Measuring The Impact of Post-Quantum Cryptography on Complex Applications: A Case Study on Federated Identity Management

Ginevra Fabrizio, Anna Sperotto, Roland van Rijswijk-Deij
University of Twente, Enschede, The Netherlands
{g.fabrizio, a.sperotto, r.m.vanrijswijk}@utwente.nl

***Abstract*—The arrival of quantum computers will significantly impact our current Internet infrastructures as quantum computers will be able to break current public key cryptography. This means that we need to replace cryptographic algorithms by quantum-safe alternatives, also known as Post Quantum Cryptography (PQC). Many protocols and architectures need to make this transition. While the Internet community is making steady progress on transitioning TLS to PQC, there are still many protocols for which action needs to be taken, especially in more complex scenarios where protocols span multiple layers of the stack. In this paper, we study such a complex scenario: Federated Identity architectures. The objective of these architectures is to allow access to multiple services using a single set of login credentials, improving convenience and security across different organizations or domains. In particular we examine the Hub'n'Spoke model, where multiple parties exchange data through a central hub. This hub manages most of the infrastructure workload and deals with many heterogeneous devices and protocols, making it a perfect test case for the PQC transition. Using real-world data from an operator of a large academic identity federation, we benchmark five PQC algorithms finalized by NIST. We also quantify the toll this transition imposes on computational efficiency and hardware capacity. We show that while there is an ever increasing interest towards PQC adoption, many technical challenges remain unaddressed, showing that the Post Quantum transition process is fragmented, with many components of the ecosystem that are still insufficiently taken into consideration.**

***Index Terms*—Post Quantum Cryptography, SAML, OpenID-Connect, Federated Identity.**

I. Introduction

Quantum computers promise to be the next revolution in computing and will change the Internet as we know it today. While quantum computers will bring many benefits, they also pose a threat: they can be used to break current public key cryptography. This means that current cryptographic algorithms need to be replaced by quantum-safe alternatives, also known as Post-Quantum Cryptography (PQC). Almost all protocols and architectures we use today will need to carry out this transition. While everyone agrees that such a transition should be as smooth as possible, there is no guarantee this will be the case. Compared to traditional cryptographic methods, PQC algorithms may require greater key sizes or demand more computational power for encryption, decryption, and key management [1]. While the community is making great strides in transitioning important protocols, such as TLS, more complex scenarios, e.g., where protocols and applications span multiple layers of the stack, remain understudied.

978-3-903176-74-4

In this paper, we investigate the impact of the PQC transition in an operational setting. We focus on Federated Identity Management (FIdM) as FIdM is much more complex than a TLS single channel communication scenario. FIdM is a crucial aspect in today's world, allowing users to access third-party services using a single identity. FIdM architectures see extensive use in academia, enterprise and governmental settings [2], [3]. Federations can have highly complex and customized implementations, often dealing with heavy computational loads. This load might increase as the switch to PQC takes place. As organizations transition to PQC, they may need to accommodate an additional strain on their resources and require more powerful hardware. These considerations make FIdM an exemplary test case for quantifying the impact of the PQC transition on more complex systems. To maximize our understanding of complexity, we focus on a particular FIdM architecture: the Hub'n'Spoke model, which spans multiple layers of the stack and needs to perform protocol translations that require additional cryptographic operations.

This paper addresses this problem by answering the question *What is the performance impact on a federated authentication process when a switch to PQC occurs?* We do so by leveraging real-world data from the operator of a large academic identity federation – that serves millions of users – to model a PQC transition scenario based on benchmarks of five Post-Quantum algorithms selected for standardization by NIST [4].

The main contributions of this paper are:

- We use real world data of a Hub'n'Spoke federation architecture that allows us to extrapolate the impact of transitioning to PQC;
- We combine this data together with PQC benchmarks and show that - surprisingly - the most likely PQC candidate algorithms may outperform existing algorithms. However, if the class of PQC algorithms based on lattices turns out to have flaws, the next possible scenario raises serious performance concerns, requiring up to five times the processing power used today to handle current workloads in terms of user authentications.

The remainder of the paper is organized as follows. Sec. II introduces the relevant background. In Sec. III we present

the dataset on which our model is based. Sec. IV details our approach, and Sec. V presents our findings. Finally, we conclude in Sec. VI.

II. Background

A. Post Quantum Migration

Secure online communication today cannot be considered safe from the future threat posed by quantum computers, as the computing power of a yet-to-come quantum machine can easily break current encryption schemes simply through Shor's algorithm [5]. To establish a secure connection we rely on the use of Key-Establishment Mechanisms (KEMs). KEMs allow two parties to agree on a shared key to communicate. KEMs are vulnerable to the so-called "Harvest Now Decrypt Later" (HNDL) attacks, in which malicious actors intercept encrypted data, store it somewhere, and attempt to decrypt it in the future. Similarly to KEMs, digital signatures and certificates (DSAs) are at risk as well, as a quantum attacker will be able to forge any signature, thus compromising a message's authenticity and integrity. Because of this, any connection that uses a non-quantum safe signature scheme is vulnerable to MitM (man-in-the-middle) attacks. The key difference here is that, unlike HNDL attacks, MitM attacks can only compromise active connections. In other words, signature schemes cannot be compromised as long as quantum computers do not become reality. However, migrating signatures is much more complicated than migrating KEMs, as certificated and trust infrastructures are tied to legacy systems that need to be audited, the use of new algorithms has to be approved by the CA/Browser forum, and certificates need to be reissued and the trust chains need to be updated. Therefore, the PQC migration can be described as a two-step process with two components, Key-Establishment Mechanisms (KEMs) and Digital Signature Schemes (DSAs). To standardize new PQC algorithms, the US National Institute of Standards and Technology (NIST) launched a competition for Post-Quantum algorithms [6]. In 2022 NIST announced 4 candidates for PQC standardization [7], one KEM (Kyber [8]) and three DSAs (Dilithium [9], Falcon [10] and SPHINCS+ [11]). The choice of having more than one algorithm is due to the fact that candidates rely on different classes of mathematical problems. This ensures that if one of the underlying classes of problems is broken, either by a classic or by a quantum computer, there are fallback solutions to rely on. For the same reason, the NIST competition has been extended to a fourth round to select more KEM algorithms with different underlying mechanisms. As of March 2025, a new candidate KEM algorithm for standardization, HQC, has been announced totaling to 5 current candidates, two KEMs and three DSAs [4]. The concept of switching to a different type of algorithm in case a class of algorithms turns out to be unsafe and easing the whole migration process is called "crypto-agility". In line with this principle, algorithms mixing current and PQ methods, called "hybrid algorithms" already exist. In fact a hybrid version of a KEM algorithm based on Kyber (X25519-MLKEM), is already in use by multiple online services and user agents. We will focus on NIST candidates and hybrid algorithms to reflect the importance of the standardization process and the deployment of hybrid algorithms such as X25519-MLKEM.

