co-design.

4.3 Task Partitioning in Hardware and Software

Implementing a Hw/Sw co-design for message authentication in V2V communication requires deciding which tasks are implemented in software and which ones are accelerated in hardware. A careful partition will result in a better usage of computing resources, avoiding excessive load on hardware implementation and taking advantage of the flexibility of software.

At the sender side, ECDSA in software meets by far the requirement of signing one BSM within 100ms. Thus, ECDSA is not required to be accelerated in hardware.

With a pure software implementation, as shown in Table 4.3, the partially hybrid scheme would only be able to verify approximately 37 vehicles, which is significantly below the recommended number. The verification of the Falcon algorithm represents a significant bottleneck, consuming 92% of the total time required to verify a vehicle. Additionally, following the practice of using a five-message cycle, the verification of the hybrid certificate occurs only in the first transmitted message. Therefore, hardware acceleration for Falcon verification in the first message is necessary to improve performance in terms of time. During the remainder of the cycle, an ECDSA

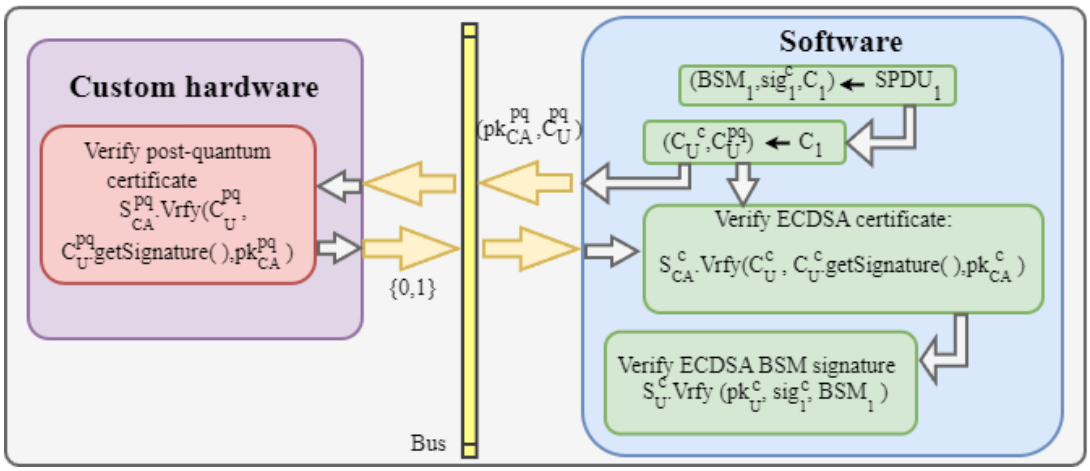


Figure 4.2: Diagram of the Hw/Sw partition for signature verification of $SPDU_1$.

signature associated with the SPDU is verified, and a 256-bit hash (e.g., SHA-256) related to the certificate is computed. On a Cortex A9, the execution time for SHA-256 is approximately 0.039 ms. Adding the time for ECDSA verification and hash calculation, the total operation takes around 0.139 ms. By repeating this process for the remaining four messages of the cycle, it would be possible to verify 100 messages in approximately 14 ms, which meets the time constraints. Therefore, on the receiver side, it is only necessary to accelerate the verification associated with the first SPDU_1 for the partially hybrid scheme using the post-quantum Falcon scheme, while other tasks can be efficiently executed in software.

It is intended to use both software and custom hardware that performs the Falcon verification. Once the post-quantum signature verification is complete, a verification value is returned for the software to proceed to verify the ECDSA-signed certificate (C^c_U) with the ECDSA public key associated with the certification authority (pk^c_{CA}) and finally, to verify the signature (sig^c) associated with the message (BSM) along with the ECDSA public key associated with the issuing user (pk^c_U). Then, the verification of SPDU_1 is completed. Figure 4.2 shows a diagram of the Hw/Sw architecture for SPDU_1 verification.

After verifying the first $SPDU$, verifying the rest of the four SPDUs within the

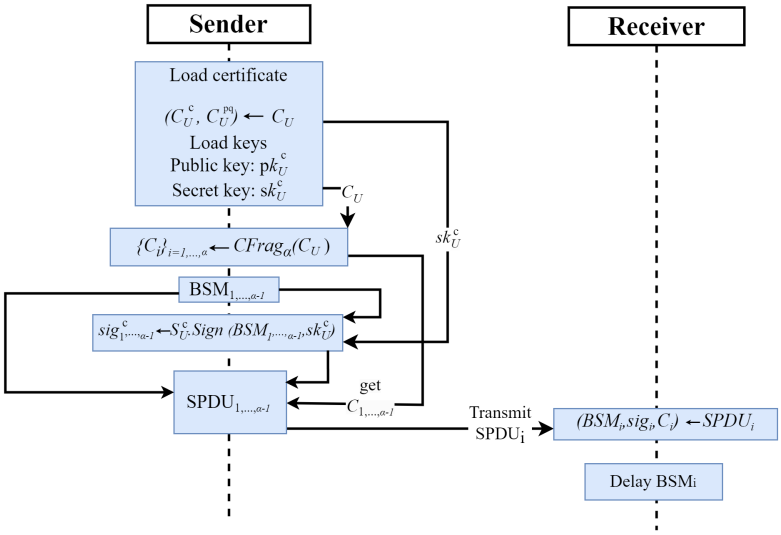
cycle of five messages consists of verifying a Hash and a signature associated with the respective BSM. As mentioned above, this can be executed in software.

Figure 4.3 presents the proposed Hw/Sw co-design for a partially hybrid scheme in V2V communication within a five-message cycle, depicted as a block diagram at the operational level. This figure highlights which operations require acceleration and which do not, according to the discussion presented in the previous sections. The figure is divided into three parts according to specific use cases.

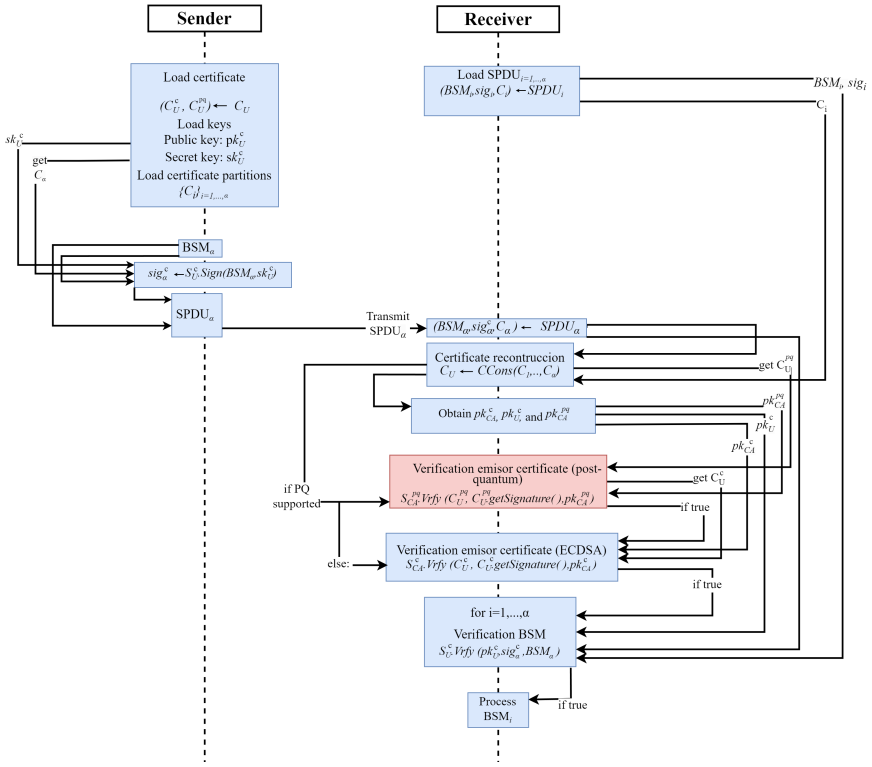
Figure 4.3a illustrates the use case where SPDU_i with $i = 1, ..., \alpha - 1$ is transmitted, representing the process of partitioning and transmitting the hybrid certificate, along with the BSM and its corresponding signature in the "Pre-α phase." During this process, the SPDU is transmitted, but the receiver keeps it on hold as verification cannot yet proceed. According to the analysis performed of the signature process, no acceleration is required at this stage of the protocol.

