

Simulation of Vehicle-to-Vehicle Communication based on selected PQC-DSA

YoungBeom Kim¹, Seog Chung Seo^{1*}

¹Department of Financial Information Security, Kookmin University, South Korea

Abstract

With the development of quantum-computing system, the security of existing public-key-based communication systems is under great threat. As a result, National Institute of Standards and Technology (NIST) started Post-Quantum Cryptography (PQC) Standardization in 2015, and now research on applying PQC to various cryptographic-protocols is actively underway. Unlike elliptic-curve-based-systems, PQC requires large memory occupancy and ciphertext/signature size; therefore, research is mainly done to optimize PQC for low-end processor. Although there are many studies in which optimized PQC software has been applied to secure protocols, it has been relatively poorly studied in vehicle-to-vehicle communication. Especially in an autonomous-driving environment, fast signature generation and verification are essential. The purpose of this research is to apply optimized PQC-DSA, selected for standardization, to V2Verifier and simulate signature generation and verification. We also discuss how to apply DSA schemes with large signature sizes in an autonomous-driving environment.

Keywords: V2Verifier, Post-Quantum Cryptography, Vehicle to Vehicle Communication, Digital Signature Algorithm

1 Introduction

Current public-key cryptographic-systems are designed as RSA and elliptic-curve theorem. Alternatives to existing public-key cryptographic-system schemes are needed, depending on the feasibility of quantum-circuits for the Shor-algorithm effective for discrete logarithms and small factorization [1]. This prompted National Institute of Standards and Technology (NIST) to launch a new hardness-based Post-Quantum Cryptography (PQC) competition, which now has selected for standardization three lattice-based algorithms and one hash-based algorithm [2]. However, PQC algorithms submitted to competition have major drawbacks in terms of signature/public key size compared to existing public key scheme algorithms, and also have a disadvantage in terms of speed [3]. Therefore, it is necessary to install the PQC cryptographic algorithm in the existing secure-protocol and evaluate its performance analysis.

Vehicle-to-Vehicle (V2V) is a protocol for communication in an autonomous-driving environment. Autonomous vehicles receive real-time information from each other vehicles, knowing the position of each vehicle, and pre-calculate the movement route. Without cryptographic-system between autonomous-driving communications, real-time repositioning, rerouting, etc. by attackers could lead to security incidents and fatalities [4]. Therefore, in an autonomous-driving environment, secure communication for inter-vehicles must be applied inherently. However, when applying the PQC algorithm chosen for standardization into a secure-protocol, the actual communication layer parameters such as latency, signature

The 6th International Symposium on Mobile Internet Security (MobiSec'22), December 15-17, 2022, 2021, Jeju Island, Republic of Korea

Article No. 1, pp. 1-7

*Corresponding author: Department of Financial Information Security, Kookmin University, South Korea,
email: scseo@kookmin.ac.kr

generation and verification latency, signature size, packet limits on public key size, etc. constraints should be carefully considered [5].

To date, research on PQC-based V2V protocols is inadequate compared to research done on other secure-protocols such as TLS/SSL, IPSEC, etc [6, 7]. In particular, benchmark studies of the latest optimizations of PQC algorithm have not yet been done. Therefore, in this paper, we apply the selected PQC-DSA algorithm, optimized using the latest research results, to V2Verifier to simulate signature generation and verification. We also discuss how to apply DSA schemes with large signature sizes (especially Crystals-Dilithium) to autonomous-driving environments.

2 Preliminary

Post-Quantum Cryptography is a next-generation public-key cryptographic system, an alternative to existing public-key cryptography, which its hardness is threatened by the quantum-circuit algorithm presented by Shor in 1994. NIST is conducting competitions in two fields, Public Key Encryption/Key Encapsulation Mechanism (PKE/KEM) and Digital Signature Algorithm (DSA), to standardize secure PQC in the quantum computing environment, and the algorithms targeted for final standardization have now been announced. The cryptographic-scheme used in V2Verifier, the target of this paper, is a DSA, and the final standardized algorithms are Crystals-Dilithium and Falcon, both of which are lattice-based algorithms. Crystals-Dilithium puts its security proofs on the Learning With Errors (LWE) scheme, and Falcon puts its security proofs on the Short Vectors Problem (SVP). Table 1 shows the parameters of the two algorithms.

