Benchmarking Scalability and Security Configuration Impact for A Distributed Sensors-Server IOT Use Case

Robert KALASKA
Gdansk, Poland, robert.kalaska@gmail.com

Pawel CZARNUL
Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology
Narutowicza 11/12, 80-392, Gdansk, Poland
pczarnul@eti.pg.edu.pl

Abstract

Internet of Things has been getting more and more attention and found numerous practical applications. Especially important in this context are performance, security and ability to cope with failures. Especially crucial is to find good trade-off between these. In this article we present results of practical tests with multiple clients representing sensors sending notifications to an IoT middleware – DeviceHive. We investigate performance under different loads by various numbers of IoT clients and show how various strengths of the hash algorithm affect the performance measured by the number of client requests handled within a predefined time frame.

Keywords: IoT security · JWT hash strength · IoT middleware performance.

Introduction

In the literature we can find many examples concerning security of IoT middlewares and whole systems for particular applications. Many comparisons of middlewares were performed such as (Fremantle & Scott, 2017) in which authors proposed a matrix of security challenges and evaluated middlewares using this matrix, (Kałaska & Czarnul, 2020) where IoT solutions were compared in terms of their security features or (Diaz-López, et al., 2018) which also evaluates IoT middlewares in terms of selected security criteria. We can also find works related to particular applications such as (Celesti, et al., 2019) which addresses an e-health scenario based on the FIWARE architecture (FIWARE , 2021) along with security issues. There are many different approaches to achieve secure and rapid platforms such as SGXIoTGUARD proposed in (Ayoade, et al., 2019), another one is FIWARE (FIWARE , 2021) or one of open source platform called DeviceHive (DeviceHive , 2021). In many of these a JSON Web Token (JWT) is a desirable mechanism for asserting claims concerning participants in IoT scenarios (Datta & Bonnet, 2019), (Ahmed & Mahmood, 2019) or (Solapurkar, 2016). In work (Shingala, 2019) the author analyses usage of JWT based client authentication in Message Queuing Telemetry Transport, compared to username/password as well as TLS Client Authentication, in the context of, in particular, confidentiality, client privacy, credential management, interruption of session, cost on the client side. In this paper we decided to check how different security setups may influence application performance. In particular, the paper evaluates system scalability and performance under various loads for multiple IoT devices reporting data to a central server and if there are visible differences between using different hash functions in JWS under a selected load and key settings.

Related work

In the literature, we can find examples of research works which focus on performance and scalability of IoT systems. In (daCruz, et al., 2018) authors compared many platforms exchanging data with RESTful methods, while the parameters were packet size and the number of concurrent users. Measured values were response time and error percentage. One of very important observations was that attention should be paid to data sent between devices as presence of white spaces or unnecessary characters may lead to extended processing time and system load. Authors also stated that it is important to provide high-quality documentation for platforms. They distinguished the SiteWhere platform in their comparison (SiteWhere, 2021) where it has the best results in their comparison (response time and error rate), especially under high load (10 000 concurrent users). Article (Ismail, et al., 2018) compares two middlewares: ThingsBoard and SiteWhere. The evaluation metrics are scalability and stability while the measures were taken for two protocols HTTP and MQTT. Authors checked how both systems acted under various loads (ending with load of 1000 publishers) and also compared how the number of parameters in request influenced the throughput. Results of their survey provide us with information of better

Cite this Article as: Robert KALASKA and Pawel CZARNUL " Benchmarking Scalability and Security Configuration Impact for A Distributed Sensors-Server IOT Use Case" Proceedings of the 37th International Business Information Management Association (IBIMA), 30-31 May 2021, Cordoba, Spain, ISBN: 978-0-9998551-6-4, ISSN: 2767-9640