Figure 4.3b details the case where SPDUα is transmitted in the "In-α phase," which contains the final partition of the hybrid certificate, BSMα, and its respective signature. At this point, the receiver can proceed with the verification of the hybrid certificate as the reconstruction is possible. However, a bottleneck related to the verification of the post-quantum signature of the hybrid certificate (C_U^{pq}) has been identified, necessitating acceleration in this step, highlighted in the figure with a red box. Subsequently, the certificate's ECDSA signature is verified (C_U^c). This transition solution is compatible with the current protocol, allowing vehicles not PQC to skip this step and proceed directly to verify C_U^c. Finally, the verification of all BSM_i where $i = 1, ..., \alpha$ is performed. As mentioned above, no acceleration is necessary to verify C_U^c.

Lastly, Figure 4.3c describes the final case, the "Post-α phase," where the five-message cycle continues after α. Here, the sender signs only the BSM_i for $i = \alpha + 1, ..., 5$ and generates a hash of the certificate. These elements are attached in an SPDU that is issued. Once the receiver obtains each SPDU_i, it proceeds to verify the certificate's hash, followed by verifying the signature, and then processing the BSM. No acceleration is required in this case.

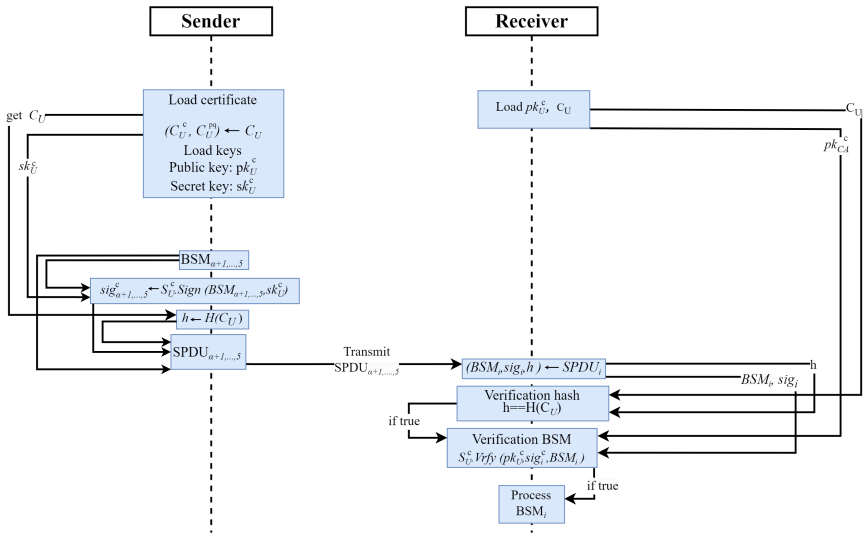


(a) Block diagram for the SPDU_i where $i = 1, ..., \alpha$



(b) Block diagram for the SPDU_α

Figure 4.3: Block diagram for V2V authentication. The red part corresponds to components subject to hardware acceleration.



(c) Block diagram for the SPDU$_i$ where $i = \alpha + 1, ..., 5$

Figure 4.3: Block diagram for V2V authentication.

4.4 Summary

This chapter focused on Hw/Sw co-design for quantum-resistant message authentication in V2V Communications. One of the design decisions was the choice of the most suitable post-quantum algorithm for this domain, with Falcon being the most convenient option. Compared to the other two standardized schemes, Falcon has the shortest signatures, approximately eleven times shorter than Sphincs$^+$ and four times shorter than Dilithium. This aspect is important in the design of the partially hybrid scheme, as it involves certificate partitioning. Falcon only requires the transmission of one SPDU for the receiver to acquire the full hybrid certificate, while Dilithium requires two, and Sphincs$^+$ requires four. Since decisions in this domain must be made as quickly as possible, certificate transmission must be fast. Also, in terms of verification performance, Falcon is about 5.00 times faster than Sphincs+ and 1.37 times faster than Dilithium, showing that Falcon has the best performance in this aspect. However, post-quantum algorithms are more computationally expensive than classical schemes. Signature verification resulted in the most time consuming part of the protocol, but also, it is feasible for hardware acceleration.

According to evaluations, Falcon verification is approximately 25 times slower than ECDSA, which creates a bottleneck in the verification of hybrid certificates, reducing the number of vehicles verified. Therefore, it is important to speed up this part. The rest of the protocol is not critical in terms of performance, so that part can be implemented in software.

CHAPTER 5

Implementation and results

This chapter discusses the experimental evaluation of the Hw/Sw co-design detailed in Chapter 4. It describes the implementation process and the experiments, as well as the evaluation platform and tools used. The results are discussed, and a comparison against a pure software solution of the hybrid scheme is presented, highlighting the acceleration obtained.

5.1 Implementation platform

The prototype of the Hw/Sw co-design detailed in Chapter 4 was developed, validated, and evaluated in the SoC PYNQ-Z2 [105], which is a TUL manufacturer platform created under the Xilinx University Program for developing embedded systems compatible with the PYNQ framework [88]. These platforms have the advantages of reducing costs, latencies, and space. Additionally, they allow for rapid prototyping and verification, offering flexibility when implementing or modifying algorithms. The outstanding specifications of PYNQ-Z2 are summarized in Table 5.1.

Components /Characteristics	**Descriptions**
SoC	Based on the XC7Z020-1CLG400C SoC core of the Zynq family. ARM Cortex-A9 dual-core processor, 650MHz.
Programmable Logic	13,300 logic slices, each with six 6-input LUTs and eight flip-flops. 630 KB of block RAM. 220 DSP blocks. On-chip analog-digital converter at 1 MSPS with six analog inputs. Allows programming through JTAG, Quad-SPI flash, or MicroSD card.
Memory and Storage	512MB DDR3 memory with 16-bit word width at 1050Mbps. 16MB Quad-SPI Flash with factory-programmed EUI-48/64. A MicroSD type expansion slot.
Alimentation	Through USB port or external regulator from 7V to 15V DC.
Connectivity USB and Ethernet	Gigabit Ethernet PHY. Micro USB port for programming via JTAG. Micro USB port for communication via UART bridge. USB 2.0 OTG PHY port (supports host mode only).

Table 5.1: PYNQ Z2 Specifications

The PYNQ-Z2 SoC can load a lightweight Linux image via the SD card, which allows the use of the PYNQ framework, which is a development platform for programming embedded systems from Xilinx, specifically the Zynq series SoCs (System on Chips), enabling access to all the board's features. Additionally, having an SoC with Linux for embedded systems offers several significant advantages: wide software

availability, flexibility and customization according to project needs, easy integration of new devices and peripherals, and support from the community and companies, ensuring long-term support.

5.2 Implementation details

Taking advantage of the workflow previously described, the hardware component in the proposed co-design, consisting of the Falcon's signature verification, was developed from a software specification. For that purpose, the implementation provided in Soni et al. [94] was utilized, which is based on the implementation developed by Schmid et al. [92]. For proper functioning, all the recursive functionality was removed, and certain libraries were changed to be compatible with the Vitis HLS framework.

