Performance analysis of ETSI GS QKD 014 protocol in 5G/6G networks

Amina Tankovic[1], Patrik Burdiak[2], Emir Dervisevic[1], Miroslav Voznak[2], Miralem Mehic[1,2], and Enio Kaljic[1]

[1] Department of Telecommunications, Faculty of Electrical Engineering, University of Sarajevo, 71000 Sarajevo, Bosnia and Herzegovina,
atankovic1@etf.unsa.ba,
[2] VSB–Technical University of Ostrava, 708 00 Ostrava, Czechia

Abstract. The advancement of mobile networks driven by the introduction of 5G/6G technologies has brought more complex security challenges for end-users. Certain services within these advanced networks demand the highest level of performance and security, as any vulnerabilities could lead to critical repercussions. One promising solution to address these concerns is Quantum Key Distribution (QKD), which leverages the principles of quantum physics to offer information-theoretically secure keys. In this study, we present the implementation of a version of the key delivery protocol outlined in the ETSI GS QKD 014 standard and evaluate its performance. Following the implementation, we propose an empirically derived model of key metrics based on regression analysis of experimental results to validate the efficiency and performance of the QKD key delivery protocol. Furthermore, we conduct a comprehensive analysis of potential use cases for this protocol in the context of 5G/6G networks, showcasing its effectiveness in enhancing security.

Keywords: security, QKD, 5G/6G networks, performance, applications, quality of service

1 Introduction

Fifth generation of mobile networks has brought significant changes in terms of metrics that affect quality of service (QoS). Latencies have been reduced on the order of several milliseconds, and their exact values depend on specific group of 5G services. Minimum requirements are 4 milliseconds for eMBB (enhanced Mobile Broadband) services and 1 millisecond for URLLC (Ultra Reliable Low Latency Communications) services [1]. Rates have been increased, with theoretical peak data rate in ideal conditions of 20 Gbps and user experienced data rate of up to 100Mbps. Availability, coverage and connection density have also been increased [2]. Due to predicted exponential growth of mobile data traffic [3], 6G networks are expected to bring further improvements, which include latencies below 1 millisecond, peak rates of 1 Tbps and user data experienced rate of up to 10 Gbps. Also, compared with 5G, 6G networks will require wider coverage, more connections and more intelligence [4].

These changes will enable the usage of 5G/6G networks in various fields such as IoT/IoE (Internet of Things/Internet of Everything), AR/VR (Augmented/Virtual Reality) and AI (Artificial Intelligence) services, autonomous driving, tactile Internet, telemedicine [4–7]. Some of these services, besides good performance in terms of latencies and speeds, obviously require high level of security, because lack of it can cause critical problems. A potential solution for providing the highest possible level of security is quantum key distribution.

Quantum key distribution (QKD) is a method of establishing information-therotically secure (ITS) keys between two nodes using the laws of quantum physics [8]. Cryptographic schemes based on QKD differ from another schemes only in the part related to the distribution of keys. After the keys have been shared between two end users, they can be used for message encryption in the same way and with the same encryption algorithms as before [9].

An example of QKD network is shown in figure 1. Typical QKD network consists of a number of nodes (i.e. secure sites) which are connected by QKD links. Main components of each secure site are QKD entities, which are responsible for executing the QKD protocol, KME (Key Management Entity) that manages key generation and storage, and SAE (Secure Application Entity) which requests desired quantity of key material from KME using key delivery APIs [10]. Every secure site has at least one KME which serves one or more SAEs within that site. QKD links between nodes consist of two channels: a quantum channel that transmits key material encoded in certain photon properties, and a public channel which is used for transmission of information related to synchronization and key distillation, including sifting, error correction, verification, and privacy amplification. Because of the laws of quantum physics, eavesdropping on quantum channel and measuring quantum states of photons will result in their changes, which means that any try of eavesdropping will be detected [9]. Also, eavesdropping on public channel will not be useful for an eavesdropper, because he will not be able to recover secret keys using obtained information. In this way, QKD provides significantly higher level of security compared to other key distribution methods.

In currently developed 5G networks, QKD can be used to secure both the part from the base station to the core network and the link from the user to the base station [11]. 6G networks are expected to bring quantum-enabled communication systems [12] with many fully quantum-capable nodes, where quantum concepts in general will be very important to achieve desired performance and quality of service. QKD can be used in different parts of these systems too, e.g. to establish keys between RAN (Radio Access Network) components, like CU (Centralized Unit) and DU (Distributed Unit), in order to prevent potential eavesdropping, man-in-the-middle attacks and other threats that can affect traffic through these nodes [13], [14]. Additionally, besides high level of security, QKD can provide improvements in terms of energy efficiency since it is not based on high

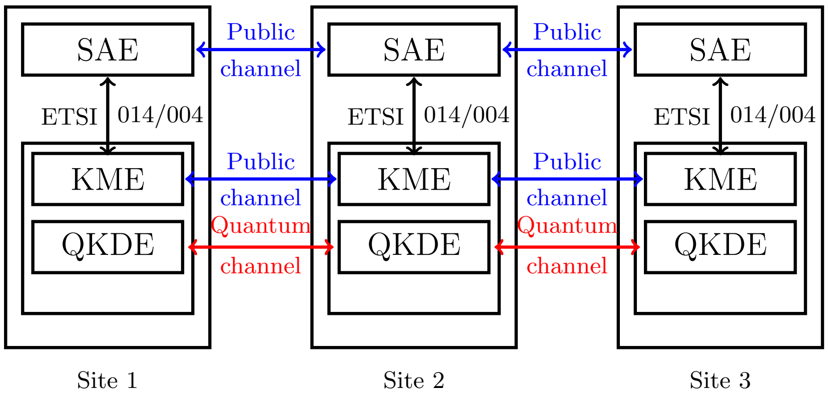


Fig. 1: Example of QKD Arhictecture

computational complexity as other cryptographic methods. Therefore, it can be an important factor towards 5G/6G green networks.