performance for HTTP protocol on ThingsBoard, while MQTT has better performance with SiteWhere, but it also has a high error rate. Another work is (Kokkonis, et al., 2018) where authors provide a survey on different protocols such as HTTP, SOAP, MQTT and CoAP under different payload and load (as number of clients). The measured value was round-trip time which shows the performance of each protocol. In that survey the authors showed that the SOAP protocol does not fit the IoT use case as round trip times (RTTs) become very long for larger sized payloads (tested 25-1000 bytes) and larger numbers of concurrent users giving e.g. the RTT of around 115 seconds with 1000 bytes payload and 50 concurrent users. Another observation was that HTTP has best performance for small load and payload, while MQTT is better under high load and medium-big payloads. Moreover CoAP, which is a UDP based protocol, has better results than MQTT for high load and also low load with big payloads. These works provide a really thorough view on performance and scalability of different protocols on various platforms under different conditions.

Moreover, there are also surveys based on benchmarking computations of different HMAC algorithms like (Rashwan, et al., 2012). As far as the research on performance of JSON Web Token is concerned, in (Rahmatulloh, et al., 2019) authors compared JWT in the context of the most optimal setup of signing algorithms and found out that HMAC compared to ECDSA and RSA is the fastest one. All of provided surveys are thorough concerning their assumptions and provide interesting conclusions but we could not find any survey which take a look into performance and scalability under different security conditions (especially JWT). That took us to take a deeper look into how different settings of JWT may affect total performance and scalability of the system in an IoT scenario. Moreover, while (Rahmatulloh, et al., 2019) shows results on differences between HMAC, RSA and ECDSA in JWT in our paper we decided to compare differences only between variants of hash options within the HMAC algorithm.

Benchmarking Environment and Procedure

In this section we describe a benchmarking environment and measurement methods along with evaluated metrics.

Test Application Model

Internet of Things can be used in many different application scenarios. One of these is the model of a sensor network sending notifications to a server periodically and continuously. Such a model can be compared to solutions presented e.g. in (Xiaojun, et al., 2015) and also (Vijayakumar & Ramya, 2015). In the first one authors proposed a sensor network composed of many independent sensors which measure the pollution level and send information to a central server. In the second one an IoT network is built with many independent Raspberry PI platforms connected with sensors to measure water parameters, each Raspberry node sends its data to a central gateway which is responsible for data analysis. Both models can be presented as many independent devices sending notifications to a central repository. In our experiment we follow such an application model which is depicted in Fig 1. Each device sends notification via HTTP protocol with TLS encryption using REST model. JWT token is included in each request.

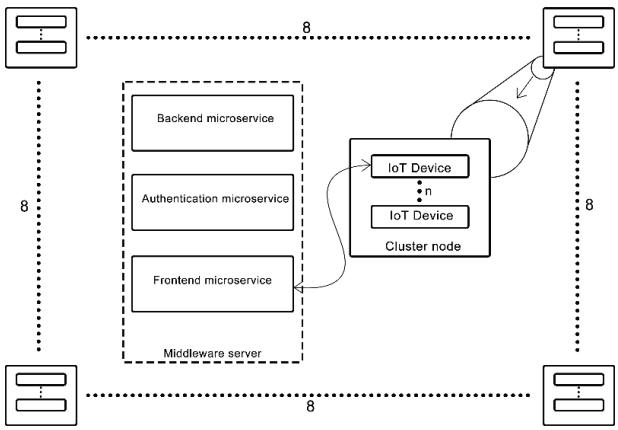

Fig 1. Application model and environment.

IOT Middleware Platform

For the analysis we chose the DeviceHive IoT middleware (DeviceHive , 2021) which is an open source Java IoT platform. It is composed of three main microservices: authentication, back-end and front-end service. The platform provides capability to work with three protocols (MQTT, HTTP and WebSocket) and there is a possibility to integrate other protocols as plugins. The above features, well documented APIs and open source code made us chose this solution. Moreover, this project is based

on maven artifacts, so while our goal was to have a convenient possibility to change inner libraries to measure time and performance, it was a viable choice.