Thus, the implementation of the hardware verification module (`Verify`) operates as follows: the function takes as input a public key, a message, and a digital signature, which includes both the nonce and the generated signature. Since the keys and signatures are generally in hexadecimal format, it is necessary to convert these hexadecimal representations into their binary equivalent, starting the Falcon signature verification process. First, the public key structure fv is initialized using `falcon_vrfy()`. The function `falcon_vrfy_set_public_key()` is responsible for setting the public key for fv. If the public key fails to meet specific criteria—such as having an insufficient size or containing reserved non-null bits—the verification process is halted and the signature is rejected. Then, the verification context is initialized with `falcon_vrfy_start()`. The signed message data is added with the call to `falcon_vrfy_update()`. Finally, `falcon_vrfy_verify()` verifies the signature against the injected message and the established public key, just as Falcon's verification mandates. Falcon signatures include an explicit nonce element, which must be provided as a parameter to `falcon_vrfy_start()`. If the verification is successful, the function returns a flag indicating whether the signature is valid or not. This procedure is formalized in Algorithm 1.

Along with the $Verify$ module in the SoC, the interconnection communicates

Algorithm 1 `Signature verification`

Requiere: Public key *(pk_hex)*, message *(msg)* , and signature *(sig_hex)*
Result: A flag that indicates if the signature is valid or not.
$pk \leftarrow decode(pk\_hex)$ `Hexadecimal to binary conversion`
$(nonce, sig) \leftarrow decode(sig\_hex)$ `Hexadecimal to binary conversion and division of sig and nonce`
Declare fv as `falcon_vrfy`
if !`falcon_vrfy_set_public_key`(fv, pk) **then**
    return "reject"
end if
`falcon_vrfy_start`$(fv, nonce)$
`falcon_vrfy_update`(fv, msg)
$z \leftarrow$ `falcon_vrfy_verify`(fv, sig)
if $z \leq 0$ **then**
    return "reject"
else
    return "valid"
end if

the hardware module and the main processor in the SoC (Hw/Sw interconnection) must also be specified using the Vitis HLS tool.

It is necessary to choose the ports correctly so that they are compatible with the co-design. The available axis ports in Vitis HLS are Simple AXI Lite (s_axilite) and Master AXI (m_axi).

1. **s_axilite:** It is a lightweight and simple memory access interface used to access blocks of memory that do not require full-width data transfers. It is suitable for applications where simplicity and low resource consumption are more important than performance efficiency.
2. **m_axi:** It is a more complete and versatile interface that is used to access blocks of memory that may require full-width data transfers and benefit from advanced transaction control features.

Two *s_axilite* interfaces were used to interconnect the *Verify* module: one to control the module via signals such as *start*, *reset*, and *done*. The other handles the results produced by the module. Additionally, an *m_axi* interface was utilized to

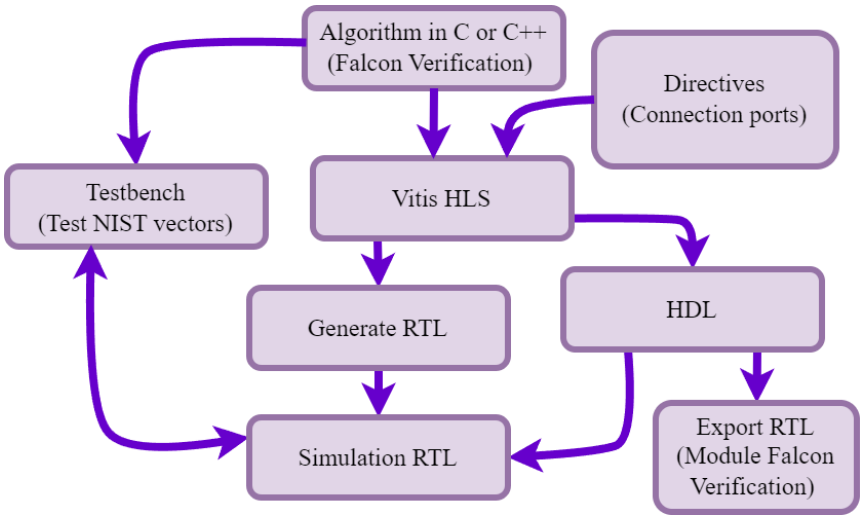


Figure 5.1: HLS design flow.

bundle all entries into a single package, including *hexpk*, the message, and the signature. This is done to reduce the number of calls to *m_axi*, which could cause a certain delay. These ports are added through directives, which later allow communication with the AXI Bus. Tests were carried out using the NIST test vectors [33]. The Vitis HLS design flow is shown in Figure 5.1.

5.3 Sintetized hardware

Once the IP (Falcon's signature verification) is created with Vitis HLS, the next step is to integrate and synthesize the design in Vivado following the design flow in Figure 5.2. The developed system in the Vivado block design of the interaction between the processor and the FPGA of the SoC is shown in Figure 5.3. This diagram highlights the Falcon verification IP block, developed from High-Level Synthesis. By following the Vivado design flow, the necessary files for programming the SoC are obtained.

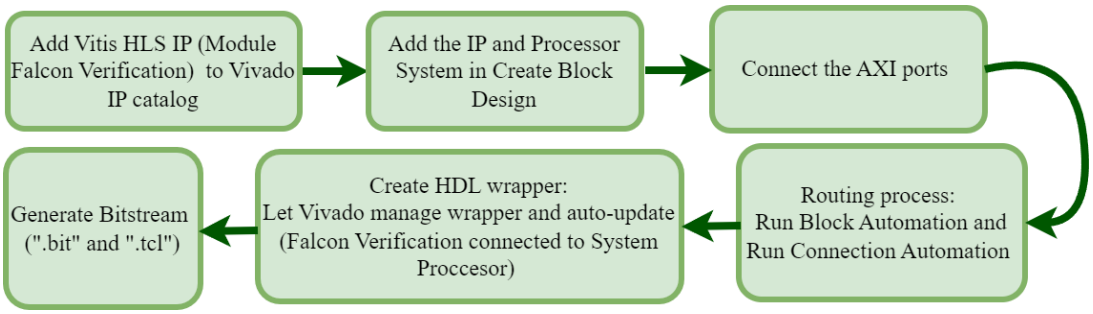


Figure 5.2: Vivado design flow.

5.4 Hw/Sw co-design implementation

Once the hardware implementation was done using High-Level Synthesis and the files necessary for programming the FPGA were generated by Vivado, the next step was the implementation of the co-design following the PYNQ methodology in Figure 5.4.

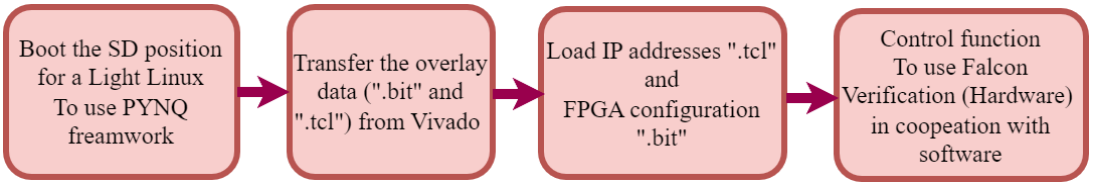


Figure 5.4: PYNQ design flow.