V2Verifier is an open-source project dedicated to wireless experimentation focused on the security of V2V communications. V2Verifier project features the first open-source implementation of the IEEE 1609.2 standard for V2V security [8]. V2Verifier uses a combination of software-defined radio such as Universal Software Radio Peripheral (USRP) and commercial V2V equipment to support extensive experimentation with V2V technology and security protocols. Table 2 shows the attributes required for V2V communication, with the following supported.

1. Security features from the IEEE 1609.2 standard for V2V security, including message signing and verification and V2V certificates
2. WAVE Short Message Protocol (IEEE 1609.3)
3. Dedicated Short Range Communication (DSRC) - adapted from the WiME Project's IEEE 802.11p transceiver
4. Cellular Vehicle-to-Everything (C-V2X) - based on the srsRAN project (formerly srsLTE) (temporarily not supported)

V2Verifier is designed to run on Software Defined Radio (SDR). V2Verifier consists of communication between multiple wireless (hereafter: Remote) vehicles and one receiving (hereafter: Local) vehicle. To run the actual V2Verifier, one PC is responsible for transmitting multiple Remote vehicles, and another PC is responsible for receiving Local vehicles.

Recently, researchers at the University of Waterloo, Canada, applied the NIST PQC Signature Round3 candidates (Crystals-Dilithium, Falcon, Rainbow) to V2Verifier at the 2021 NIST PQC conference [5].

Table 1: Parameters of Crystals-Dilithium and Falcon

	Crystals-Dilithium			Falcon		
Problem	Module-LWE			NTRU		
Parameters (bytes)	public key	secret key	signature	public key	secret key	signature
1	-	-	-	897	1,281	690
2	1,312	2,528	2,420	-	-	-
3	1,952	4,000	3,293	-	-	-
5	2,592	4,864	4,595	1,793	2,305	1,330

They presented an experiment to verify the feasibility of the PQC-DSA algorithm in a secure V2V communication (IEEE 1609.2) environment as a replacement for ECDSA, an existing elliptic curve-based digital signature. In terms of packet size and delay time, the applicability of PQC-DSA electronic signature algorithm to autonomous driving environment is analyzed. In addition, the test was designed considering various environments, such as the general autonomous driving environment and the platooning driving stage.

According to the result of [5], the minimum signature size of Crystals-Dilithium is about 2500 bytes based on security level 2, which exceeds WAVE's packet size of 2000 bytes; therefore, verification fails during actual communication. Falcon has a maximum signature size of about 1330 bytes, so it meets WAVE's requirements. For signature generation, Crystals-Dilithium achieved better performance than ECDSA, while Falcon achieved lower performance than ECDSA. Both Crystals-Dilithium and Falcon outperformed ECDSA in signature verification.

Table 2: Attribute parameters required for V2X communication

type	WAVE (IEEE 802.11p)	C-V2X (3GPP Rel.14)	eC-V2X (3GPP Rel.15)	5G-V2X
Processing rate (Mbps)	3~27	10~100 (Maximum 1000)		Maximum 2000
BSM packet size	2,000	400 (advanced : 12,000)		platooning : 6,000 advanced : 12,000 autonomous : 41,700
Maximum communication radius	1 km	320m	1km	platooning : 350m advanced : 700m autonomous : 1km
End to end Latency (ms)	100	100	100 (platooning : 10)	platooning : 25 advanced : 100 autonomous : 5
Vehicle maximum relative speed (km/h)	200	280		500
Correctness		80%~95%	90%~100%	90%~100%

3 Simulation

3.1 Selection of Code

Selected algorithms to be standardized in the DSA field among PQC are Crystals-Dilithium and Falcon. In Crystals-Dilithium, various optimization methods have been proposed for low-end Processor and high-end Processor. In this study, we have V2Verifier as our testbed, and since the most commonly used equipment in autonomous driving environments are ARMv8-based On Board Units (OBU), we investigated the latest optimization implementations on that equipment. Among them, we select [2]’s research done on OBU equipment as a test code. For Falcon, there is only [9] optimization study done in an embedded environment, and we choose [9]’s code as the target code.