Hardware Environment

While we decided to simulate an IoT network of many independent devices we used a cluster built of 33 nodes. 32 simulated IoT devices while one was used to deploy the middleware platform receiving requests from clients. Each node featured an Intel Xeon CPU E5345 @ 2.33 GHz CPU with 8GB RAM and CentOS 6. All computers were connected with Gigabit Ethernet.

IoT Device Simulator

On each of the 32 nodes we ran an IoT device simulator multithreaded Java application. Each thread was simulating one IoT device. It gave us the opportunity to simulate many more than only 32 IoT devices. Thus, the total device count is $N = 32 * th$ where th is the number of threads per node.

Such an approach is typically used also in testing scalability of Data Acquisition Systems such as shown in (Czarnul, et al., 2020) which benchmarks a use case in which multiple multithreaded client processes perform writes and reads associated with key/value pairs to multiple multithreaded server processes in a cluster environment.

Each thread is running same code. In the beginning it retrieves JWT tokens (one for operation and one for renewal) from the platform, then starts a loop checking periodically if the operation token is still valid (the token has the life time of 30 seconds), if not it retrieves a new token (based on the renewal token). It continuously sends notification to the middleware platform. The notification is a randomly generated UUID string which simulates data sent from a device. The size of data is 36 bytes which is sufficient to include information from 9 sensors each of 4 bytes length.

Moreover, there is also an additional loop which periodically and randomly performs disconnection from IoT platform (ends HTTP session and gets new tokens again). This part simulates possible failures in a real time system. The disconnection time is randomly generated number from 10 to 1440 representing number of seconds after which disconnection occurs (after that new value is generated). While the test time is 600 seconds the drawn value maybe longer than time of measures so the disconnection will not occur.

Benchmarking process

It is important to notice that each test suite was running under same environment conditions and application state. Moreover, before any test and after setting up the middleware platform we had to register each IoT device within the system. All devices had their own credentials and IDs, which allowed to generate unique JWTs for each one.

We measured time difference between events in the process and decided to log all important events with their timestamps to a log file:

- message or answer send,
- message or answer receive,
- JWT string processing begin,
- JWT string processing end,
- token cryptographic validation begin,
- token cryptographic validation end.

Each test was performed within the same time frame of 10 minutes. Such a time slot was measured on each node simulating IoT device. All threads of IoT devices were started in parallel. The test ended after the last node ended its application loop.

After all test cases have been completed, we processed log files to get all measures in sets. We decided to calculate medians for all distinguished configurations for which a test was repeated 5 times.

Evaluation Metrics

As we decided to investigate how JWT influences the performance of IoT platform, we decided to measure both the scalability of the system as well as impact of security algorithms and options on performance.

Consequently we decided to test various conditions reflected by:

1. different JWT hash algorithms,
2. various numbers of IoT devices.

Considering the obtained results we determined the load giving the largest number of messages served within a time frame under which we took additional measures. These measures lead us to detailed view of JWT processing in IoT ecosystem. We defined additional metrics:

- influence of JWT configuration on request processing time,
- influence of JWT configuration on cryptographic validation of token correctness.

As JWT may have different representations (Jones, et al., 2015) in this experiment we decided to use JWT represented as a JSON Web Signature (JWS) (Jonesl, et al., 2015). Moreover JWS can also use many different signature algorithms. In this article we performed a comparison for various JWS setups with hash-based message authentication code (HMAC) protection. Investigated hash algorithms are as follows:

- HS256 – token has message authentication code hashed to 256 bits output,
- HS384 – token has message authentication code hashed to 384 bits output,
- HS512 – token has message authentication code hashed to 512 bits output,
- NONE – token is not signed, no message authentication code is generated version used as reference.

Key lengths set for hash algorithms were 64 bytes for HMAC with HS256 and 128 bytes for HMAC with HS384 and HS512. These are most optimal values in terms of security (no additional hash operations on key is done and also the key is not padded with zero values (Krawczyk, et al., 1997).