A script was created using the PYNQ framework to facilitate the loading of files intended for FPGA programming, along with a control function `Control Falcon` for the operation of the Falcon verification implemented in hardware. This procedure creates a buffer for the input data (key, signature, and message) accessed from the IP. When the input data is ready in the buffer, a start signal is asserted to begin the execution of the Falcon verification algorithm. Subsequently, the main process enters a waiting loop until the algorithm finishes. At this point, the memory address assigned to the output is read, and a valid or rejected verification signal is returned. Thus, the software can communicate with the hardware module for data transfer to

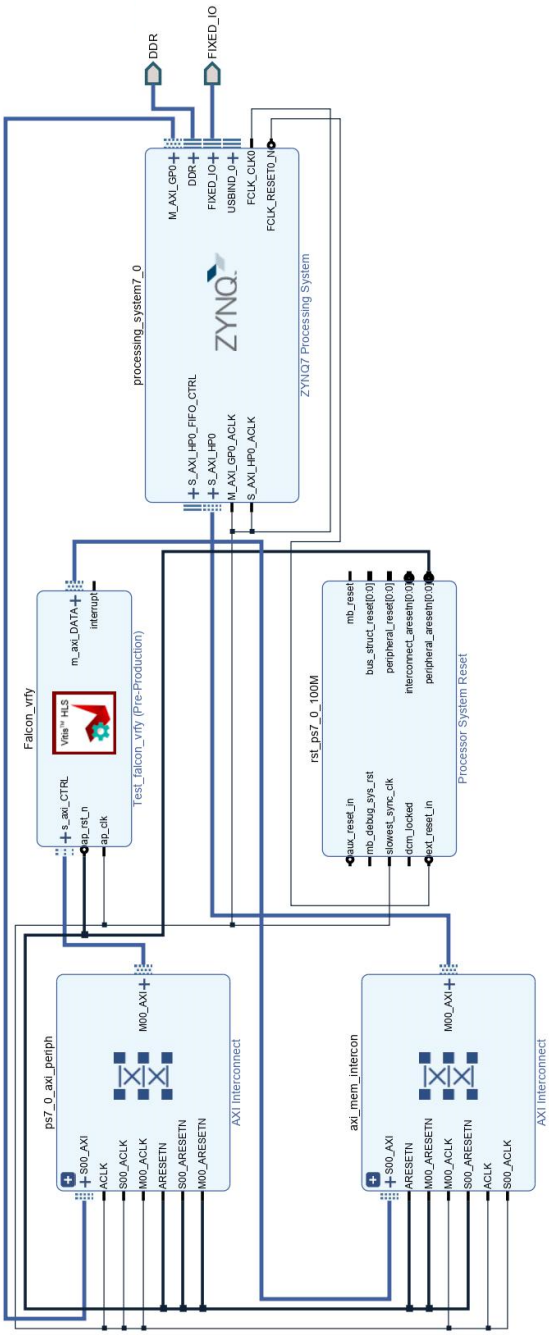


Figure 5.3: Prototype for quantum-resistant V2V communication using the HLS and Vivado methodology.

execute the Falcon verification algorithm. Once obtained, the verification result is transferred to the main program running in the processor.

5.5 Results

The experiments consist of evaluating the hardware acceleration of Falcon's verification, which involves varying the operating frequency of the module using the Vitis HLS tool. This approach requires optimizing the hardware to support the desired frequencies. It is anticipated that increasing the frequency will result in greater speedup, which will impact the co-design implementation. Additionally, the amount of resources needed for these configurations are measured, which are expected to increase as the frequency increases.

Subsequently, the co-design is evaluated using the measured frequencies to determine the achieved performance gain, specifically in terms of the number of vehicles that can be verified compared to a software-only implementation.

First, the results of Falcon's verification for two security levels (512 and 1024) are presented to observe the benefit obtained in hardware. Figure 5.5 shows the speed-up obtained in both cases, using Falcon 512 and Falcon 1024 in different configurations. The clock frequency started at 100 MHz, as it is relatively low and facilitates the initial evaluation of the design, ensuring a proper balance between processing speed and resource consumption [101]. The frequency was increased to assess the gain achieved in terms of performance. The frequencies were 200 MHz, 333 MHz, and 400 MHz. However, it was observed that at higher frequencies, the design failed to function correctly and did not pass the tests. This issue arises from the reduced time available for signals to propagate between registers at higher frequencies, which can lead to timing violations and result in design failures. For Falcon 512, the minimum gain using the lowest frequency (100 MHz) is 1.47×, while the maximum gain obtained at the highest frequency (400 MHz) is 3.40×. For Falcon 1024, the minimum gain is 1.90×, while the maximum gain obtained is 4.24×. The obtained results show the positive impact of hardware speed-up on Falcon verification performance. It highlights the potential of hardware implementation to optimize compute-intensive tasks such as

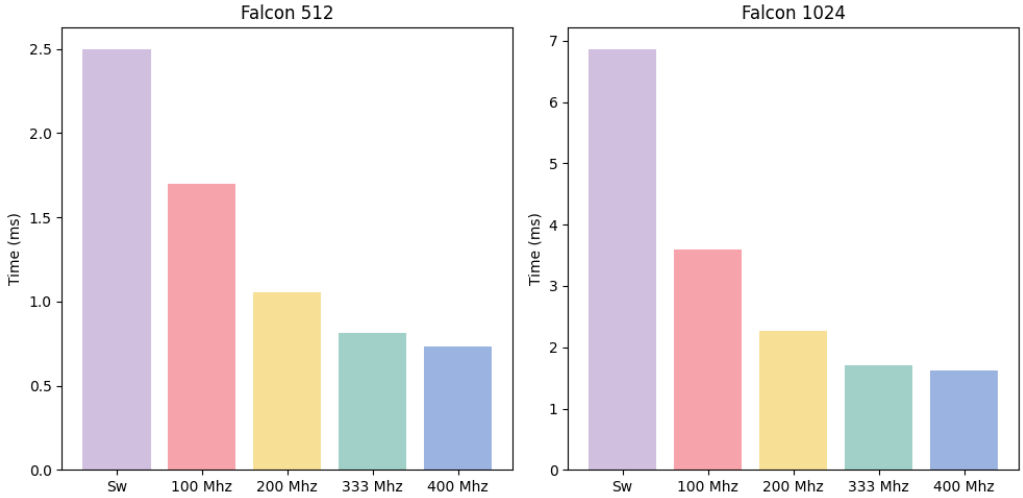


Figure 5.5: Accelerating Falcon's verification in hardware using different clock frequencies. The timing of the same operation in software is presented just for a reference.

post-quantum signature verification. Taking advantage of these results in V2V communications for message verification allows increasing the number of vehicles verified using the partially hybrid scheme in order to get closer to the recommended number of verified vehicles.

Table 5.2 details the resources used to verify the Falcon algorithm in different configurations, addressing two security levels. Each level is analyzed in terms of its operating frequency in MHz, the number of clock cycles needed for verification, as well as the utilization of specific FPGA resources, such as BRAM memory, Digital Processing Blocks (DSP), Flip-Flops (FF), Lookup Tables (LUT), and LUTRAM. It can be seen that at a higher frequency, the number of clock cycles increases, as well as the number of resources, despite the fact that it is the same design. Vitis HLS aims to achieve a design that operates at the desired frequency, which often results in higher resource usage and increased latency.

From this point forward, Falcon verification refers to the Falcon 512 security level, which is the recommended level for these communications, as indicated above.

The following shows the evaluation of the co-design for the authentication protocol in V2V communications using a hybrid scheme with the Falcon algorithm. During this evaluation, the delay caused by data transfer for the IP evaluation performed

Security level	**Frequency (MHz)**	**Clock cycles**	**BRAM**	**DSP**	**FF**	**LUT**	**LUTRAM**
Falcon 1024	400	646648	14 (10.00%)	27 (12.27%)	60630 (56.98%)	33179 (62.37%)	442 (2.54%)
	333	568558	15 (10.71%)	27 (12.27%)	54059 (50.80%)	31929 (60.01%)	379 (2.18%)
	200	453894	15 (10.71%)	27 (12.27%)	34690 (32.60%)	29755 (55.93%)	264 (1.52%)
	100	360093	15 (10.71%)	27 (12.27%)	20153 (18.94%)	26716 (50.21%)	210 (1.21%)
Falcon 512	400	293271	11 (7.85%)	24 (10.90%)	59386 (55.81%)	32395 (60.89%)	353 (2.03%)
	333	269856	13 (9.28%)	24 (10.90%)	53071 (49.87%)	31062 (58.38%)	342 (1.97%)
	200	210861	13 (9.28%)	24 (10.90%)	34251 (32.19%)	29343 (55.15%)	244 (1.40%)
	100	169570	13 (9.28%)	24 (10.90%)	19970 (18.76%)	26583 (49.96%)	210 (1.21%)

Table 5.2: Resources used for Falcon verification in different configurations.

by Falcon verification was analyzed. This process involves the input of the public key, message, and signature. A single entry is possible because the message or certificate has a fixed length, which allows it to be done efficiently, reducing the number of write calls necessary to activate the IP. To determine whether the signature was valid or rejected, a single-read call is made. The total delay includes the call to the write function, the information transfer, and the read function, which was approximately $300\mu s$. It is important to take this value into account when evaluating the performance of the co-design.