In this study, we have analysed one part of communication in QKD networks, between SAE and KME. This communication is defined through key delivery API in two group specifications, ETSI GS QKD 004 [15] and ETSI GS QKD 014 [16]. In this paper, focus is on ETSI GS QKD 014. This specification defines format of HTTPS requests and responses that SAE and KME exchange. Using these requests, SAE can get status information for its communication pair and desired quantity of key material (generated by QKD and stored on KME) for encryption messages between them.

The rest of this paper is organised as follows. In section II is given related work including different implementations and testbeds for ETSI GS QKD 014 standard. In section III, we have described our implementation of ETSI GS QKD 014 and some initial test scenarios that we used to estimate unknown coefficients of mathematical model. Section IV gives mathematical model of two important metrics for 5G/6G networks, latency and efficiency. Section V gives results for several testing scenarios considering potential use cases and applications. At the end, there is a conclusion and directions for future work.

2 Related Work

A significant number of studies analyse key delivery protocol in different test environments and for different purposes. Some of them include implementation of protocol using ETSI GS QKD 014 standard, showing importance of using ETSI QKD standards for compatibility reasons.

Authors in [17] implemented MACsec protocol on MACsec nodes, that are set up on the FSP150 ProVMe edge device, composed of a field-programmable gate array (FPGA) and a Linux host. Keys are delivered from an external QKD key supplier to nodes via interface defined in ETSI GS QKD 014. Experiments in this testbed have shown that proposed protocol can be performed with acceptable latency in Ethernet networks.

In [18] authors presented an experimental setup that consists of QKD equipment, servers that represent end users, a switch and encryptors which obtain keys using API defined in ETSI GS QKD 014. Several scenarios that include different combinations of previously mentioned equipment are analysed and impact of adding new elements on network performance is discussed. It is shown that QKD encryption, which adds some overhead and causes additional delay, has small impact on total latency and throughput.

Authors in [19] demonstrated a practical key management scheme for a QKD network. System is composed of commercial, multi-vendor devices, and communication between them is implemented implemented using ETSI GS QKD 014 standard. SDN controller is used for dynamically key relay routing. Communication with SDN controller is defined in ETSI GS QKD 015 standard. Since this solution is based on the standard interface and uses multi-vendor devices, it can provide simple integration of QKD network with standard telecommunication network.

A digital QKD (D-QKD) platform based on quantum permutation pad (QPP) is introduced in [20]. It offers distribution of quantum keys and quantum entropy, generated from physical quantum random number generators. D-QKD interfaces have been developed to support ETSI GS QKD 014 standard, which makes this platform compatibile with any other systems that use the same standard.

Authors in [21] have presented simulation of denial of service attacks against KMEs. Simulations were performed using QKDNetSim module developed in ns-3 simulator. Communication between end-user applications and their KMEs are implemented using ETSI GS QKD 014. Malicious applications, as simplified end-user applications, send invalid requests to KMEs at regular intervals which are predefined to the desired values. This paper has shown that KME must pay attention to number of requests that applications generate, and if it detects unusual behavior, it must temporarily deny access to QKD network for these applications.

3 Protocol implementation

ETSI GS QKD 014 defines a REST-based API for key delivery from QKD network to applications. This API is used between two entities: SAE, which makes HTTPS requests in order to get keys and status information for its communication pair, and KME which acts as a server, performs SAE identity check and sends HTTPS responses.

Our solution for KME was implemented using CppServer[1], an open-source C++ library which has support for HTTPS protocol and provides usage of SSL certificates and main HTTP request methods (GET, POST, PUT, DELETE). The class diagram for implemented solution is shown in figure 2.

[1] https://github.com/chronoxor/CppServer

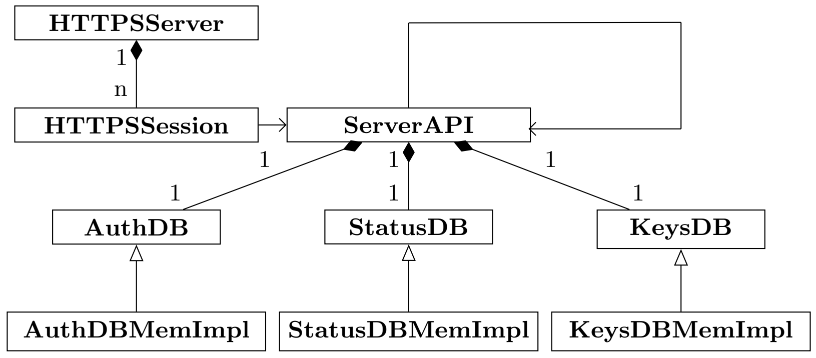


Fig. 2: UML class diagram for implemented KME solution

We have introduced three databases that will be maintained by KME. Authentication database (AuthDB) contains SAE IDs and their X.509 public keys. Status database (StatusDB) contains pairs of master-slave SAE IDs, status information and key material shared between master and slave SAE entities. Based on data from these databases, the server performs authentication and access control and provides keys and status information. The third database (KeysDB) stores all previously generated keys with their IDs and the IDs of master and slave SAEs that share each of these keys. This database is used when SAE requests specific key using its ID. These databases are implemented as abstract classes (for compatibility reasons with future implementations), and their implementations are inherited from these classes.