All of above configurations can be used in practical use cases. Tokens with message authentication code should be used within untrusted networks. The HMAC algorithm prevents an attacker from tampering the message payload, so the identification of the device and its claims is certain. Usage of JWT with no signature can potentially be useful e.g. in a closed, controlled network in a factory, where all machines are identified with JWT tokens. This allows to take advantage of the JWT technology and achieve slightly better performance.

Results Analysis

Scalability

We measured system performance under various load. Load was simulated by changing the number of simulated IoT devices. We started with 160 devices (5 threads per node) and ended with 2240 devices (70 threads per node). Scalability results are presented in Fig. **2**, Fig. **3**. and Fig. **4**. Fig. **2**. presents the number of successfully (complete) processed requests. The linear representation depicts changes of performance under load. The column representation on right side provides a detailed view on differences between the tested algorithms. We can see that the number of completely processed requests during the test is continuously growing until the load achieves 1280 IoT devices, then the system becomes overloaded and the number of processed requests starts to decrease. It is also important to see how quickly the number of incomplete requests increases when the total number of IoT devices exceeds the limit of 1920. This could be seen in Fig. **3**. Fig. **4**. represents processing time on the client side. We can see that the request processing time on the client side first grows linearly and starting with the load of 1600 devices it starts to increase exponentially. Taking into account all of the measures we can conclude that most optimal load for this setup is about 1280 IoT devices, while a higher load shows that the system becomes overloaded.

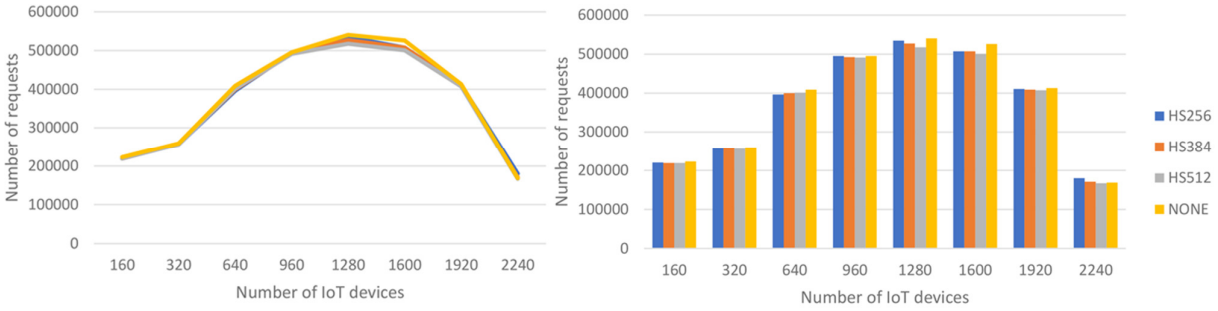

Fig. 2: Complete requests processed by middleware.

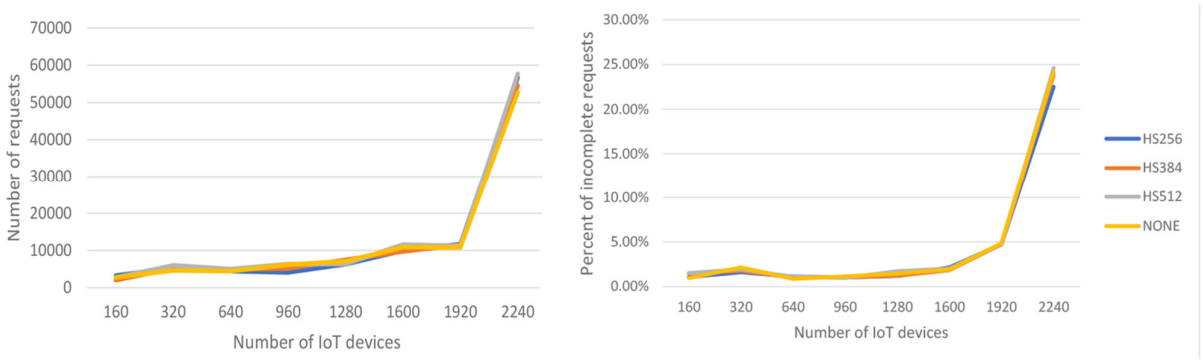

Fig. 3: Incomplete requests processed by middleware.