Considering the speed-up obtained with the frequencies of 333 MHz, taking into account the delay and the two ECDSA verifications, the time to verify the first message is 1.30 ms, equivalent to a speed-up of 2.07× capable of verifying approximately 77 vehicles in 100 ms. The frequency of 400 MHz gives a time of 1.23 ms, which would give a speed-up of 2.19×, capable of verifying 81 vehicles in 100 ms. Although 100 vehicles were not possible, the number was close to this amount compared to a software implementation. A total of 81 vehicles was reached, which is a cost that must be considered for post-quantum security in the transition stage. Table 5.3 shows the time performance in the verification of the SPDU's signatures according to the

Design	$\mathbf{SPDU}_1$	$\mathbf{SPDU}_{2,3,4,5}$	$\nu_{\max}$
Pure ECDSA(Sw)	0.20	0.14	500
Partially hybrid Falcon (Sw)	2.70	0.14	37
Partially hybrid Falcon (Hw/Sw) [a]	1.30	0.14	76
Partially hybrid Falcon (Hw/Sw) [b]	1.23	0.14	81

[a] 333 MHz, [b] 400 MHz

Table 5.3: Comparison of verification performance in a partially hybrid scheme in different configurations.

5-message cycle between a pure ECDSA scheme as well as the pure software hybrid scheme and the Hw/Sw co-design and the maximum number of vehicles (ν_{max}) that can be verified within a period of 100 ms.

As it can be observed, implementing a partially hybrid scheme purely in software is inefficient due to the limited vehicle verification capacity, which prevents taking advantage of the benefits of VANET communications. In contrast, through a co-design approach, the running time was accelerated by up to 2.19×, which significantly approaches the desired vehicle verification capacity and enhances VANET communication implementation. This achievement represents a positive advancement specifically derived from the co-design approach.

5.6 Summary

This chapter addressed the implementation and evaluation of the Hw/Sw co-design proposed in this thesis. An SoC was used to allow communication between the hardware component implemented in programmable logic and the system processor that executed the software part in the protocol. In this case, the PYNQ-Z2 was used, which incorporates an FPGA with an ARM Cortex-A9 processor. Vitis HLS enables the development of algorithms on high-level hardware using C/C++ code. For the hardware implementation of the Falcon verification, [94] was taken as a base; in addition, necessary functions were added for the co-design to work correctly in the communication protocol, which resulted in a speedup of Falcon verification of up to 3.40×.

Vivado tool was used to facilitate the integration of the hardware module into the SoC, ensuring a proper connection between the customized hardware and the other components of the SoC, such as the CPU. Vivado generated the configuration files necessary to program the FPGA with the implemented hardware design (Falcon verification).

Finally, the PYNQ framework was used, which offers a flexible way to validate and prototype complex embedded systems that combine hardware and software. Compared to a pure software implementation, co-design was accelerated up to 2.19×, resulting in an increase in vehicle verification from 37 to 81 vehicles.

CHAPTER 6

Conclusions

In this section, the contributions of this thesis, the obtained results, and future work are presented.

6.1 Contributions

Authentication in V2V communications is crucial due to the inherent risks of wireless transmission, such as denial of service, identity spoofing, and false message transmission, which threaten road safety. This thesis has addressed the limitations of the current ECDSA-based protocol, which, according to projections, will be vulnerable to quantum computing by the 2030s. In response to this threat, NIST has initiated the standardization of PQC algorithms, selecting in 2023 three digital signature algorithms: Crystals Dilithium, Sphincs$^+$, and Falcon, whose security is based on problems considered intractable even for quantum computers. However, these PQC algorithms present significant challenges compared to classical algorithms, such as increased memory usage, high processing costs, and longer execution times, which is critical for applications like V2V communications which operates under specific constrains, in terms of time and space. In this context, it is crucial to develop a solution that is secure against quantum computing and viable during the transition phase, considering vehicle longevity and coexistence with older technologies.

To address this threat, this work proposed a Hw/Sw co-design for efficient and resilient authentication in V2V communications within VANETs. The selected post-

quantum algorithm for this purpose is Falcon, due to its superior software performance, small key and signature sizes, and feasibility for hardware implementation. It was observed that the signature verification operation in this PQC algorithm is crucial for maintaining vehicle authentication capability in accordance with current recommendations.

It is important to note that validation was conducted in a controlled environment, which may limit the generalization of results to other vehicle configurations and network architectures. Additionally, the hardware implementation was carried out using FPGA, which may not accurately reflect all real-world Hw deployment environments. Implementation using HLS also increases resource consumption within the FPGA.

This work proposes a hybrid solution for the transition to a post-quantum era. The accelerated implementation of Falcon in hardware, through Hw/Sw co-design techniques, allows verification of up to 81 vehicles in 100 ms, representing a 2.19× acceleration compared to software implementation. Although this advancement does not fully cover the recommended number of vehicles, it is applicable in an environment with 80% of the suggested vehicle density, providing quantum-resistant authentication and suitability for the transition phase.

6.2 Future Work

Future work should focus on expanding the validation of the solution in more diverse and realistic environments. This includes testing in various vehicle configurations and network architectures to evaluate the generalization of the results obtained in the controlled environment. Additionally, exploring the implementation of the solution on different hardware platforms, beyond FPGAs, is recommended to ensure adaptability and effectiveness in a variety of deployment environments. It is also devised to consider implementing the post-quantum algorithm in a hardware description language like VHDL or verilog, which could reduce resource usage and offer greater acceleration.

List of Figures

List of Tables

Bibliography

[1] Syed Adnan Yusuf, Arshad Khan, and Riad Souissi. Vehicle-to-everything (v2x) in the autonomous vehicles domain – a technical review of communication, sensor, and ai technologies for road user safety. *Transportation Research Interdisciplinary Perspectives*, 23:100980, 2024. ISSN 2590-1982. doi: 10.1016/j.trip.2023.100980.

[2] Zehra Afzal and Manoj Kumar. Security of vehicular ad-hoc networks (vanet): A survey. *Journal of Physics: Conference Series*, 1427(1):012015, jan 2020. doi: 10.1088/1742-6596/1427/1/012015.

[3] Ali Al-Mahmood and Michael Opoku Agyeman. A study of fpga-based system-on-chip designs for real-time industrial application. *International Journal of Computer Applications*, 163:9–19, 04 2017. doi: 10.5120/ijca2017913544.

[4] Gorjan Alagic, Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, et al. Status report on the third round of the nist post-quantum cryptography standardization process. 2022.

[5] Gorjan Alagic, Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, et al. Status report on the third round of the nist post-quantum cryptography standardization process. 2022.

[6] Winfred Allgyer, Tyler White, and Tarek A. Youssef. Securing the future: A comprehensive review of post-quantum cryptography and emerging algorithms. pages 1282–1287, 2024. doi: 10.1109/SoutheastCon52093.2024.10500031.

[7] Nouri Alnahawi, Nicolai Schmitt, Alexander Wiesmaier, Andreas Heinemann, and Tobias Grasmeyer. On the state of crypto-agility. Cryptology ePrint Archive, Paper 2023/487, 2023.

[8] AMD. Vivado overview. https://www.xilinx.com/products/design-tools/vivado.html, 2023. Accessed, March. 10, 2024. [Online].

[9] Dorian Amiet, Lukas Leuenberger, Andreas Curiger, and Paul Zbinden. Fpga-based sphincs+ implementations: Mind the glitch. In *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pages 229–237, 2020. doi: 10.1109/DSD51259.2020.00046.

[10] Rameez Asif. Post-quantum cryptosystems for internet-of-things: A survey on lattice-based algorithms. *IoT*, 2:71–91, 02 2021. doi: 10.3390/iot2010005.