Three methods defined in protocol specification are implemented in class *ServerAPI* as follows:

– *Get status* - KME checks if SAE_ID of caller exists as master SAE_ID in StatusDB. If it does, KME checks if its slave SAE_ID is same as slave ID requested in URL. If that is also satisfied, KME sends status information in JSON format.
– *Get key* - KME checks if SAE_ID of caller exists as master SAE_ID in StatusDB and if its slave SAE_ID is same as slave ID requested in URL, but also checks if requested size of key is multiple of 8, and if there is enough quantity of key material for requested parameters. If everything is satisfied, server sends response in JSON format to SAE. The response contains requested number of keys (encoded to Base64 format) and unique ID for each of them. These keys with their IDs are then written in KeysDB, and key material that was used for generating keys is deleted from database.
– *Get key with key IDs* - KME checks if there is a pair in StatusDB which contains SAE_ID of caller as slave, and SAE_ID from URL as master. After that, if requested key ID exists in KeysDB, KME sends the key with its ID to SAE and deletes it from KeysDB database.

These methods are implemented as thread-safe to support consistent key management in a multithreading environment.

A computer on which KME is implemented is connected with 100Mbps Ethernet link to another one which serves as a SAE. Specificiations of these computers are given in table 1. Grafana k6[2], an open-source load testing tool, is used to emulate clients. This tool provides reliability and performance testing using simple scripts written in JavaScript. Testing was performed for various scenarios, that include every combination of number and size that are listed in table 2. Values for size parameter (in number of bits) are chosen according to the key sizes of the most used encryption algorithms (e.g. AES-128, AES-256, ECC P-384 [22], [23]) and some specific values of packet sizes (e.g. MTU [24] and jumbo frame size) to cover OTP (one-time pad) encryption. Values for number parameter are chosen arbitrarily to get different quantities of key material, but also to analyse whether is better to request one larger or several smaller keys in same request. To get representative sample, every scenario was performed 1000 times.

Table 1: Specifications of used devices

(a) KME

Component	Specification
Processor	Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz
Memory	32GB DDR3, dual-channel

(b) SAE

Component	Specification
Processor	Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz
Memory	16GB DDR4, dual-channel

Table 2: Simulation parameters

Parameter	Value
Key size	128, 256, 384, 512, 1280, 4608, 12000, 64000
Number of keys	1, 2, 3...50, 100, 150...950
Number of iterations	1000

Experimental results show that latency of key delivery is increasing linearly, both with number of keys and size of them (figure 3). Also, it is shown that better performance can be achieved in scenarios when SAE requests one larger instead of several smaller keys. These results are used in next sections, in modelling important metrics and use-case specific scenarios.

[2] https://k6.io/open-source/

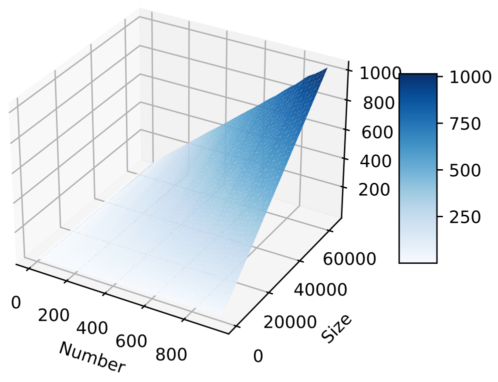


Fig. 3: Measured latency of key delivery

4 Mathematical model of key metrics

In this section, latency and efficiency are described using mathematical equations with precisely defined parameters based on known network procedures and sizes with values estimated from experimental results using regression or averaging. Symbols that are used in equations with their descriptions are given in table 3.

Table 3: List of used symbols with their descriptions

Symbol	Description
N	Number of requested keys
S	Size of requested keys
T_{TCP}	Duration of TCP handshake
T_{TLS}	Duration of TLS handshake
T_S	Time spent sending HTTP request
T_W	Time spent waiting for HTTP response
T_R	Time spent receiving HTTP response
T	Total latency of key delivery
N_A	Number of ACK packets
N_D	Number of data packets
P	Size of HTTP response payload
B	Total number of exchanged bytes
T_{TD}	Transport delay for one data packet
T_{PD}	Processing delay for one data packet
T_{TA}	Transport delay for one ACK packet
T_{PA}	Processing delay for one ACK packet
η	Efficiency