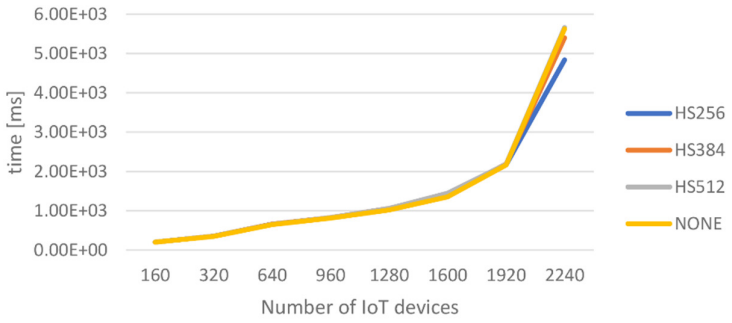

Fig. 4: Processing time on client side.

Performance vs security

Taking a look at the right side of Fig. **2**. we can see that a JWT token generated with different hash algorithms does not have a strong impact on performance results while one configuration differs from others visibly. Token configuration with no HMAC security is processed visibly faster than the others. This is because there is no additional time spent on cryptographic operations and starts to be visible only under significant system load (beginning with 1280 IoT devices).

Influence of JWT configuration on processing time

The JWT configuration which we measured was based on JWS with the HMAC algorithm. The token in the payload part contains following information: user id, network ids, device type id, token type, expiration time, available actions. This information refers to the default token configuration for the DeviceHive IoT platform. The size of that data including token header in our experiment is 117 bytes. Measures were taken under most optimal conditions for each of hash algorithms in HMAC. The key length for HS256 was 64 bytes while for HS384 and HS512 it was 128 bytes. Chosen key lengths are equal to block lengths in each algorithm (Krawczyk, et al., 1997). In general differences in processing times between HS256 and both HS384, HS512 depend on the actual size of data, 32 or 64-bit platform and possibly special instructions used for implementation. Consequently, relative times might be different for another implementation with other data sizes.

Measured Results

Fig 5. and Fig 6. represent obtained results for cryptographic measures under the selected load of 1280 IoT devices. In Fig 5. we can see that processing times of cryptographic operations are on a similar level for hash algorithms HS384 and HS512 which is expected. It is also clearly visible that processing time for hash algorithm HS256 is lowest. These results may indicate that combination of hash algorithm HS256 with 64 bytes key should result in a visible difference in processing results. Unfortunately, it is not a straightforward dependence. Fig 6. represents processing time of JWT beginning with base64 string until getting whole token structures. We can see that processing time for JWT which includes cryptographic processing time is about 13-14 times bigger than only cryptographic processing, so even visible difference in cryptographic processing times become insignificant in total. In Table 2 we can see that JWT processing time is less than 1% of message processing by middleware, while cryptographic processing is about 0.6‰ of whole processing time. Additionally, in Table 3 we presented differences in processing times on client side between hash algorithm scenarios compared to a use case with no HMAC function for the three measures - before optimal load, under optimal load and under heavy load. As we can see the difference does not exceed 7%.

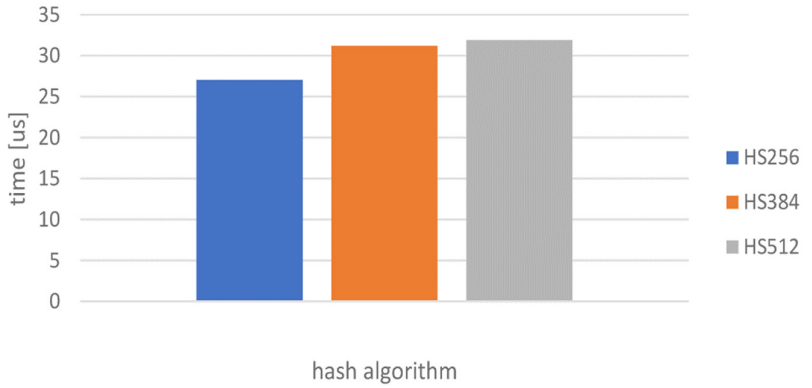

Fig 5. Processing time of cryptographic operations by hash algorithm.