[11] Sarah Baras, Iman Saeed, Hadeel Tabaza, and Mourad Elhadef. *VANETs-Based Intelligent Transportation Systems: An Overview*, pages 265–273. 01 2018. ISBN 978-981-10-7604-6. doi: 10.1007/978-981-10-7605-3_44.

[12] Gourav Bathla, Kishor Bhadane, Rahul Singh, Rajneesh Kumar, Rajanikanth Aluvalu, Rajalakshmi Krishnamurthi, Adarsh Kumar, R N Thakur, and Shakila Basheer. Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities. *Mobile Information Systems*, 2022:1–36, 06 2022. doi: 10.1155/2022/7632892.

[13] Luke Beckwith, Duc Tri Nguyen, and Kris Gaj. High-performance hardware implementation of lattice-based digital signatures. Cryptology ePrint Archive, Paper 2022/217, 2022. Publication info: Preprint. MINOR revision.

[14] Daniel Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549: 188–194, 09 2017. doi: 10.1038/nature23461.

[15] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS+ signature framework. Cryptology ePrint Archive, Paper 2019/1086, 2019.

[16] Quentin Berthet, Andres Upegui, Laurent Gantel, Alexandre Duc, and Giulia Traverso. An area-efficient sphincs+ post-quantum signature coprocessor. In *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 180–187, 2021. doi: 10.1109/IPDPSW52791.2021.00034.

[17] N. Bindel and S. McCarthy. The need for being explicit: Failed attempts to construct implicit certificates from lattices. *Computer Journal*, 66(6):1320–1334, 2023. ISSN 0010-4620. doi: 10.1093/comjnl/bxac132.

[18] Botan. Crypto and tls for modern c++. https://botan.randombit.net, 2023. Accessed, Feb. 28, 2024. [Online].

[19] Benedikt Brecht, Dean Therriault, André Weimerskirch, William Whyte, Virendra Kumar, Thorsten Hehn, and Roy Goudy. A security credential management system for v2x communications. *IEEE Transactions on Intelligent Transportation Systems*, 19(12):3850–3871, 2018. doi: 10.1109/TITS.2018.2797529.

[20] Pierre-Louis Cayrel, Sidi Alaoui, Gerhard Hoffmann, Mohammed Meziani, and Robert Niebuhr. Recent progress in code-based cryptography, 01 2011.

[21] Dong Pyo Chi, Jeong Woon Choi, Jeong San Kim, and Taewan Kim. Lattice based cryptography for beginners. *Cryptology ePrint Archive*, 2015.

[22] Nicola Di Chiano, Riccardo Longo, Alessio Meneghetti, and Giordano Santilli. A survey on nist pq signatures. *ArXiv*, abs/2107.11082, 2021.

[23] M. Chiodo, P. Giusto, A. Jurecska, H.C. Hsieh, A. Sangiovanni-Vincentelli, and L. Lavagno. Hardware-software codesign of embedded systems. *IEEE Micro*, 14(4):26–36, 1994. doi: 10.1109/40.296155.

[24] DSRC Technical Committee. Dedicated short range communication (dsrc) systems engineering process guidance for sae j2945/x documents and common design concepts™, dec 2017.

[25] Louise H Crockett, Ross A Elliot, Martin A Enderwitz, and Robert W Stewart. *The Zynq book: embedded processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 all programmable SoC*. Strathclyde Academic Media, 2014.

[26] Yueqin Dai, Yifeng Song, Jing Tian, and Zhongfeng Wang. High-throughput hardware implementation for haraka in sphincs+. In *2023 24th International Symposium on Quality Electronic Design (ISQED)*, pages 1–6, 2023. doi: 10.1109/ISQED57927.2023.10129310.

[27] G. De Michell and R.K. Gupta. Hardware/software co-design. *Proceedings of the IEEE*, 85(3):349–365, 1997. doi: 10.1109/5.558708.

[28] Dharminder Dharminder and Dheerendra Mishra. LCPPA: Lattice-based conditional privacy preserving authentication in vehicular communication. *Transactions on Emerging Telecommunications Technologies*, 31(2):e3810, 2020. ISSN 2161-3915. doi: 10.1002/ett.3810.

[29] Jintai Ding and Albrecht Petzoldt. Current state of multivariate cryptography. *IEEE Security & Privacy*, 15(4):28–36, 2017.

[30] Bartosz Drzazga and Łukasz Krzywiecki. Review of chosen isogeny-based cryptographic schemes. *Cryptography*, 6(2):27, 2022.

[31] Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle. CRYSTALS – dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Paper 2017/633, 2017.

[32] Chuck Easttom. *Quantum Computing and Cryptography*, pages 397–407. Springer International Publishing, Cham, 2022. ISBN 978-3-031-12304-7. doi: 10.1007/978-3-031-12304-7_19.

[33] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over NTRU, 2023.

[34] José María de Fuentes García-Romero de Tejada. *Improvements on the enforcement process based on intelligent transportation techniques: model and mechanisms for electronic reporting, offence notification, and evidence generation.* PhD thesis, UNIVERSIDAD CARLOS III DE MADRID, July 2012.

[35] Tommaso Gagliardoni. Quantum attack resource estimate: Using shor's algorithm to break RSA vs DH/DSA VS ECC. https://research.kudelskisecurity.com/2021/08/24/quantum-attack-resource-estimate-using-shors-algorithm-to-break-rsa-vs-dh-dsa-vs-ecc/, 2021. Accessed, April. 10, 2024. [Online].

[36] Mao Gaoyu, Chen Donglong, Li Guangyan, Dai Wangchen, Sanka Abdurrashid Ibrahim, KoÇ Çetin Kaya, and Ray C. C. Cheung. High-performance and Configurable SW/HW Co-design of Post-quantum Signature CRYSTALS-Dilithium | ACM Transactions on Reconfigurable Technology and Systems. https://dl.acm.org/doi/10.1145/3569456, 2023.

[37] Linus Gasser. Post-quantum cryptography. In Valentin Mulder, Alain Mermoud, Vincent Lenders, and Bernhard Tellenbach, editors, *Trends in Data Protection and Encryption Technologies*, pages 47–52. Springer Nature Switzerland, 2023. ISBN 978-3-031-33386-6. doi: 10.1007/978-3-031-33386-6_10.

[38] Hadi Gharavi, Jorge Granjal, and Edmundo Monteiro. Post-quantum blockchain security for the internet of things: Survey and research directions. *IEEE Communications Surveys & Tutorials*, pages 1–1, 2024. doi: 10.1109/COMST.2024.3355222.

[39] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, April 2021. ISSN 2521-327X. doi: 10.22331/q-2021-04-15-433.

[40] Jens Groth and Victor Shoup. Design and analysis of a distributed ECDSA signing service. Cryptology ePrint Archive, Paper 2022/506, 2022.

[41] Hamsa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. Vanet security challenges and solutions: A survey. *Vehicular Communications*, 7:7–20, 01 2017. doi: 10.1016/j.vehcom.2017.01.002.

[42] Stephen Holmes and Liqun Chen. Assessment of quantum threat to bitcoin and derived cryptocurrencies. Cryptology ePrint Archive, Paper 2021/967, 2021.

[43] M.-L. How and S.-M. Cheah. Business renaissance: Opportunities and challenges at the dawn of the quantum computing era. *Businesses*, 3:585–605, 2023. doi: 10.3390/businesses3040036.

[44] James Howe, Thomas Prest, and Daniel Apon. Sok: How (not) to design and implement post-quantum cryptography. In *Cryptographers' Track at the RSA Conference*, pages 444–477. Springer, 2021.

[45] IBM. The IBM Quantum Development Roadmap. https://www.ibm.com/quantum/roadmap, 22 de Agosto de 2023. Accessed, Feb. 10, 2024. [Online].

[46] IEEE. Ieee standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications, 2021.

[47] IEEE. Ieee approved draft standard for wireless access in vehicular environments–security services for applications and management messages. *Std 1609.2-2022 (Revision of IEEE Std 1609.2-2016)*, pages 1–349, 2023. doi: 10.1109/IEEESTD.2023.10075082.