4.1 Latency

MSC diagram of communication between SAE and KME is shown in figure 4. Five main parts that can be noticed on the diagram are typical procedures related to HTTPS communication: TCP handshake, TLS handshake, sending request, waiting for response and receiving response. According to this, total latency for delivering keys from KME to SAE can be written as follows:

$$T = T_{TCP} + T_{TLS} + T_S + T_W + T_R \tag{1}$$

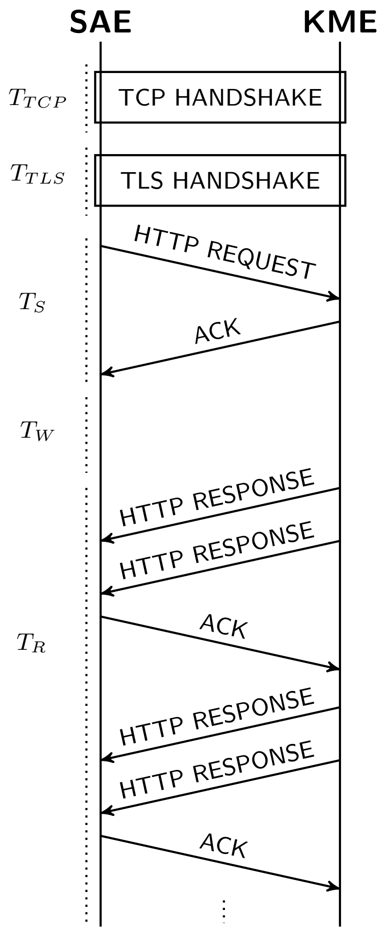


Fig. 4: MSC diagram of communication between SAE and KME

Communication begins with establishing connection and authentication procedure through TCP and TLS handshakes. These handshakes have approximately same duration, regardless of testing scenario. Thus, their values can be estimated from experimental results using ensemble averaging:

$$T_{TCP} = 1.16\,\mathrm{ms} \tag{2}$$

$$T_{TLS} = 20.194\,\mathrm{ms} \tag{3}$$

It is important to emphasize that this is initial latency that occurs only at the beginning of session. Once after TCP connection is established, and client has confirmed his identity with certificate, he can send as much requests as he wants until the end of session. Latency of key delivery then depends only on latencies related to HTTP protocol.

After initial handshakes, HTTP request is sent to KME. Time spent sending request depends on lengths of request's body and URL. Maximum difference between these lengths for different scenarios is only few bytes, and differences between latencies for these cases are negligible. Thus, duration of sending HTTP request from SAE to KME can be estimated from experimental results using ensemble averaging again:

$$T_S = 0.018\,56\,\mathrm{ms} \tag{4}$$

Time spent waiting for response from KME is time that server needs for searching databases and preparing the answer, so it can be concluded that it depends on the quantity of key material that SAE requests. Experimental results show that it grows linearly (figure 5), and using linear regression, estimated slope has following value:

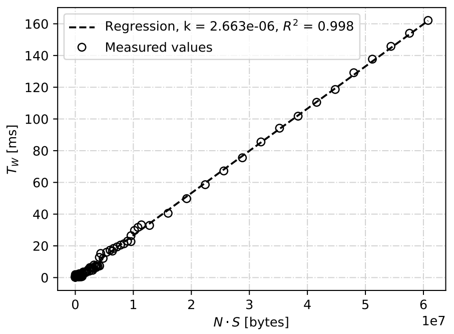
$$k_W = 2.663 \cdot 10^{-6}\ \mathrm{ms/bit} \tag{5}$$

Now, waiting latency (in ms) can be written as:

$$T_W = k_W \cdot N \cdot S \tag{6}$$

Last component of total latency, time spent receiving HTTPS response from KME, depends on total number of packets that will be exchanged between KME and SAE. KME sends data packets that contain fragments of requested key material, and SAE sends ACK packets after a certain number of data packets or after a certain time interval defined in [25]. This means that relation between number of data and ACK packets can be written as:

$$N_A = k \cdot N_D \tag{7}$$



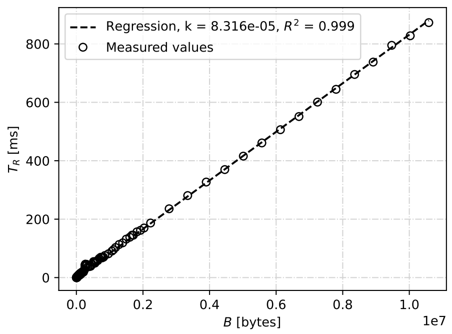


Fig. 5: Linear regression for waiting and receiving latency

Coefficient k is usually around 0.5, but it cannot be precisely defined because it depends on both conditions that are previously mentioned.

Total number of bytes in data packets that are sent from KME to SAE include headers on all network layers and overhead of TLS and TCP fragmentation [24], [26–29]. Size of response payload (in bytes) can be calculated by following equation:

$$P = 10 + 59 \cdot N + 4 \cdot N \cdot \left\lceil \frac{S}{24} \right\rceil \tag{8}$$

Previous equation was written considering JSON formatting and Base64 encoding. Total number of bytes that should be sent on the wire from KME to SAE is then:

$$\begin{aligned} B = L_{HTTP} + (L_{IV} + L_{TLS}) \cdot \left\lceil \frac{P}{TLS_{MSS}} \right\rceil \\ + (L_{MAC} + L_{IP} + L_{TCP}) \cdot \left\lceil \frac{P}{TCP_{MSS}} \right\rceil + P \end{aligned} \tag{9}$$

Parameters from previous equation with their descriptions and exact values are given in table 4. All values are expressed in bytes.

Receiving latency includes transport and processing latency for all packets that need to be sent between KME and SAE. Therefore, T_R can be written as:

$$T_R = N_D \cdot (T_{TD} + T_{PD}) + N_A \cdot (T_{TA} + T_{PA}) \tag{10}$$

Transport latencies for data and ACK packets can be calculated using sizes of these packets and speed of Ethernet link that connects KME and SAE. Processing latencies cannot be calculated precisely, because they depend on network parameters that cannot be predicted, so they will be estimated from test results using regression.