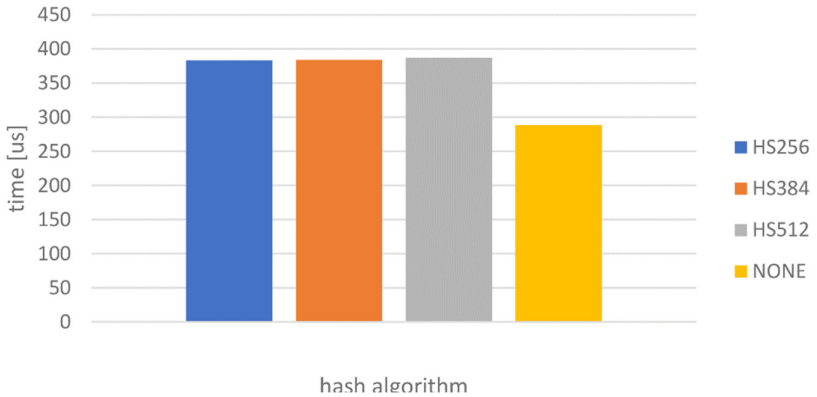

Fig 6. Processing time of JWT from base64 string to parsed objects by hash algorithm.

Table 1: Processing time of cryptographic operation for token.

Key length	Hash algorithm	Cryptographic processing time [us]
64	HS256	27.0
128	HS384	31.2
128	HS512	31.9

Table 2: Percent of JWT processing time in whole request processing.

Hash algorithm	Cryptographic processing	JWT processing
NONE	-	0.59%
HS256	0.05%	0.76%
HS384	0.05%	0.77%
HS512	0.06%	0.76%

Table 3: Differences in processing times between hash algorithm scenarios compared to a use case with no HMAC function.

Hash algorithm	Number of devices - load	Max difference in processing time
HS256	960	0.63%
HS384	960	1.19%
HS512	960	1.64%
HS256	1280	2.16%
HS384	1280	4.05%
HS512	1280	4.41%

HS256	1600	4.08%
HS384	1600	4.32%
HS512	1600	6.97%

Conclusions

In this paper, we benchmarked the performance of IoT platform under the load of various numbers of IoT clients and with different JWT configurations. Results show that while we can observe visible differences in processing time for different token setups, taking into account the overall request processing times, differences between configurations do not exceed roughly 7% under optimal load. Consequently, when setting up a token for a given scenario it should be adjusted to hardware and network requirements -- a shorter hash may better fit limited devices and bandwidth while a longer hash might be preferable in terms of potential security breaches (Krotkiewicz, 2016). Furthermore, the experiments have shown scalability of the solution up to 1280 IoT clients using 32 nodes, each with a 4-core CPU.