B. Federated Identity Management Architectures

An Identity Federation is a framework to manage user identities across multiple services. A federated approach (FIdM) allows users to access resources from different domains without having to create new accounts or credentials.

In an Identity Federation, user identities are managed by an Identity Provider (IdP), while Service Providers (SPs) are responsible for delivering services to the end user. IdPs and SPs typically rely on standardized protocols such as:

- OpenID Connect (OIDC): OpenID Connect [12] is an extension of the OAuth 2.0 protocol [13], specifically designed for user authentication.
- Security Assertion Markup Language (SAML): SAML is an XML-based standard to exchange authentication and authorization data between systems [14].

Even though these protocols' objectives differ from each other, they can be used either standalone or together, depending on the implementation. Every federation framework can be designed in two ways:

- Mesh: each node can authenticate with any other node, without the need for a central hub. The entities know to whom they have to communicate thanks to a centrally distributed SAML metadata file that describes who is in the federation. A discovery service can be centralized or can be accessed locally by SPs. The management of the metadata file and the attribute release in Identity Providers is challenging.
- Hub'n'Spoke: this architecture relies on a central hub that authenticates users and then issues tokens to the service providers. The hub receives all the authentication requests, meaning that it is a single point of failure, but also that we have an easier authentication flow. Additionally, Hub'n'Spoke federations can accommodate mixing multiple FIdM protocols, such as SAML and OIDC.

Both Hub'n'Spoke and Mesh federations will be affected by the PQ migration at two different levels. First, both architectures are based on SAML or OIDC protocols at the application layer. These protocols make use of signatures (XML signatures for SAML [15], and Java Web Token (JWT) [16] for OIDC) which will need to be adapted in a PQ scenario. Secondly, both architectures rely on Transport Layer Security (TLS). It is important to note that cryptographic operations happening at the transport layer need to be migrated to PQC and also produce notable overhead [17]. We can see that a single connection via a FIdM therefore involves a large number of cryptographic operations, all of which may potentially become more computationally expensive. Considering that in a real-world scenario millions of connections happen at the same time, the need to quantify the performance impact of PQC in these architectures becomes clear.

Hub'n'Spoke authentication flow. In this paper, we focus on the Hub'n'Spoke model. Fig. 1 shows that the Hub'n'spoke

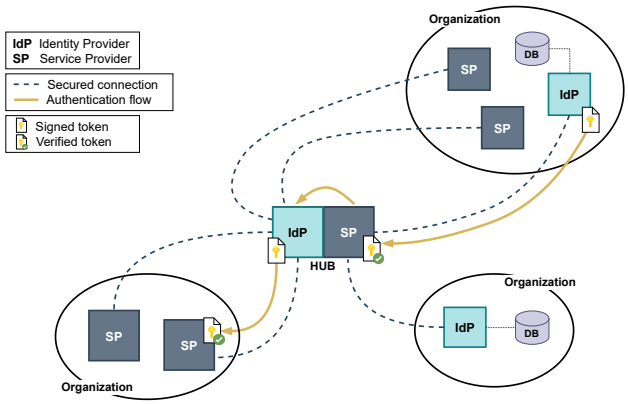

Fig. 1. A simplified scheme showing part of an authentication flow, focused on signature operations, in a Hub'n'Spoke architecture federated login.

model has two main components, Service Providers (SPs) and Identity Providers (IdPs). The two components communicate through a central hub that acts as an SP with the IdP side of the connection and as an IdP with the SP. When, e.g., user authentication takes place, an IdP sends a signed token to the hub. The hub verifies the token and signs it again (as an IdP) and sends it to the SP that requires the user to authenticate to their service. The SP then verifies the token and admits the user. This operation of course happens at the same time for thousands of SPs, IdPs and users, all in one central hub, that has to manage the secure connection and the correct authentication flow.

C. Related work

Several studies have explored PQC authentication protocols, showing the impact of migrating these protocols to quantum-safe ones. Recent work by Müller et al. [18] designed a PQ solution for SAML XML signatures, integrating them into a Single Sign-On protocol. Schardong et al. [19] focused on the OAuth2.0 and OpenID Connect authentication flow, showing how TLS handshake accounts for half of the connection load. We refer the reader to the following work [20], for a good overview on the work conducted so far on TLS and post-quantum adaptation. Regarding other works on PQC adaptation, Müller et al. [21] also analyzed how DNSSEC can achieve algorithm agility in light of a quantum transition in the future and in their work Goertzen et al. [22] also investigate application of PQ digital signatures selected by NIST in a DNSSEC application-level implementation. To the best of our knowledge this is the first paper to analyze federated identity with real-world data.

III. Dataset

We worked with SURF, the operator of a large European academic research network, to study their federation called SURFconext. SURFconext is one of the oldest identity federations in operation and occupies a fairly notable position in the FIdM landscape [23]. SURF's federation connects more than 3,000 service providers to over 200 identity providers [24]. Furthermore, it is interconnected to the global eduGAIN federation [25]. SURFconext uses a Hub'n'Spoke architecture, and we focus on their central hub, called Engine. We analyzed Engine HTTPS requests for 25 days, from 2024-09-13 to 2024-10-08. This period covers what is generally the busiest period of the year for SURF's identity federation as it coincides with the start of the academic year when students need to (re-)enrol for classes and a new cohort of students starts. The requests are logged in JSON format, showing timestamp, the server that generated the log entry and the type of log entry. A log entry can be of three types: HAproxy (all HTTPS requests), eb-auth (successful authentications), or stepup-auth (successful step-up authentications).

Step-up authentication, the two-factor authentication that SURF offers as a value-added service, is not considered in the scope of this paper. As such, data related to step-up authentication will not be described nor taken into consideration.