3.2 Application plan for V2Verifier

The structure of V2Verifier simulates a situation where one local car sends and receives communication with n remote cars. In this research, we proceed with the simulation using the same structure. Unfortunately, equipment limitations existed in this study, and we were not able to use the original V2Verifier test equipment, the HW USRP (Universal Software Radio Peripheral). Therefore, we modify the code to simulate on one PC and configure local and remote cars through multiprocessing. The api that local cars run is **run_local** and the api that remote cars run is **run_remote**. Here’s what each function does:

1. **run_remote** : This process maintains the IDs and locations of n cars in context. The payload generation process includes a signature to the current location. Also, when the payload is generated, it uses multiprocessing to send the payload to the **run_local** api. The initial communication with each vehicle’s local vehicle involves the process of generating a key used for signing.
2. **run_local** : The local car receives a payload containing the remote car’s current position. It then goes through the payload’s signature verification process and deconstructs the payload when the verification is successful.

V2Verifier is python based code, and PQC-DSA codes suggested in [9] and [2] are C code mixed with hand-written assembly. To apply it to Python language, recompile that code to gcc and build it in the form of a dynamic library. First, we generate a Payload using the coordinate information used in the V2Verifier source code (choose the number of coordinates variably). We consider and generate packets according to signature size and WAVE protocol requirements. For example, the minimum signature size for Crystals-Dilithium is approximately 2500 bytes (security level 2), which exceeds the WAVE protocol requirement of 2000 bytes. However, testing is possible because our simulation proceeds without actual communication. For Falcon, the maximum signature size is about 1330 bytes (Falcon-1024) and the total payload length (1330 + 500) meets WAVE’s requirements. Here are the steps to simulate:

1. **generating payload** : V2Verifier generates payload in V2Verifier using `Get_wsm_payload` api. The payload consists of three components (Bytestring, Wsm-header and IEE 1609.data). The payload consists of a total of 16 pieces of information. The payload requires approximately 500 bytes, excluding signature length information and the signature.

2. **generating key pair** : The process for generating public and private keys for PQC-DSA. It is done in the first session for local and remote cars. It doesn't work on session reconnect.
3. **signing payload** : The process by which the remote car generates a signature to send its current location to the local car.
4. **verifying signature** : The process by which the remote car generates a signature to send its current location to the local car.
5. **descomposing payload** : Once validated, it is the process of breaking down the payload.

Table 3: Measurements including [generating payload + generating key pair + signing payload + verifying signature + descomposing payload] for 300 payloads

Algorithm	Security	Reference (ms/300)	Optimization (ms/300) [2, 9]
ECDSA using P-256 python module	1	3.9	-
Dilithium	2	2.24	1.87
	3	4.06	2.36
	5	4.73	2.98
Falcon	1	14.28	12.68
	5	38.91	33.31

Table 4: Measurement result of maximum termination delay time in $\frac{\text{run_remote}}{\text{run_local}}$: marked by

Algorithm	Security	Reference (ms/300)	Optimization (ms/300) [2, 9]
ECDSA using P-256 python module	1	$\frac{1.78}{2.9}$	-
Dilithium	2	$\frac{1.69}{0.28}$	$\frac{1.43}{0.19}$
	3	$\frac{3.06}{0.52}$	$\frac{1.86}{0.24}$
	5	$\frac{3.30}{0.67}$	$\frac{2.19}{0.40}$
Falcon	1	$\frac{1.15}{0.09}$	$\frac{1.10}{0.06}$
	5	$\frac{2.17}{0.16}$	$\frac{2.04}{0.11}$

* run_remote : generating payload + signing payload
run_local : verifying signature + descomposing payload

4 Result

NVIDIA Jetson AGX Xavier is our target platform containing a 64-bit ARM A5x processor with 32 GB of RAM, 32 GB of storage, and 512-Core Volta GPU with 64-Tensor Cores (8 SM, 64kB register, 128kB shared memory). All PQC-DSA code for our simulation is compiled with gcc with `-shared, -O3` options and integrated into V2Verifier in the form of a dynamic library. The benchmark uses exactly the method described in the simulation process and performs two tests. The first is when the remote and local cars make up the first session, and the second simulation is when the remote and local cars communicate in successive sessions. Each apis are calculated with 300 average values. It inserts dummy operations in between to avoid using cache memory, and this process is excluded when measuring performance. Configure the **run_remote** and **run_local** apis in multiprocess. The operating environment is Linux environment and the kernel version is 4.4.0-59-generic.

Table 3 shows measurements including [generating payload + generating key pair + generating signature + verifying signature] for average of 300 payloads. For the scenario when a single remote car communicates with a local car, and for Crystals-Dilithium, the software applied the technique proposed in [2] achieved up to 72% performance improvement over the reference code.