References

- Ahmed, S. and Mahmoo, Q. (2019) 'An authentication based scheme for applications using JSON web token.' 2019 22nd International Multitopic Conference (INMIC), Islamabad, pp. 1-6.
- Ayoade, G., El-Ghamry, A. and Karande, V. (2019) 'Secure data processing for IoT middleware systems.' *The Journal of Supercomputing*.
- Celesti, A., Fazio, M., Fermín, GM., Glikson, A., Mauwa, H., Bagula, A., Celesti, F. and Villari, M. (2019) 'How to Develop IoT Cloud e-Health Systems Based on FIWARE: A Lesson Learnt.' *Journal of Sensor and Actuator Networks,* 8(1), p. 7.
- Czarnul, P., Golaszewski, G., Jereczek, G. and Maciejewski, M. (2020) 'Development and benchmarking a parallel Data AcQuisition framework using MPI with hash and hash+tree structures in a cluster environment.' ISPDC, Warsaw
- Cruz, MAA., Rodrigues, JJPC., Sangaiah, A.K., Al-Muhtadi, J., Korotaev V. (2018) 'Performance evaluation of IoT middleware.' *Journal of Network and Computer Applications,* Issue 109, pp. 53-65.
- Datta, S. K. and Bonnet, C. (2019) 'Securing IoT Platforms' 2019 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas pp. 1-2.
- DeviceHive , *DeviceHive Homepage.* [Online] [Accessed 7 January 2021], https://devicehive.com
- Diaz-López, D., Blanco Uribe, M., Santiago Cely, C., Tarquino Murgueitio, D., Garcia Garcia, E.., Nespoli, P. and Gómez Mármol, F. (2018) 'Developing Secure IoT Services: A Security-Oriented Review of IoT Platforms.' *Symmetry,* 10(12), p. 669.
- FIWARE , *FIWARE Homepage.* [Online] [Accessed 18 February 2021], https://www.fiware.org/
- Fremantle, P. and Scott, P. (2017) 'A survey of secure middleware for the Internet of Things.' *PeerJ Computer Science*.
- Ismail, A. A., Hamza, H. S. and Kotb, A. M. (2018) 'Performance Evaluation of Open Source IoT Platforms.' 2018 IEEE Global Conference on Internet of Things (GCIoT), Alexandria, pp. 1-5.
- Jonesl, M., Bradley, J. and Sakimura, N. (2015) 'JSON Web Signature (JWS)'*, RFC Editor.*
- Jones, M., Bradley, J. and Sakimura, N. (2015) 'JSON Web Token (JWT)' *RFC Editor.*
- Kałaska, R. and Czarnul, P. (2020) 'Some security features of selected iot platforms.' *TASK Quarterly : scientific bulletin of Academic Computer Centre in Gdansk,* 24(1), pp. 29-61.
- Kokkonis, G., Chatzimparmpas, A. and Kontogiannis, S. (2018) 'Middleware IoT protocols performance evaluation for carrying out clustered data.' 2018 South-Eastern European Design Automation, Computer Engineering, Computer Networks and Society Media Conference (SEEDA_ECNSM), Kastoria, pp. 1-5.
- Krawczyk, H., Bellare, M. and Canetti, R. (1997) 'HMAC: Keyed-Hashing for Message Authentication.' *RFC Editor.*
- Krotkiewicz, J. (2016) 'An In-Depth Look into Cryptographic Hashing Algorithms.' *Algoma University at Sault Ste. Marie.*
- Rahmatulloh, A., Gunawan, R. and Nursuwars, F. M. S. (2019) 'Performance comparison of signed algorithms on JSON Web Token.' IOP Conference Series: Materials Science and Engineering, Bandung
- Rashwan, A. M., Taha, A. M. and Hassanein, H. S. (2012) 'Benchmarking message authentication code functions for mobile computing.' 2012 IEEE Global Communications Conference (GLOBECOM), Anaheim, pp. 2585-2590.
- Shingala, K. (2019) 'JSON Web Token (JWT) based client authentication in Message Queuing Telemetry Transport (MQTT)' *arXiv*.
- SiteWhere, *SiteWhere Homepage.* [Online] [Accessed 22 February 2021], https://sitewhere.io/en/
- Solapurkar, P. (2016) 'Building secure healthcare services using OAuth 2.0 and JSON web token in IOT cloud scenario.' 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, pp. 99-104.
- Vijayakumar, N. and Ramya, R. (2015) 'The real time monitoring of water quality in IoT environment.' 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], Nagercoil, pp. 1-4.
- Xiaojun, C., Xianpeng, L. and Peng, X. (2015) 'IOT-based air pollution monitoring and forecasting system.' 2015 International Conference on Computer and Computational Sciences (ICCCS), Noida, pp. 257-260.