HAproxy logs contain all HTTPS requests handled by SURF's infrastructure. All data is filtered and all fields are anonymized. The dataset contains detailed information on the HTTP requests received. Here we describe only the fields that were used to estimate the load on Engine infrastructure:

- `backend-app`: the SURFconext service that handled the request. We will use this to distinguish between the SAML or the OIDC authentication flow;
- `http-code`: the HTTP response code. A response in the 400 or 500 range means that some signing or verification operations may not have been performed;
- `path`: the request's URL path. From this entry we derive what step of the authentication flow was performed and what cryptographic operation(s) occurred.

Each eb-auth log entry corresponds to a successful SAML or OIDC authentication event.

IV. Approach

This section discusses the approach taken to predict the impact of the PQC transition. It starts by analyzing what factors play a role in the number of signature validations in a Hub'n'Spoke federated architecture. We identify two main scenarios, and for each one of them we introduce a model describing the relationship between the tasks of a Hub'n'Spoke federation and the number of cryptographic operations executed. The section ends with a validation of these models against real-world data.

As already discussed, PQC migration can be split into two components: key exchange (KEM) and digital signatures (DSA). Even though these two components will likely be migrated at different points in time, for the sake of simplicity, and to estimate the eventual load the system will need to process once the transition completes, here we assume that both will transition at the same time. We take into consideration the following NIST PQC algorithm candidates and a hybrid KEM algorithm that is already being used by several services:

- Key Encapsulation Mechanisms (KEMs):

  - CRYSTALS-Kyber (ML-KEM);
  - X25519+Kyber (Hybrid KEM);
  - HQC (newly selected, March 2025 [26]);
- Digital Signature Algorithms (DSAs):
  - SPHINCS+ (SLH-DSA);
  - CRYSTALS-Dilithium (ML-DSA);
  - FALCON (FN-DSA).

We then compare the performance of these algorithms with the one currently used by SURF, which is RSA with 2048- and 3072-bit keys (both key sizes are used at different points in the architecture). We want to assess the load that the federation hub, ENGINE, has to bear, as this component carries out the most cryptographic operations. We assume that most of the load on ENGINE is caused by signing, signature validation and key exchange operations, which we will refer to as cryptographic operations from now on. We do this to reflect the fact that in a PQC migration the main concern is the potentially (much) higher processing requirements of these operations. Of course other operations related to request processing contribute to the entire infrastructure load, but their effect is beyond the scope of this research. To quantify the number of cryptographic operations we describe what happens in the SURFCONEXT architecture with two simple workflows, by looking at the HTTP requests and the logs that SURF gave us. The main focus is to understand who performs the operations and when they are carried out. For each operation we will always take into consideration the scenario when the highest number of operations are performed by ENGINE, as a worst-case scenario in case of performance impact.

A. Authentication flows

The Hub'n'Spoke architecture analyzed here is the SURFCONEXT infrastructure. The request structures that we illustrate here are specific to SURFCONEXT's Hub called ENGINE. We use the path of requests contained in the analyzed logs to infer the operations that happen in the authentication flows. ENGINE can handle two different types of authentication flow:

SAML Authentication. This is ENGINE's most common authentication flow. It makes use of SAML requests. All SAML requests pass through the front channel (the user's browser) and make use of HTTP redirects. The flow, shown in Fig. 2, consists of four steps:

1) `/idp/single-sign-on`: ENGINE receives a SAML Authentication request from an external SP. In this step no signature validation is performed;
2) ENGINE sends a SAML Authentication request to the user's home institution IdP;
3) `/sp/consume-assertion`: ENGINE receives a SAML Response from the institution's IdP. Each SAML Response is signed, so *a signature validation is performed*;
4) if the previous HTTP request goes well, ENGINE creates and signs a SAML Response for the external SP. *A signing operation is executed.*

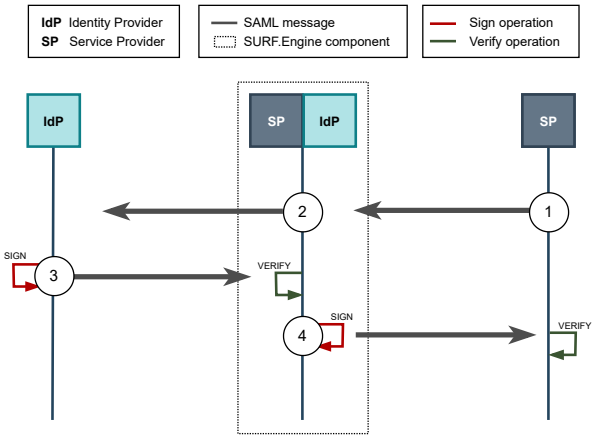

Fig. 2. A summary of cryptographic operations carried out in a typical SAML flow within the ENGINE Hub'n'Spoke architecture.

OIDC Authentication. This flow uses a mix of OpenID Connect and SAML. The OIDC requests are translated by a proxy called OIDCNG that we will consider to be part of the ENGINE ecosystem. In this flow we have 10 steps, shown in Fig. 3, where the last two are not necessarily performed with each authentication:

1) `/oidc/authorize`: OIDCNG receives an OIDC authorize request from an external SP[1];
2) `/saml2/authenticate/oidcng`: OIDCNG starts a SAML authentication to ENGINE A signed SAML Authentication request is sent to ENGINE. *OIDCNG performs a signature operation*;
3) `/idp/single-sign-on`: ENGINE receives a SAML Authentication request from OIDCNG, *a signature validation is performed by* ENGINE;
4) ENGINE sends a SAML Authentication request to the user's home institution's IdP;
5) `/sp/consume-assertion`: ENGINE receives a SAML Response from institution's IdP, *a signature validation is performed.*
6) ENGINE creates a SAML Response for the external SP, *the SAML Response is signed.*
7) `/oidc/saml2/sso/oidcng`: OIDCNG receives a SAML Response from ENGINE. *A signature validation is performed.*
8) `/oidc/token`: OIDCNG sends an OIDC token response to the external SP. The OIDC token contains three signed Java Web Tokens (JWTs), the `id_token`, the `access_token` and the `refresh_token`. These tokens are signed using RS256 (RSA 2048 with SHA-256). *Three signing operations are performed*;
9) `/oidc/introspect`: the introspect endpoint checks the validity of the access token. *A JWT signature validation is performed*;

[1]In OpenID Connect, Service Providers are typically referred to as Relying Parties (RPs), for simplicity we use the term SP throughout the paper.