Due to link speed, transport latencies are negligible in experimental results, so they can be ignored in previous equation. Considering this and equation 7, equation 10 can be approximated with following:

$$T_R \approx N_D \cdot T_{PD} + N_A \cdot T_{PA} = N_D \cdot (T_{PD} + k \cdot T_{PA}) \tag{11}$$

This equation can be written in similar way using number of bytes B as parameter instead of number of data packets:

$$T_R = B \cdot (k_D + k \cdot k_A) \tag{12}$$

where k_D and k_A denote processing coefficients expressed in $\frac{ms}{B}$, which are estimated from test results using regression. Time spent receiving request increases linearly with number of bytes in data packets, which is shown in figure 5. Estimated slope (written as k_R for simplicity) has following value:

$$k_D + k \cdot k_A = k_R = 8.316 \cdot 10^{-5} \text{ ms/B} \tag{13}$$

Finally, receiving latency (in ms) can be written as

$$T_R = k_R \cdot B \tag{14}$$

Summary of all parameters used in previous equations with their exact or estimated values are given in table 4. Estimated values are related to described implementation and testbed and they are not applicable to other environments.

4.2 Efficiency

Efficiency of key delivery is expressed as ratio of quantity of key material that SAE requests and total number of bytes that needs to be sent over network:

$$\eta = \frac{N \cdot S}{B \cdot 8} \tag{15}$$

Efficiency is better for a more extensive quantity of keys or larger keys per request because of lower overhead. For each key size, there is a limit where efficiency is the highest, and further improvements with larger quantity of requested key cannot be achieved. Also, it can be seen that even the highest achieved efficiency is very low for short keys. Therefore, it is better to request larger quantity of key material and process it later, to obtain keys of desired size for specific encryption algorithm.

Table 4: Parameters with their descriptions and exact/estimated values

Parameter	Description	Exact value	Estimated value
T_{TCP}	TCP connection setup latency	-	1.16 ms
T_{TLS}	TLS handshake latency	-	20.194 ms
T_S	Sending HTTP request time	-	0.01856 ms
L_{HTTP}	HTTP response header length	84 + log(P)	-
L_{IV}	Nonce/IV + Auth tag length	24	-
L_{TLS}	TLS header length	5	-
TLS_{MSS}	TLS maximum segment size	16408	-
L_{MAC}	MAC header length	14	-
L_{IP}	IP header length	20	-
L_{TCP}	TCP header length	20	-
TCP_{MSS}	TCP maximum segment size	1460	-
k_W	Slope coefficient of waiting latency	-	$2.663 \cdot 10^{-6} ms/bit$
k_R	Slope coefficient of receiving latency	-	$8.316 \cdot 10^{-5} ms/B$

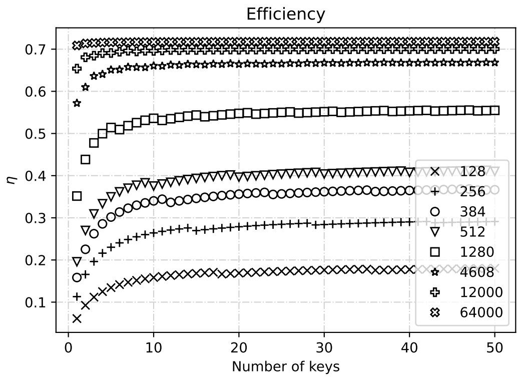


Fig. 6: Efficiency of key delivery

Efficiency of key delivery can also be used as a measure of additional bandwidth required for key transmission:

$$BW_{key} = \frac{BW_{app}}{\eta} \tag{16}$$

where BW_{app} is required bandwidth for specific application.

After analysis of considered metrics, it can be concluded that there is no single answer to questions about the quantity of key material which should be requested and number of requests that should be sent to obtain that quantity. It depends on specific application requirements, is it delay-sensitive or does it have more strict requirements in terms of efficiency. In next section, several scenarios with different requirements are analysed.

5 Potential use cases

All applications in mobile networks include some kind of voice, video or data transfer. In this section, these types of traffic are analysed considering some use cases for 5G/6G networks specified in [30]. Since it was shown that efficiency is low for short keys, we have considered OTP encryption where quantity of key material that SAE requests is equal to quantity of data that will be transmitted in one şession. A key for one session can be obtained completely in one request, or through fragments in several requests. Therefore, we have considered different fragmentation levels, i.e. different number of requests for each quantity of key material, to analyse latency and efficiency and find minimal value that satisfies latency requirements of specific application.

In order to analyse voice traffic, we have chosen a call duration of 202 seconds, EVS-WB codec at 24.4 kbps with maximum allowable latency of 20 ms and 50% voice activity factor [30].

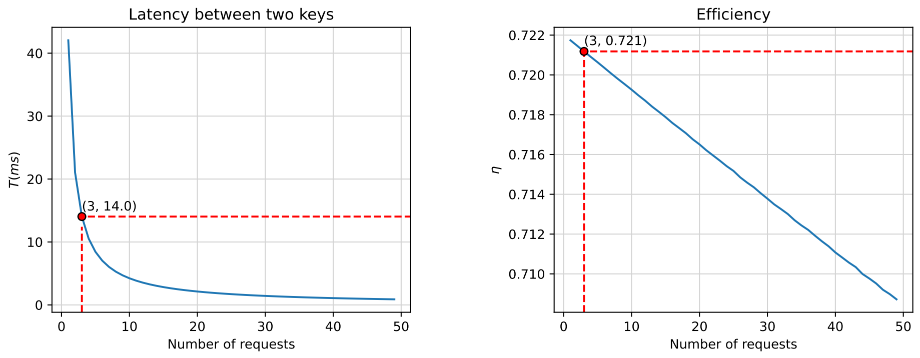


Fig. 7: Use case: Voice traffic

It can be seen in figure 7 that scenario with 3 requests (with same size of key in each of them) will satisfy latency requirements for selected codec and requested quantity of key material. Every scenario with larger number of requests will result in lower latency between two keys, but also in lower efficiency. However, decrease in efficiency for larger number of requests is not significant, since requested quantity of key is pretty huge.