[48] Malik Imran, Zain Ul Abideen, and Samuel Pagliarini. An experimental study of building blocks of lattice-based NIST post-quantum cryptographic algorithms. *Electronics*, 9(11):1953, 2020. ISSN 2079-9292. doi:

10.3390/electronics9111953. Number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

[49] H. R. Ismael, S. Y. Ameen, S. F. Kak, H. M. Yasin, I. M. Ibrahim, A. M. Ahmed, and D. M. Ahmed. Reliable communications for vehicular networks. *Asian Journal of Research in Computer Science*, 10(2):33–49, 2021.

[50] J2945_201712. Dedicated Short Range Communication (DSRC) Systems Engineering Process Guidance for SAE J2945/X Documents and Common Design Concepts™ - SAE International. https://www.sae.org/standards/content/j2945_201712/, 2023.

[51] Thamer Jamel. Implementation of fft algorithm using fpga technique. https://api.semanticscholar.org/CorpusID:17916994, 01 2006.

[52] Vinita Jindal and Punam Bedi. Vehicular ad-hoc networks- introduction, standards, routing protocols and challenges. *International Journal of Computer Science Issues, IJCSI*, 13:44–55, 03 2016. doi: 10.20943/01201602.4455.

[53] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001. ISSN 1615-5262. doi: 10.1007/s102070100002.

[54] David Joseph, Rafael Misoczki, Marc Manzano, Joe Tricot, Fernando Dominguez Pinuaga, Olivier Lacombe, Stefan Leichenauer, Jack Hidary, Phil Venables, and Royal Hansen. Transitioning organizations to post-quantum cryptography. *Nature*, 605(7909):237–243, 05 2022. ISSN 1476-4687. doi: 10.1038/s41586-022-04623-2.

[55] Galathara Kahanda, Vraj Patel, Mihir Parikh, Michael Ippolito, Maansi Solanki, and Sakib Ahmed. The future era of quantum computing. In Hamid Jahankhani, editor, *Cybersecurity in the Age of Smart Societies*, pages 469–484, Cham, 2023. Springer International Publishing. ISBN 978-3-031-20160-8.

[56] Emre Karabulut and Aydin Aysu. A Hardware-Software Co-Design for the Discrete Gaussian Sampling of FALCON Digital Signature. Cryptology ePrint Archive, Paper 2023/908, 2023. Publication info: Preprint.

[57] Muhammet Ali Karabulut, A. F. M. Shahen Shah, Haci Ilhan, Al-Sakib Khan Pathan, and Mohammed Atiquzzaman. Inspecting vanet with various critical aspects – a systematic review. *Ad Hoc Networks*, 150:103281, 2023. ISSN 1570-8705. doi: 10.1016/j.adhoc.2023.103281.

[58] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing*. Oxford University Press, 11 2006. ISBN 9780198570004. doi: 10.1093/oso/9780198570004.001.0001.

[59] John B. Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011. doi: 10.1109/JPROC.2011.2132790.

[60] Jin-Woo Kim, Jae-Wan Kim, and Dong-Keun Jeon. A Cooperative Communication Protocol for QoS Provisioning in IEEE 802.11p/Wave Vehicular Networks. *Sensors*, 18(11):3622, 2018. ISSN 1424-8220. doi: 10.3390/s18113622. Number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

[61] Youngbeom Kim and Seog Chung Seo. Signature split method for a PQC-DSA compliant with v2v communication standards. *Applied Sciences*, 13(10):5874, 2023. ISSN 2076-3417. doi: 10.3390/app13105874. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute.

[62] Priya Kohli, Sakshi Painuly, Priya Matta, and Sachin Sharma. Future trends of security and privacy in next generation vanet. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pages 1372–1375, 2020. doi: 10.1109/ICISS49785.2020.9316043.

[63] Vipin Kumar Kukkala, Jordan Tunnell, Sudeep Pasricha, and Thomas Bradley. Advanced driver-assistance systems: A path toward autonomous vehicles.

IEEE Consumer Electronics Magazine, 7(5):18–25, 2018. doi: 10.1109/MCE.2018.2828440.

[64] Manish Kumar. Post-quantum cryptography algorithm's standardization and performance analysis. *Array*, 15:100242, 2022. ISSN 2590-0056. doi: 10.1016/j.array.2022.100242.

[65] Hee-Yong Kwon, Indra Bajuna, and Mun-Kyu Lee. Compact hybrid signature for secure transition to post-quantum era. *IEEE Access*, pages 1–1, 2024. doi: 10.1109/ACCESS.2024.3374645.

[66] Georg Land, Pascal Sasdrich, and Tim Güneysu. A hard crystal - implementing dilithium on reconfigurable hardware. Cryptology ePrint Archive, Paper 2021/355, 2021.

[67] Carlos Andres Lara-Nino, Arturo Diaz-Perez, and Miguel Morales-Sandoval. Post-quantum cryptography for embedded systems. In *2022 IEEE Mexican International Conference on Computer Science (ENC)*, pages 1–8, 2022. doi: 10.1109/ENC56672.2022.9882904.

[68] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & Information Systems Engineering*, 6(4):239–242, 2014. ISSN 1867-0202. doi: 10.1007/s12599-014-0334-4.

[69] Michael Lee and Travis Atkison. Vanet applications: Past, present, and future. *Vehicular Communications*, 28:100310, 2021. ISSN 2214-2096. doi: 10.1016/j.vehcom.2020.100310.

[70] Sharon Levy. Performance and security of ecdsa. https://api.semanticscholar.org/CorpusID:12722775, 2015.

[71] Fengyin Li, Yang Cui, Junhui Wang, Huiyu Zhou, Xiaoying Wang, and Qintai Yang. Lattice-based batch authentication scheme with dynamic identity revocation in VANET. *International Journal of Intelligent Systems*, 37(11):9442–9460, 2022. ISSN 1098-111X. doi: 10.1002/int.23004.

[72] Zhe Liu, Kim-Kwang Raymond Choo, and Johann Grossschadl. Securing edge devices in the post-quantum internet of things using lattice-based cryptography. *IEEE Communications Magazine*, 56(2):158–162, 2018. doi: 10.1109/MCOM.2018.1700330.

[73] D. Manivannan, Shafika Showkat Moni, and Sherali Zeadally. Secure authentication and privacy-preserving techniques in vehicular ad-hoc networks (vanets). *Vehicular Communications*, 25:100247, 2020. ISSN 2214-2096. doi: 10.1016/j.vehcom.2020.100247.

[74] Roger Massmann, Nick Grantham, and Akalanka Mailewa. Quantum computing: An assessment into the impacts of post-quantum cryptography, 2023.

[75] Kunal Meher and Divya Midhunchakkaravarthy. New approach to combine secret keys for post-quantum (pq) transition. *Indian Journal of Computer Science and Engineering*, 12(3):629–633, 2021.

[76] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, pages 369–378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg. ISBN 978-3-540-48184-3.

[77] Ralph C Merkle. A certified digital signature. In *Conference on the Theory and Application of Cryptology*, pages 218–238. Springer, 1989.

[78] Akmalov Zayniddin Mexriddinovich. Safeguarding digital security: Addressing quantum computing threats. *The Role of Exact Sciences in the Era of Modern Development*, 1(4):1–7, Oct. 2023. https://uzresearchers.com/index.php/RESMD/article/view/873.

[79] Mohd Mohamad, Roshidi Din, and Jasmin Ahmad. Research trends review on rsa scheme of asymmetric cryptography techniques. *Bulletin of Electrical Engineering and Informatics*, 10(1):487–492, 2021. ISSN 2302-9285. doi: 10.11591/eei.v10i1.2493.

[80] Eric Monmasson, Mickaël Hilairet, Giovanni Spagnuolo, and Marcian N. Cirstea. System-on-chip fpga devices for complex electrical energy systems control. *IEEE Industrial Electronics Magazine*, 16(2):53–64, 2022. doi: 10.1109/MIE.2021.3052179.