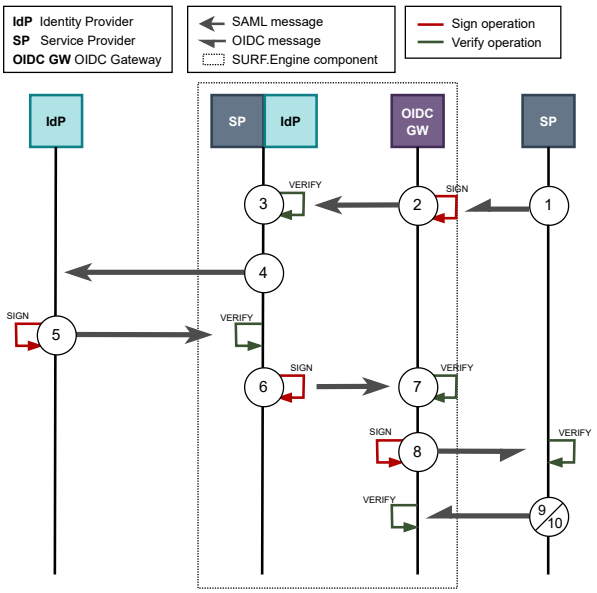

Fig. 3. A summary of cryptographic operations carried out in a typical OIDC Authorization Code flow within the ENGINE Hub'n'Spoke architecture.

10) `/oidc/userinfo`: the userinfo endpoint checks the validity of the access token. *A JWT signature validation is performed*;

Even if ENGINE and its OIDC gateway are two different components, the load from cryptographic operations by OIDCNG will be considered part of the ENGINE infrastructure. So, every time we refer to the ENGINE infrastructure, we will also refer to the OIDCNG proxy that handles OIDC requests. We provide a detailed breakdown of how each step contributes to the load in Section V (Tables III and IV).

B. Cryptographic operations

To each path of the request URL reported in the logs, we assign a signing or validation operation. In this way, we can count how many cryptographic operations are performed. In our analysis, we consider the day with the highest number of HTTP requests received by ENGINE because we always want to consider the worst-case scenario for migrating to PQC algorithms. The worst-case scenario for the migration case is the case where the majority of cryptographic operations are performed by ENGINE. The logic for SAML flows is pretty straightforward, as the SAML XML signature scheme requires only a single signing operation (and consequently a single validation). This means that in a SAML flow one path request usually corresponds to one cryptographic operation. That is not the case for an OIDC authentication flow, where one operation can imply several signing and validation operations. How many cryptographic operations are performed depends on the type of OIDC Authentication flow that is implemented and how. We follow the worst-case scenario, where we only consider the case where ENGINE infrastructure performs the highest number of cryptographic operations. The OIDC Authorization Code flow involves the highest number of cryptographic operations and as such it will be the only one considered. Other flows, such as the Implicit Grant and the Hybrid Grant, are not considered in this paper. Furthermore, in this study we do not consider the Base64 encoding/decoding times for SAML and JWT digital signatures and focus purely on the processing time required for signing and validation.

On top of the cryptographic operations required for signing and validation, for both authentication flows we have to consider all the incoming TLS connections that are managed by the ENGINE infrastructure, where ENGINE act as the server endpoint in the connection. This means that, always following the worst-case scenario logic, each request to each path will be counted as an incoming TLS connection. Each TLS connection then requires two cryptographic operations: a key exchange and a signing operation (to sign the `ServerHello` message to authenticate the server). For the sake of simplicity, we assume that all connections to ENGINE use TLS 1.3 with the key agreement mechanism preferred by the ENGINE server (X25519). For signing ENGINE is currently configured to use RSA 2048-bit. This information has been profiled with the SSL Server Test tool available on the SSL Labs website [27].

V. RESULTS

In this section we discuss our results. As discussed in Sec. III, we use 25 days of real-world data from the busiest period for SURFCONEXT of the academic year. Our dataset contains information on 106M requests to ENGINE, with an average load of 3.9M requests per day and a peak load of 579 266 requests per hour, typically occurring during working hours. In the remainder of this section we first discuss how we benchmarked cryptographic algorithms. Then, using these benchmarks, we take the data on requests to ENGINE to infer the impact of migrating to PQC algorithms for different combinations of PQC algorithms.

A. Benchmarks

To estimate the time elapsed for each possible cryptographic operation, we conducted benchmark tests for each algorithm. For DSAs we selected three commonly used algorithms that are in current use, two of which (RSA 2048- and 3072-bit) are extensively used by the SURFCONEXT infrastructure and four PQC algorithms, standardized or selected for standardization by NIST. In general, we selected variants of PQC algorithms at the basic NIST security level (level 1) as the security this provides corresponds to that of algorithms currently used in SURFCONEXT. To benchmark the PQC algorithms we use OpenSSL v3 with the Open Quantum Safe (OQS) provider, which implements both NIST-approved and novel PQC algorithms [28]. For more information on how to use `oqs-provider` with OpenSSL we refer the reader to the official guide provided by the `liboqs` team [29]. Benchmark data was collected by performing 100 independent OpenSSL speed tests for each class of algorithms. A single OpenSSL speed test consists of a 10-second run with continuous calls

TABLE I
BENCHMARKS FOR SIGNATURE ALGORITHMS; CURRENT ALGORITHMS ARE AT THE TOP, PQC ALGORITHMS ARE AT THE BOTTOM.

	Signing (ops/s)		Validation (ops/s)	
Algorithm	**mean**	σ	**mean**	σ
RSA 2048-bit	4 152.56	12.84	61 298.74	208.51
RSA 3072-bit	1 453.12	3.84	28 692.43	169.98
ECDSA P-256	54 085.43	148.55	17 186.14	49.07
MLDSA-44	17 386.10	83.39	46 578.86	356.02
Falcon 512	5 182.85	65.21	29 422.18	79.07
sphincssha2128fsimple	164.85	1.82	2 143.63	42.46
sphincssha2128ssimple	8.00	0.04	5 764.51	167.43

TABLE II
BENCHMARKS FOR KEY AGREEMENT ALGORITHMS; CURRENT ALGORITHMS ARE AT THE TOP, PQC ALGORITHMS ARE AT THE BOTTOM.