For real-time video traffic, one-hour video conference with 1.5 Mbps CBR was considered [30]. Latency between two keys must be below 36 ms. Results in figure 8 show that in this case, SAE needs to send a larger number of requests compared to previous case. These results were expected since the quantity of key material for a video conference is larger and latency requirements are still very strict.

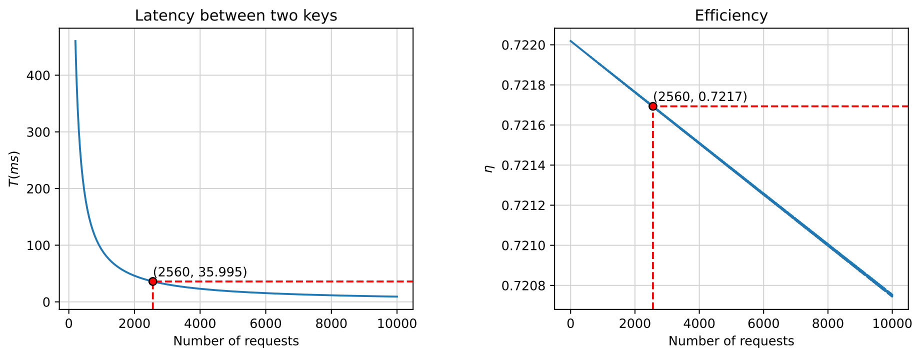


Fig. 8: Use case: Real-time video traffic (RT VT)

Non-real time video traffic has significantly more relaxed requirements in terms of latency, since inter packet delay for this case is 1 s. For one 35-minute episode of some series [31], SAE needs to send large number of requests for key material again. Since requested quantity of data is huge again, efficiency is still very good, and decrease with number of requests is even smaller.

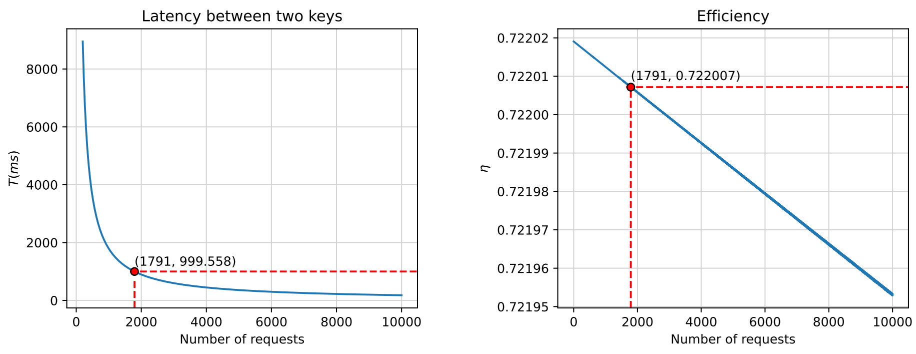


Fig. 9: Use case: Non-real time video traffic (NRT VT)

Latency and efficiency results for one-hour online gaming session, as an example of BUD (Bursty User-Driven) traffic, are shown in figure 10. Latency requirements are strict (58 ms, calculated using largest extreme value distribution given in [30]) in order to achieve interactivity, and efficiency is still high.

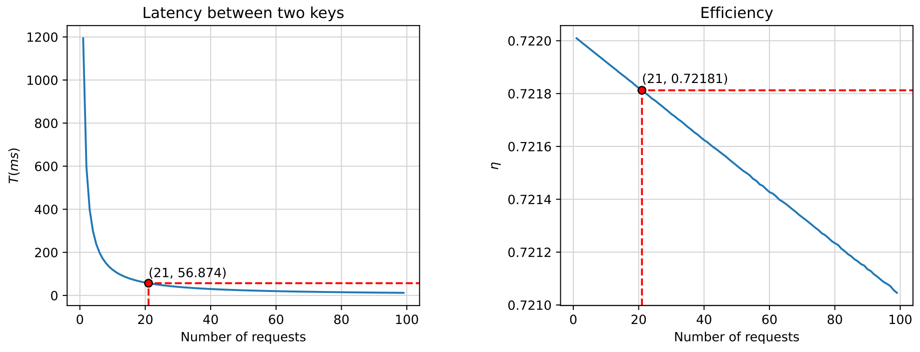


Fig. 10: Use case: Gaming traffic

BAD (Bursty Application-Driven) traffic include M2M services like smart metering, environment monitoring and control, telemedicine, home automation etc. All of these services include periodical sessions during the day, and in each of these sessions hundreds of kilobytes can be sent, which is smaller quantity of data compared to previous cases. Considering upper bound for transmitted quantity of data in one telemedicine session and proposed latency requirements in [30], it can be shown (figure 11) that key for one whole session can be obtained in one request. Larger number of requests will bring lower latencies and negligible decrease in efficiency.