[81] Sankar Mukherjee, Daya Sagar Gupta, and G. P. Biswas. An efficient and batch verifiable conditional privacy-preserving authentication scheme for VANETs using lattice. *Computing*, 101(12):1763–1788, dec 2019. ISSN 0010-485X, 1436-5057. doi: 10.1007/s00607-018-0689-3.

[82] M.M.A. Muslam. Enhancing security in vehicle-to-vehicle communication: A comprehensive review of protocols and techniques. *Vehicles*, 6:450–467, 2024. doi: 10.3390/vehicles6010020.

[83] Taleb Samad Obaid. Study a public key in rsa algorithm. *European Journal of Engineering and Technology Research*, 5(4):395–398, Apr. 2020. doi: 10.24018/ejeng.2020.5.4.1843.

[84] Open-Quantum-Safe. Github - open-quantum-safe/liboqs: C library for prototyping and experimenting with quantum-resistant cryptography. https://github.com/open-quantum-safe/liboqs, 2023. Accessed, Feb. 20, 2024. [Online].

[85] Oscar Orozco Sarasti and Gonzalo Ramírez. Aplicaciones para redes vanet enfocadas en la sostenibilidad ambiental, una revisión sistemática. *Ciencia e Ingeniería Neogranadina*, 24:111–132, 06 2014. doi: 10.18359/rcin.396.

[86] Binod Kumar Pattanayak, Omkar Pattnaik, and Sasmita Pani. Dealing with sybil attack in vanet. In Debahuti Mishra, Rajkumar Buyya, Prasant Mohapatra, and Srikanta Patnaik, editors, *Intelligent and Cloud Computing*, pages 471–480, Singapore, 2021. Springer Singapore. ISBN 978-981-15-5971-6.

[87] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. L. Pereira, M. Razavi, J. Shamsul Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and

P. Wallden. Advances in quantum cryptography. *Adv. Opt. Photon.*, 12(4): 1012–1236, Dec 2020. doi: 10.1364/AOP.361502.

[88] PYNQ. Python productivity for zynq. http://www.pynq.io/board.html, 2023. Accessed, March. 10, 2024. [Online].

[89] Asim Rasheed, Saira Gillani, Sana Ajmal, and Amir Qayyum. Vehicular ad hoc network (vanet): A survey, challenges, and applications. In Anis Laouiti, Amir Qayyum, and Mohamad Naufal Mohamad Saad, editors, *Vehicular Ad-Hoc Networks for Smart Cities*, pages 39–51, Singapore, 2017. Springer Singapore. ISBN 978-981-10-3503-6.

[90] M. Sadaf, Z. Iqbal, A. R. Javed, I. Saba, M. Krichen, S. Majeed, and A. Raza. Connected and automated vehicles: Infrastructure, applications, security, critical challenges, and future aspects. *Technologies*, 11(5):117, 2023. doi: 10.3390/technologies11050117.

[91] Ahmad Salman and Zachary Blankinship. Implementing butterfly key expansion using post-quantum algorithms. In Xin-She Yang, Simon Sherratt, Nilanjan Dey, and Amit Joshi, editors, *Proceedings of Seventh International Congress on Information and Communication Technology*, Lecture Notes in Networks and Systems, pages 507–516. Springer Nature, 2023. ISBN 978-981-19160-7-6. doi: 10.1007/978-981-19-1607-6_45.

[92] Michael Schmid, Dorian Amiet, Jan Wendler, Paul Zbinden, and Tao Wei. Falcon takes off - a hardware implementation of the falcon signature scheme. Cryptology ePrint Archive, Paper 2023/1885, 2023. Publication info: Preprint.

[93] Syed Sarmad Shah, Asad Waqar Malik, Anis U. Rahman, Sohail Iqbal, and Samee U. Khan. Time barrier-based emergency message dissemination in vehicular ad-hoc networks. *IEEE Access*, 7:16494–16503, 2019. doi: 10.1109/ACCESS.2019.2895114.

[94] Deepraj Soni, Kanad Basu, Mohammed Nabeel, Najwa Aaraj, Marc Manzano, and Ramesh Karri. *Hardware Architectures for Post-Quantum Digital Signa-*

ture Schemes. Springer Cham, 1 edition, 2021. ISBN 978-3-030-57681-3. doi: 10.1007/978-3-030-57682-0. Springer Nature Switzerland AG.

[95] Statista. Average age of vehicles in u.s. 2023 — statista. https://www.statista.com/statistics/738667/us-vehicles-projected-age/, March 2024. Accessed: 2024-04-12.

[96] Sára Szatmáry. Quantum computers—security threats and solutions. In Tünde Anna Kovács, Zoltán Nyikes, Tamás Berek, Norbert Daruka, and László Tóth, editors, *Critical Infrastructure Protection in the Light of the Armed Conflicts*, pages 431–441, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-47990-8.

[97] Mohammed Talib, Aslinda Hassan, Burairah Hussin, Ali Mohammed, Ali Hassan, and Ammar Mutlag. Vehicular ad hoc network for intelligent transport system: A review. 7:350–353, 12 2018. doi: 10.14419/ijet.v7i4.36.23803.

[98] Jürgen Teich. Hardware/software codesign: The past, the present, and predicting the future. *Proceedings of the IEEE*, 100(Special Centennial Issue): 1411–1430, 2012. doi: 10.1109/JPROC.2011.2182009.

[99] Oguz Tengilimoglu, Oliver Carsten, and Zia Wadud. Implications of automated vehicles for physical road environment: A comprehensive review. *Transportation Research Part E: Logistics and Transportation Review*, 169:102989, 2023. ISSN 1366-5545. doi: 10.1016/j.tre.2022.102989.

[100] Geoff Twardokus, Nina Bindel, Hanif Rahbari, and Sarah McCarthy. When cryptography needs a hand: Practical post-quantum authentication for v2v communications. *Network and Distributed System Security (NDSS) Symposium*, 2022. doi: 10.14722/ndss.2024.24267.

[101] David Alejandro Urquiza Villalonga and Jorge Torres Gómez. Diseño en fpga de esquema de detección de señales acoplado a sistema de recolección de energía inalámbrica. *Telemática*, 18(3), 2019.

[102] Tengfei Wang, Chi Zhang, Pei Cao, and Dawu Gu. Efficient implementation of dilithium signature scheme on FPGA SoC platform. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(9):1158–1171, 2022. ISSN 1557-9999. doi: 10.1109/TVLSI.2022.3179459. Conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

[103] W. Wolf. A decade of hardware/software codesign. *Computer*, 36(4):38–43, 2003. doi: 10.1109/MC.2003.1193227.

[104] Marie A Wright. The advanced encryption standard. *Network Security*, 2001 (10):11–13, 2001. ISSN 1353-4858. doi: 10.1016/S1353-4858(01)01018-2.

[105] Xilinx. Pynq-z2 development board is based on the xilinx zynq xc7z020 system on chip (soc), 2019. https://www.amd.com/en/corporate/university-program/aup-boards/pynq-z2.html.

[106] Xilinx Inc. Xilinx high-level synthesis user guide. https://www.xilinx.com/products/design-tools/vitis/vitis-hls.html, 2023. Accessed, March. 10, 2024. [Online].

[107] Engin Zeydan, Yekta Turk, Berkin Aksoy, and S. Bugrahan Ozturk. Recent advances in post-quantum cryptography for networks: A survey. pages 1–8, 2022. doi: 10.1109/MobiSecServ50855.2022.9727214.

[108] Zhen Zhou, Debiao He, Zhe Liu, Min Luo, and Kim-Kwang Raymond Choo. A Software/Hardware Co-Design of Crystals-Dilithium Signature Scheme. *ACM Transactions on Reconfigurable Technology and Systems*, 14(2):11:1–11:21, June 2021. ISSN 1936-7406. doi: 10.1145/3447812.