	Encapsulate (ops/s)		Decapsulate (ops/s)	
Algorithm	**mean**	σ	**mean**	σ
X25519	15 293.55	169.34	29 449.67	68.87
EC P-256	14 197.04	58.25	17 488.46	58.49
EC P-384	696.64	5.83	1 389.27	13.23
MLKEM512	147 695.81	1 627.40	155 274.59	1 552.68
MLKEM768	96 067.42	939.99	98 044.23	930.79
X25519-MLKEM512	14 268.43	46.74	14 460.57	40.06
X25519-MLKEM768	13 571.50	39.70	13 739.70	39.23
HQC-128	313.04	1.28	206.26	0.97

to signature generation, signature validation and key exchange functions. The benchmark tests were run on a single core of an AMD EPYC 9534P 64-core running at 3.1 GHz and with 4 GB memory (4800MT/s). This CPU architecture is representative of modern server architectures and comparable in performance to the systems used by SURF for their federation. Table I shows the average and standard deviation over 100 tests for DSA speeds, with their speed expressed as operations per second. Here we can already see that the most attractive PQC candidate to replace signatures is MLDSA-44 followed by Falcon 512 while the worst is SPHINCS+. Regarding signature validation, SPHINCS+ performs slightly better, but not enough to justify it as a feasible option in case lattice-based algorithms were to be broken, as we will see later on when we use these benchmarks to evaluate the impact on the real-world infrastructure of SURFCONEXT.

For KEMs we also selected three algorithms that are in current use, one of which – X52219 – is the preferred algorithm for the server component ENGINE in SURFCONEXT, and five PQC algorithms, two of which are hybrid variants that combines a current algorithm (X52219) with a PQC algorithm. Table II shows the results of our benchmark tests. Note that while we apply the same principle we used for DSAs of using the basic NIST security level (level 1), we make an exception for MLKEM, where we include MLKEM768, which has security level 3. Similarly, for the hybrid variant we include X25519-MLKEM768 as hybrid variant. The reason for this is that several large organisations have deployed TLS with hybrid KEM X25519-MLKEM768 in the wild [30]–[32]. Regarding key agreement algorithms, there are not many surprises, with MLKEM performing even better than "traditional" key exchange mechanisms such as X25519 or ECP. The newly announced NIST KEM PQC candidate, HQC, does not show palatable performance metrics.

B. Costs and load on the infrastructure

We now combine the results of our benchmarking with the real-world data we obtained from SURF for their identity federation. To give a feel for the data, Fig. 4 shows the number of successful authentications for the two different FIdM protocols, SAML and OpenID Connect, for the 10 biggest Identity Providers in the federation. As the figure shows, SAML is the most widely used. In a discussion with

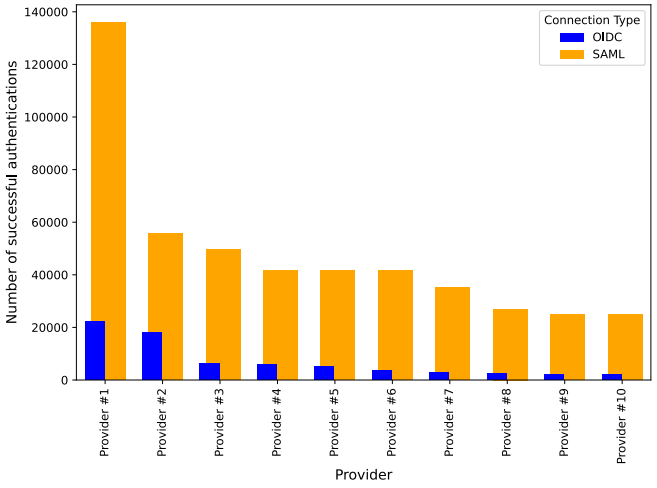

Fig. 4. Number of successful authentications by IdP grouped by authentication flow. We can see that the OIDC authentications are less than the SAML ones.

SURF we learned that OpenID Connect is used primarily in mobile app scenarios.

To model the impact of migrating SURFCONEXT to PQC algorithms, we must first calculate the number of cryptographic operations it performs for a realistic workload. To do this, we analyze the dataset we obtained from SURF. For each request path, as discussed in Sec. IV-A, we assign a number of operations carried out and estimate their load on ENGINE infrastructure. We do this for both authentication flows. Tables III and IV show the number of times different endpoints involved in authentication flows were accessed on ENGINE for a single day in our dataset (2024-09-16). We distinguish between three main operations: signatures, validations and TLS connections. To give an intuition for the load on a typical day, we plot the number of operations over time for 2024-09-16 in Fig. 5. As the figure shows, the load follows a clear diurnal pattern with peak load occurring during working hours. For each operation we use the logs described in Sec. III to estimate the number of cryptographic operations performed by unit of time, which we fix to one second.

With the computed number of operations for each type, we then use the benchmarks previously shown to estimate how many seconds are spent performing cryptographic operations by the ENGINE infrastructure. In the ENGINE infrastruc-

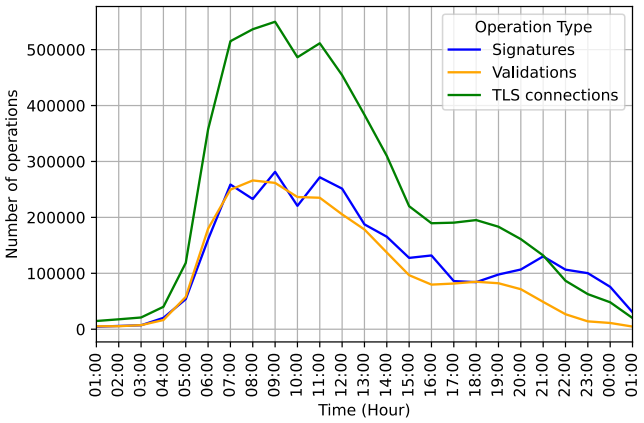

Fig. 5. Number of operations in a day (2024-09-16) sampled by hour and grouped by type of authentication (time in UTC).