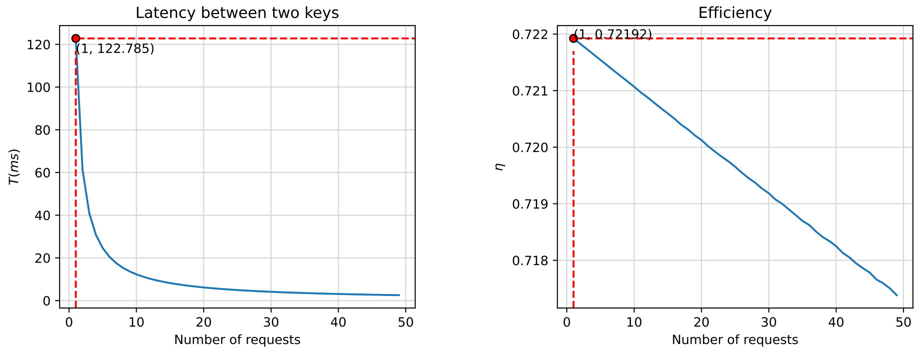


Fig. 11: Use case: BAD traffic

6 Conclusion

In this paper, we have presented our analysis of ETSI GS QKD 014 key delivery protocol - its implementation, mathematical model of important metrics and test scenarios. Initial test results have confirmed the assumption that latency of key delivery (without considering latency due to communication between KMEs) grows linearly with both number and size of requested key. These results were used for estimating coefficients of mathematical model, such as slopes for latencies that depend on requested quantity of key material (time spent waiting for and receiving the HTTP response), but also average values of latencies that do not depend on this quantity (initial handshakes and latency of sending HTTP request). Estimations were performed using linear regression and ensemble averaging. Additionally, it was shown that efficiency of key delivery is higher for more extensive quantities of key material and larger key sizes, but also that there is a limit after which the efficiency cannot be further improved. After completing the model, several tests were performed to see whether ETSI GS QKD 014 can be used in 5G/6G applications. These tests have shown how many key requests should be sent to satisfy latency requirements for different applications and provide good quality of service.

Future work may include further analysis and potential improvements of the implemented solution. It should be analysed whether and how performance of the implemented server can be improved (e.g. using better memory management, equipment with better performance where KME and SAE are implemented, or using additional hardware for offloading some of protocol functions). Another protocol for key delivery, ETSI GS QKD 004, should also be analysed and compared with implemented protocol. Since the observed metrics were modelled for scenario with one user, the impact of additional users should be considered in order to analyse the possibility of using the implemented solution in multithreading applications.

Further development of mobile networks will bring new applications with more strict requirements in terms of all QoS metrics, but also in terms of security of communication. Therefore, it is important to continue research in security and in that way support development of new, smart applications that will change and improve human life.

Acknowledgements

The research leading to the published results was supported under the NATO SPS G894 project "Quantum Cybersecurity in 5G Networks (QUANTUM5)" and in part by the Ministry of the Interior of the Czech Republic within the project Network Cybersecurity in Post-Quantum Era under Grant VJ01010008; in part by the Ministry of Science, Higher Education and Youth of Canton Sarajevo, Bosnia and Herzegovina, under the project DQKDNM 2023; in part by Federal Ministry of Education and Science, Bosnia and Herzegovina, under Grant 05-35-2140-1/22; in part by the Ministry of Science, Higher Education and Youth of Canton Sarajevo, Bosnia and Herzegovina, under Grant 27-02-35-35137-30/22.

References

1. ITU-R "Minimum requirements related to technical performance for IMT-2020 radio interface(s),", 2017.
2. Shafi M., Molisch A. F., Smith P. J., Haustein T., Zhu P., De Silva P., Tufvesson F., Benjebbour A., Wunder G., "5G: A Tutorial Overview of Standards, Trials, Challenges, Deployment and Practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201 – 1221, 2017. doi:10.1109/JSAC.2017.2692307
3. Tariq F., Khandaker M. R. A., Wong K., Imran M. A., Bennis M., Debbah M., "A Speculative Study on 6G," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 118 – 125, 2020. doi:10.1109/MWC.001.1900488
4. Wang C.-X., You X., Gao X., Zhu X., Li Z., Zhang C., Wang H., Huang Y., Chen Y., Haas H., Thompson J. S., Larsson E. G., Di Renzo M., Tong W., Zhu P., Shen X., Poor H. V., Hanzo L., "On the Road to 6G: Visions, Requirements, Key Technologies, and Testbeds," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 905–974, 2023. doi:10.1109/COMST.2023.3249835
5. Akpakwu G. A., Silva B., Hancke G. P., Abu-Mahfouz A. M., "A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges," *IEEE Access*, vol. 6, pp. 3619 – 3647, 2017. doi:10.1109/ACCESS.2017.2779844
6. Nguyen D. C., Ding M., Pathirana P. N., Seneviratne A., Li J., Niyato D., Dobre O., Poor H. V., "6G Internet of Things: A Comprehensive Survey," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 359–383, 2022. doi:10.1109/JIOT.2021.3103320
7. Soos G., Ficzere D., Varga P., "Towards Traffic Identification and Modeling for 5G Application Use-Cases," *Electronics*, vol. 9, no. 4, 2020. doi:10.3390/electronics9040640
8. Mehic M., Maurhart O., Rass S., Komosny D., Rezac F., Voznak M., "Analysis of the Public Channel of Quantum Key Distribution Link," *IEEE Journal of Quantum Electronics*, vol. 53, no. 5, 2017. doi:10.1109/JQE.2017.2740426
9. Cao Y., Zhao Y., Wang Q., Zhang J., Ng S. X., Hanzo L., "The Evolution of Quantum Key Distribution Networks: On the Road to the Qinternet," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 839 – 894, 2022. doi:10.1109/COMST.2022.3144219
10. Tsai C.-W., Yang C.-W., Lin J., Chang Y.-C., Chang R.-S., "Quantum Key Distribution Networks: Challenges and Future Research Issues in Security," *Applied Sciences*, vol. 11, no. 9, 2021. doi:10.3390/app11093767
11. M. Mehic, L. Michalek, E. Dervisevic, P. Burdiak, M. Plakalovic, J. Rozhon, N. Mahovac, F. Richter, E. Kaljic, F. Lauterbach, P. Njemcevic, A. Maric, M. Hamza, P. Fazio, and M. Voznak, "Quantum cryptography in 5G networks: A comprehensive overview," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2023. doi:10.1109/COMST.2023.3309051
12. Muheidat, F., Dajani, K., Lo'ai, A.T., "Security Concerns for 5G/6G Mobile Network Technology and Quantum Communication," *Procedia Computer Science*, vol. 203, pp. 32–40, 2022. doi:10.1016/j.procs.2022.07.007
13. Wang C., Rahman A., "Quantum-Enabled 6G Wireless Networks: Opportunities and Challenges," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 58–69, 2022. doi:10.1109/MWC.006.00340
14. Ali M. Z., Abohmra A., Usman M., Zahid A., Heidari H., Imran M. A., Abbasi Q. H., "Quantum for 6G communication: A perspective," *IET Quantum Communication*, 2023. doi:10.1049/qtc2.12060