Table 4 shows measurement result of maximum termination delay time in `run_remote`, `run_local` : marked by $\frac{\text{run_remote}}{\text{run_local}}$. In this scenario it consists of the same scenario as Table 3, only the measurement function has changed. We measured the maximum termination delay time. Both the Reference code and the optimized Crystals-Dilithium, Falcon code achieve a maximum termination delay of less than 5m/s for 5G-V2X in autonomous driving environments.

5 Discussion

There are two main points of discussion in this section. The first is the need to apply an optimized PQC-DSA compared to the reference implementation. Based on a single packet, the reference implementation already meets the run5G-V2X requirement of less than 5 m/s latency. However, remote vehicles need to continuously transmit their location to nearby remote vehicles.

Therefore, research should be advanced not only on the performance of a single **run_remote**, but also on generating signatures concurrently using parallel implementations later. Simultaneous verification of signatures should also be considered for local vehicles. Therefore, parallel optimization implementation methodology for PQC should be considered in the future.

The second case is when the signature size in a packet size limited protocol such as WAVE exceeds the packet size. If the actual transmission layer does not guarantee packet ordering reliability, such as TCP, then it may not arrive in the correct order at the receiver. Especially in the case of Crystals-Dilithium, the signature size is basically 2000 bytes or more, so it is always divided and sent in the WAVE environment. No specific solution is proposed, but the key is how to ensure reliability while still meeting IEEE 802.11p. The most promising method is the operation of PQC in which the sending and receiving layers are application layers. Similar to generating a BSM, the signature order is packed into the msb of the signature and used so that once an agreement is reached between the sender and receiver, the receiver can verify the signature correctly. Of course, this is a temporary solution. Furthermore, for WAVE protocol, one bit can represent the order regardless of the Crystals-Dilithium security level, whereas more restrictive protocols than WAVE require more bits to represent the order.

6 Conclusion

In this paper, we proceed with research to apply the optimized implementation of PQC-DSA proposed so far to V2Verifier, an open source test-bed for Vehicle-to-Vehicle Communication. We formulate a simulation method for testing with V2Verifier, closely investigate the performance of V2Verifier applying PQC-DSA, and perform detailed performance analysis based on IEEE 1609.3 security features. In addition, for WAVE protocol, which has packet size limitations, we describe how to apply large signature size schemes.

Acknowledgments

YoungBeom Kim writes-original draft and Seog Chung Seo writes-review and edits the article. All authors have read and agreed to the published version of the manuscript. The authors would like to thank the anonymous reviewers for their helpful comments. This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.A2021-0270, 6G autonomous security internalization-based technology research to ensure security quality at all times, 50%) and the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2022R1C1C1013368, 50%)

Conflict of interest

All authors have no conflict of interest

References

- [1] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.
- [2] Youngbeom Kim, Jingyo Song, Taek-Young Youn, and Seog Chung Seo. Crystals-dilithium on armv8. *Security and Communication Networks*, 2022, 2022.
- [3] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum tls without handshake signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1461–1480, 2020.
- [4] Nina Bindel, Sarah McCarthy, Geoff Twardokus, and Hanif Rahbari. Drive (quantum) safe!—towards post-quantum security for v2v communications. *Cryptology ePrint Archive*, 2022.
- [5] Nina Bindel. Suitability of 3rd round signature candidates for vehicle-to-vehicle communication, June 07, 2021. available online: <https://csrc.nist.gov/Presentations/2021/suitability-of-3rd-round-signature-candidates-for>.
- [6] Maximilian Schöffel, Frederik Lauer, Carl C Rheinländer, and Norbert Wehn. On the energy costs of post-quantum kems in tls-based low-power secure iot. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, pages 158–168, 2021.
- [7] Kevin Bürstinghaus-Steinbach, Christoph Krauß, Ruben Niederhagen, and Michael Schneider. Post-quantum tls on embedded systems: Integrating and evaluating kyber and sphincs+ with mbed tls. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pages 841–852, 2020.
- [8] Geoff Twardokus. github code of v2verifier, Jun 13, 2022. available online: <https://github.com/twardokus/v2verifier>.
- [9] Youngbeom Kim, Jingyo Song, and Seog Chung Seo. Accelerating falcon on armv8. *IEEE Access*, 10:44446–44460, 2022.