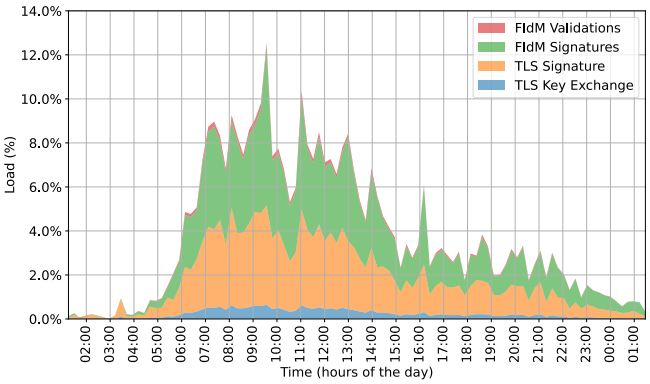

Fig. 6. Load of all cryptographic operations for ENGINE's current implementation (Scenario 1). FIdM signatures and validations are largely RSA 3072-bit based. TLS key exchange is based on X25519, while TLS signature is RSA 2048-bit based.

TABLE III
OPERATIONS HTTP PATH COUNT - SAML AUTHENTICATION FLOW

Path	Requests	Signing	Validations
/idp/metadata	5 070	1 (SAML)	none
/idp/single-sign-on	2 356 773	none	none
/idp/unsolicited-single-sign-on	778	none	none
/proxy/idps-metadata	659	1 (SAML)	none
/sp/consume-assertion	1 602 601	1 (SAML)	1 (SAML)
/sp/metadata	57	1 (SAML)	none

ture we also include the load of the OIDCNG proxy that translates the OIDC requests to SAML ones. To determine whether cryptographic operations actually took place, we check if the `http-code` mentioned in Sec. III returns a value of 4xx or 5xx. Always following the worst case scenario we assume all cryptographic operations have happened, except for cases discussed with SURF. If, e.g., a request for `/sp/consume-assertion` returns an error, we can assume that a validation took place, but no signature was created.

The CPU load is expressed as percentage and shows the CPU time used on a single core to perform a set of cryptographic operations. This means that the maximum load of 100% corresponds to a single CPU core being completely saturated by cryptographic operations. Next to the maximum load we also consider the desired overprovisioning level as expressed by SURF. SURF would prefer a situation where systems never exceed 60% load. In case this load is exceeded

TABLE IV
OPERATIONS HTTP PATH COUNT - OIDC AUTHENTICATION FLOW

Path	Requests	Signing	Validations
/login/saml2/sso/oidcng	110 674	none	1 (SAML)
/oidc/authorize	285 298	none	none
/oidc/introspect	409 486	none	1 (JWT)
/oidc/token	134 428	3 (JWT)	none
/oidc/userinfo	328 144	none	1 (JWT)
/saml2/authenticate/oidcng	169 763	1 (SAML)	1 (SAML)

the requests queue is filling up. The maximum load and overprovisioning is calculated considering only a single core. We note that there are other constraints on resource usage by PQC algorithms, notably memory use. In practice, however, none of the algorithms we study in this paper have memory requirements that would tax a modern system such as the one we used for benchmarking [33].

To evaluate the impact of PQC on the operation of an identity federation we consider four scenarios: i) the current situation (which relies on RSA for signatures and X25519 for key exchange), ii) a PQC scenario with the fastest PQC signature algorithm from our benchmarks and a hybrid key agreement scheme, iii) a PQC-only scheme with the fastest signature- and key agreement schemes, and iv) a PQC scheme that avoids the use of lattice-based PQC algorithms. We explain each scenario in more detail below.

Figures 6, 7 and 8 show three of these scenarios (we omit the PQC-only scenario (iii) as the plot is very similar to Fig. 7). We show in the figures that there are two types of signature operations performed, one at the transport level (TLS) and one at the application level (FIdM). In the current implementation, these two use different algorithms, but in the PQC models we assume they use the same signing algorithm regardless of the ISO/OSI layer considered. In these figures we will only consider the impact of signatures, validations and key exchanges that are performed by ENGINE. The figure shows a breakdown of the load based on TLS and FIdM operations, and the sum of the costs for each of those operations determines the total load of the system. While we are aware that SURF servers are equipped with 4-core architectures, for this analysis we consider a single-core implementation scenario.

Fig. 6 shows the load for all cryptographic operations based on the current implementation at SURFCONEXT. As the figure shows, the load is well within bounds even on a single CPU core, reaching a maximum of 12.56% and an average value of 3.75% throughout the whole day. What is also clear from the figure is that in the current situation, the majority of the load is

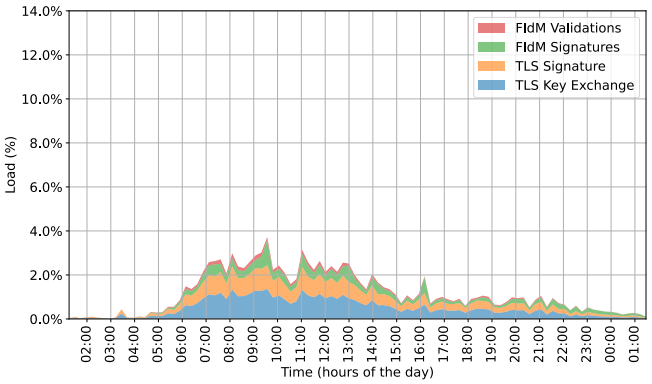

Fig. 7. Load of all cryptographic operations for ENGINE's lattice based possible implementation. All digital signatures schemes are based on NIST standard MLDSA, while key exchanges are based on hybrid scheme X25519MLKEM768 (Scenario 2).

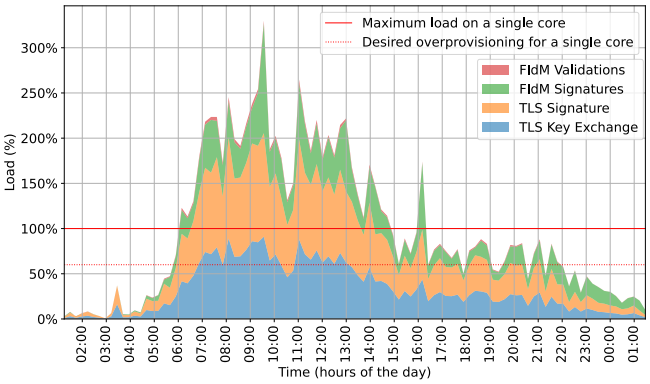

Fig. 8. Load of all cryptographic operations for ENGINE's non lattice based possible implementation. All digital signatures schemes are based on NIST standard SPHINCS+, while key exchanges are based on HQC, another (newly appointed) NIST standard (Scenario 4).

due to signing operations, which makes sense considering the current signature algorithm is RSA. As Table I shows, RSA signing is an order of magnitude slower than validation.

