COMP.SEC.220 Security Protocol*

Cuong Nguyen — Coursework II

15/12/2022

1 Experiment Results

In this section, I evaluate the performance of the core algorithms of AinQ. I followed exactly measurements and tried to re-generate the results in the given paper[1]. In this experiment, I implemented the protocol on my personal laptop, a standard machine which is commercially available. Both edge drones and a team leader are operated on the same device. Its specifications are shown in Table 1.

Respect to evaluate the performance of edge drones, **GenSecretValue** and **KeyRetrieval** functions are focused. On the other hand, in terms of the performance of team leader, **GenSecretValue, GenGroupKey** and **Re — Key** functions are focused.

1.1 Edge Drone

It is stated in the paper that the performance of the core functions on the resource- constrained devices depends on the number of EC point multiplications[1]. However, in the case of the standard powerful device, the difference is not obvious. My laptop executed a single EC Multiplication in approximately 0.014 seconds, the **GenSecretValue** in approximately 0.016 seconds, and the **KeyRetrival** in approximately 0.015 seconds.

*github — `https://github.com/ancuongnguyen07/SecurityProtocol`

Chipset	Apple Silicon M1
RAM	16GB
OS	MacOS Monterey 12.6.1

Table 1: My laptop specifications

	EM	Time (s)
EC Multiplication	0	0.014
GenSecretValue	1	0.016
KeyRetrieval	1	0.015

Table 2: Edge Drone Performance

1.2 Team Leader

The performance of team leader is evaluated based on the execution time of two core functions, **GenGroupKey** and **Re — Key**. Hence, I measured the execution time of the **GenGroupKey** function for a variying number of existing edge drones, ranging from 1 to 2,000. In the case of only 1 edge drone existed, **GenGroupKey** took approximately 0.044 seconds to execute , while the number of edge drones was 2,000, it took 59.5 seconds to complete. Due to the re-use of the same V parameter for all edge drones, the execution time increased in an efficient manner when I raised the number of edge drones. In more detail, multiplying 0.044 seconds by 2,000 drones lead to approximately 88 seconds. As a result, I conclude that the **GenGroupKey** function obtained about 40% more efficient than the expected performance.

Respect to the **Re — Key** algorithm, executed when an edge drone joins the group, leaves the group or the current group key expires, I performed two experiments to evaluate its performance. The first experiment focused on the key expiration. For a variying number of existing edge drones, ranging from 1 to 2,000, I measured the execution time of the **Re — Key** function. In the case of existing only 1 drone in the group, it took 0.029 seconds to execute, while the number of edge drones was 2,000, the execution time was 0.094 seconds. Figure 1 illustrates the overall execution time of both the **GenGroupKey** and **Re — Key** functions for a variying number of edge drones, ranging from 1 to 2,000.

For a second experiment, I measured the excution time of the **Re — Key** function when a variying number of new drones join a group including a variying number of existing edge drones. When a new drone joins a group of only one member, the **Re — Key** took approximately 0.028 seconds to execute, while 1,000 new drones join a group containing 1,000 existing drones, it took approximately 30.716 seconds. Table 3 shows the results of these settings.

2 Limitation and Future Work

As you can see from the above experiment results, the execution time of core functions in my implementation are much longer than those of authors' implementation. The difference partly depends on the programming language and the cryptographic library I utilized. Hence, I would try implement the protocol in Rust which has better performance than Python, the programming language

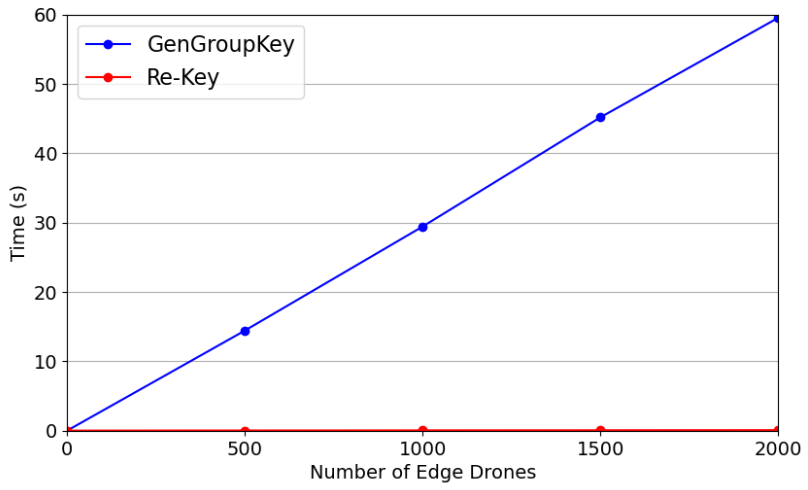

Figure 1: Performance of the Team Leader

Existing Group Members	**New Group Members**	**Time (s)**
1	1	0.028
1	100	2.919
1	500	14.463
1	1000	29.571
100	1	0.035
100	100	3.025
100	500	14.731
100	1000	29.913
500	1	0.047
500	100	2.990
500	500	15.143
500	1000	29.939
1000	1	0.064
1000	100	3.029
1000	500	15.252
1000	1000	30.716

Table 3: Group Re-key Function

I utilized in this work.

References

[1] Eugene Frimpong, Reyhaneh Rabbaninejad, and Antonis Michalas. "Arrows in a Quiver: A Secure Certificateless Group Key Distribution Protocol for Drones". In: *Secure IT Systems: 26th Nordic Conference, NordSec 2021, Virtual Event, November 29–30, 2021, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2021, pp. 31–48. ISBN: 978-3-030-91624-4. DOI: `10.1007/978-3-030-91625-1_3`. URL: `https://doi.org/10.1007/978-3-030-91625-1_3`.