15. ETSI GS QKD 004, "Quantum Key Distribution (QKD); Application Interface," V2.1.1, 2020.
16. ETSI GS QKD 014, "Quantum Key Distribution (QKD); Protocol and data format of REST-based key delivery API," V1.1.1, 2019.
17. Cho J. Y., Sergeev A., "Using QKD in MACsec for secure Ethernet networks," *IET Quantum Communication*, 2021. doi:10.1049/qtc2.12006
18. Stan C., Garcia C. R., Cimoli B., Olmos J. J. V., Monroy I. T., Rommel S., "Securing communication with quantum key distribution: Implications and impact on network performance," in *Optica Advanced Photonics Congress*, 2022. doi:10.1364/SPPCOM.2022.SpW2J.2
19. Cho J. Y., Pedreno-Manresa J.-J., Patri S., Sergeev A., Elbers J.-P., Griesser H., White C., Lord A., "Demonstration of software-defined key management for quantum key distribution network," in *Optical Fiber Communications Conference and Exhibition*, 2021.
20. Lou D., He A., Redding M., Geitz M., Toth R., Doring R., Carson R., Kuang R., "Benchmark Performance of Digital QKD Platform Using Quantum Permutation Pad," *IEEE Access*, vol. 10, pp. 107 066 – 107 076, 2022. doi:10.1109/ACCESS.2022.3212738
21. Mehic M., Rass S., Dervisevic E., Voznak M., "Tackling Denial of Service Attacks on Key Management in Software-Defined Quantum Key Distribution Networks," *IEEE Access*, vol. 10, pp. 110 512 – 110 520, 2022. doi:10.1109/ACCESS.2022.3214511
22. Nguyen V.-L., Lin P.-C., Cheng B.-C., Hwang R.-H., Lin Y.-D., "Security and Privacy for 6G: A Survey on Prospective Technologies and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2384 – 2428, 2021. doi:10.1109/COMST.2021.3108618
23. Chorti A., Barreto A. N., Kopsell S., Zoli M., Chafii M., Sehier P., Fettweis G., Poor H. V., "Context-Aware Security for 6G Wireless: The Role of Physical Layer Security," *IEEE Communications Standards Magazine*, vol. 6, no. 1, pp. 102 – 108, 2022. doi:10.1109/MCOMSTD.0001.2000082
24. McCloghrie K., Mogul J., Kent C. A., Partridge C, "IP MTU discovery options," RFC 1063, 1988. doi:10.17487/RFC1063
25. Braden R. T., "Requirements for Internet Hosts - Communication Layers," RFC 1122, 1989. doi:10.17487/RFC1122
26. "Internet Protocol," RFC 791, 1981. doi:10.17487/RFC0791
27. Rescorla E., Dierks T., "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, 2008. doi:10.17487/RFC5246
28. Thomson M., "Record Size Limit Extension for TLS," RFC 8449, 2018.
29. Eddy W., "Transmission Control Protocol (TCP)," RFC 9293, 2022. doi:10.17487/RFC9293
30. Navarro-Ortiz J., Romero-Diaz P., Sendra S., Ameigeiras P., Ramos-Munoz J.-J., Lopez-Soler J. M., "A Survey on 5G Usage Scenarios and Traffic Models," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 905 – 929, 2020. doi:10.1109/COMST.2020.2971781
31. Claeys M., Bouten N., De Vleeschauwer D., Van Leekwijck W., Latre S., De Turck F., "Cooperative Announcement-Based Caching for Video-on-Demand Streaming," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 308 – 321, 2016. doi:10.1109/TNSM.2016.2546459