Fig. 7 shows our first PQC scenario. For this scenario, we pick MLDSA for signatures and X25519-MLKEM768 for key establishment. We can see that in this case we have a maximum load of 3.73%, with an average load of 1.16%. We make these choices as MLDSA has the most promising performance (as shown in Table I), and we pick X25519-MLKEM768 since it is already being deployed by several large operators as discussed in Sec. V-A. We do not consider a hybrid signature scheme as there are no standards for this yet. It is, however, safe to assume that adding the load of the PQC scheme and the current non-PQC scheme gives a realistic estimate for the load of a hybrid scheme. As discussed, we also consider a PQC-only scenario (i.e., with a PQC KEM instead of a hybrid variant), in particular with MLDSA-44 as signature scheme and MLKEM512 as key exchange mechanism. For this pure PQC scenario, we specifically pick these two algorithms as they both belong to NIST security level number 1 and are the fastest ones of their category, as shown in Table I and II. As expected, this case performs even better than the hybrid scheme shown in Fig. 7. In this pure PQC scenario, we reach a maximum load of 2.61%, with an average of 0.76%.

Figure 7 shows that the load of the MLDSA and X25519-MLKEM768 PQC scenario is on par with, if not even slightly better than the load of the existing cryptographic algorithms shown in Fig. 6. This makes a lattice-based implementation the most optimal transition scenario since the load on the infrastructure is the same or better. What is also clear is that in this scenario, the load is more evenly balanced between key establishment and signing. This means that if lattices are proven to remain (quantum-)safe then migrating to a pure lattice-based system is optimal.

Relying only on a single family of PQC algorithms, such as lattices, is a cause for concern [4], [7]. For example, recent works include successful [34] and less successful attempts [35] at reducing the complexity of solving the lattices problem. For this reason NIST is choosing several candidates belonging to different families of mathematical problems. For the second PQC scenario, shown in Fig. 8, the algorithms chosen are SPHINCS+ and HQC because they are the only non-lattices NIST standards as of now. In particular, we pick HQC-128 and SPHINCS+ 128 as they fall into the NIST level 1 category of PQC algorithms. Note that we do not consider a hybrid variant of HQC, since there is no standard to support this at present. However, it is safe to assume that a hybrid variant that includes HQC would perform slightly worse (as is the case for "pure" MLKEM768 versus X25519-MLKEM768 as shown in Table II). Compared to the lattices-based scenario, the performance of SPHINCS+ and HQC is worrisome. Compared to Fig. 6 and 7, Fig. 8 also shows the desired 60% overprovisioning and the maximum load for a single core. We can see that we not only exceed the 60% overprovisioning threshold for most of the day (from indicatively 6:00h till 22:00h), we also significantly exceed the maximum load for office hours. In addition, we have more than 3 seconds of core load at the peak of operations on a day, which is dangerously close to a 60% overprovisioning threshold even on a 4-core architecture. This means that SURF will likely have to expand their infrastructure if they ever need to accommodate such a PQC scenario. They would need more than 5 times the computational resources they currently have (3/0.6), requiring an investment on their part. Even taking into consideration that the OQS library has limitations [36] and that performance might improve compared to our current benchmarks, SPHINCS+ and HQC remain unattractive choices.

VI. CONCLUSIONS

In this paper we set out to estimate the impact of transitioning a real-world Hub'n'Spoke identity federation to two different PQC implementations. Our operational scenario is based on data from SURF, a large research network operator for their SURFCONEXT. Based on our analysis we draw following conclusions:

Lattices and alternatives – We show that while a lattice-based PQC implementation has the potential to perform even better than the current implementation used by SURF, a non lattice-based one leads to performance concerns. Relying solely on lattice-based schemes for post-quantum security, however, is not a safe option, in case lattices turn out to be unsafe. The current PQC candidates therefore pose a conundrum. We can, in operational settings, either choose for a performant combination banking on the safety of lattices (effectively putting all our eggs in one basket), or face investments in new infrastructure. There is therefore a clear risk that operational deployments might move to lattice-based PQC. NIST's press release and their draft documents [4], [7] emphasize the importance of exploring alternatives, and for this reasons candidates such as SPHINCS+ and HQC have been included. Unfortunately, their performance is problematic. It is therefore advisable to invest more time in selecting standards other than lattices especially ones that are more performant than SPHINCS+ and HQC.

Existing protocols – The current SAML standard, widely used for federated identity management and the main protocol in SURFCONEXT, has to be revised to accommodate post-quantum resilience. The existing version of SAML 2.0, requires the use of RSA 2048-bit signatures, but leaves other signatures as optional or up to the implementer [37]. This will not hold up in the quantum computing era and necessitates an update to the standard to protect against quantum threats. As research on other Internet protocols shows (e.g., Müller et al. [21]), this is a time-consuming process.

Future Work – We did not consider the impact of PQC algorithms on network transport requirements (bandwidth use, latency inflation due to additional round trips, etc.). Recent work by Cloudflare has shown that transport of larger signatures can cause additional delays when establishing TLS connections [38]. In FIdM scenarios, we not only need to consider the impact on TLS session establishment, we also need to consider the transport of SAML and OpenID Connect signature data encoded at the HTTP level, where both protocols encode signature data as Base64 data, which further inflates the size of the already large signatures created by PQC algorithms.

REFERENCES

[1] M. Kumar, "Post-quantum Cryptography Algorithms Standardization and Performance Analysis," *Array*, vol. 15, p. 100242, Sep. 2022. [Online]. Available: http://dx.doi.org/10.1016/j.array.2022.100242

[2] M. Aldosary and N. Alqahtani, "A Survey on Federated Identity Management Systems Limitation and Solutions," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 13, 2021.

[3] D. Pöhn and W. Hommel, "An Overview of Limitations and Approaches in Identity Management," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ser. ARES '20. New York, NY, USA: Association for Computing Machinery, 2020.

[4] G. Alagic, *Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process*, 2025. [Online]. Available: http://dx.doi.org/10.6028/NIST.IR.8545

[5] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, ser. SFCS-94. IEEE Comput. Soc. Press, 1994, p. 124–134. [Online]. Available: http://dx.doi.org/10.1109/SFCS.1994.365700

[6] "Post-Quantum Cryptography — CSRC — CSRC — csrc.nist.gov," https://csrc.nist.gov/Projects/Post-Quantum-Cryptography, [Accessed 2025-03-25].

[7] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone, *Status report on the third round of the NIST Post-Quantum Cryptography Standardization process*, Sep. 2022. [Online]. Available: http://dx.doi.org/10.6028/NIST.IR.8413-upd1

[8] "Kyber — pq-crystals.org," https://pq-crystals.org/kyber/, [Accessed 2025-03-25].

[9] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Dilithium — pq-crystals.org," https://pq-crystals.org/dilithium/, [Accessed 2025-03-25].

[10] "Falcon — falcon-sign.info," https://falcon-sign.info, [Accessed 2025-03-25].

[11] J.-P. Aumasson, D. J. Bernstein, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange *et al.*, "Sphincs+ — sphincs.org," https://sphincs.org, 2019, [Accessed 2025-03-25].

[12] "Final: OpenID Connect Core 1.0 incorporating errata set 2 — openid.net," https://openid.net/specs/openid-connect-core-1_0.html, [Accessed 2025-03-25].

[13] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, Oct. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6749

[14] "Security Assertion Markup Language (SAML) v2.0 — oasis-open.org," https://www.oasis-open.org/standard/saml/, [Accessed 2025-03-25].

[15] "XML Signature Syntax and Processing Version 1.1 — w3.org," https://www.w3.org/TR/xmldsig-core1/, [Accessed 2025-03-25].

[16] M. B. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, May 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7519

[17] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Assessing the Overhead of Post-Quantum Cryptography in TLS 1.3 and SSH," ser. CoNEXT '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 149–156. [Online]. Available: https://doi.org/10.1145/3386367.3431305

[18] J. Müller and J. Oupický, "Post-quantum XML and SAML single sign-on," Cryptology ePrint Archive, Paper 2024/828, 2024. [Online]. Available: https://eprint.iacr.org/2024/828

[19] F. Schardong, A. A. Giron, F. L. Müller, and R. Custódio, *Post-Quantum Electronic Identity: Adapting OpenID Connect and OAuth 2.0 to the Post-Quantum Era*. Springer International Publishing, 2022, p. 371–390. [Online]. Available: http://dx.doi.org/10.1007/978-3-031-20974-1_20

[20] N. Alnahawi, J. Müller, J. Oupický, and A. Wiesmaier, "A Comprehensive Survey on Post-Quantum TLS," *IACR Communications in Cryptology*, Jul. 2024. [Online]. Available: http://dx.doi.org/10.62056/ahee0iuc

[21] M. Müller, W. Toorop, T. Chung, J. Jansen, and R. van Rijswijk-Deij, "The Reality of Algorithm Agility: Studying the DNSSEC Algorithm Life-Cycle," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. ACM, Oct. 2020, p. 295–308. [Online]. Available: http://dx.doi.org/10.1145/3419394.3423638

[22] J. Goertzen and D. Stebila, *Post-Quantum Signatures in DNSSEC via Request-Based Fragmentation*. Springer Nature Switzerland, 2023, p. 535–564. [Online]. Available: http://dx.doi.org/10.1007/978-3-031-40003-2_20

[23] "Federation Architectures - eduGAIN - G&xC9;ANT federated confluence — wiki.geant.org," https://wiki.geant.org/display/eduGAIN/Federation+Architectures, [Accessed 2025-05-07].

[24] "surf.nl," https://www.surf.nl/files/2025-02/surf-conext-annual-report-2024.pdf, [Accessed 2025-05-07].

[25] eduGAIN, "eduGAIN membership status," https://technical.edugain.org/status, [Accessed 2025-05-13].

[26] NIST, "NIST Selects HQC as Fifth Algorithm for Post-Quantum Encryption," https://www.nist.gov/news-events/news/2025/03/nist-selects-hqc-fifth-algorithm-post-quantum-encryption, [Accessed 2025-05-13].

[27] "SSL Server Test (Powered by Qualys SSL Labs) — ssllabs.com," https://www.ssllabs.com/ssltest/, [Accessed 2025-03-27].

[28] "GitHub - open-quantum-safe/oqs-provider: OpenSSL 3 provider containing post-quantum algorithms — github.com,"

https://github.com/open-quantum-safe/oqs-provider, [Accessed 2025-03-27].
[29] "oqs-provider/USAGE.md at main · open-quantum-safe/oqs-provider — github.com," https://github.com/open-quantum-safe/oqs-provider/blob/main/USAGE.md, [Accessed 2025-05-07].
[30] "Protecting Chrome Traffic with Hybrid Kyber KEM — blog.chromium.org," https://blog.chromium.org/2023/08/protecting-chrome-traffic-with-hybrid.html, 2023, [Accessed 2025-03-25].
[31] "Defending against future threats: Cloudflare goes post-quantum — blog.cloudflare.com," https://blog.cloudflare.com/post-quantum-for-all/, [Accessed 2025-03-25].
[32] "Firefox Nightly 126.0a1, See All New Features, Updates and Fixes — mozilla.org," https://www.mozilla.org/en-US/firefox/126.0a1/releasenotes/, [Accessed 2025-03-25].
[33] A. P. Fournaris, G. Tasopoulos, M. Brohet, and F. Regazzoni, "Running Longer To Slim Down: Post-Quantum Cryptography on Memory-Constrained Devices," in *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, 2023, pp. 1–6.
[34] "XJTLU team sets code-breaking record for testing post-quantum online security - Xi'an Jiaotong-Liverpool University — xjtlu.edu.cn," https://www.xjtlu.edu.cn/en/news/2025/03/xjtlu-team-sets-code-breaking-record-for-testing-post-quantum-online-security, [Accessed 2025-03-28].
[35] Y. Chen, "Quantum algorithms for lattice problems," Cryptology ePrint Archive, Paper 2024/555, 2024. [Online]. Available: https://eprint.iacr.org/2024/555
[36] "About our project — openquantumsafe.org," https://openquantumsafe.org/about/#limitations, [Accessed 2025-03-27].
[37] B. Campbell, C. Mortimore, and M. B. Jones, "Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants," RFC 7522, May 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7522
[38] B. Westerbaan and L. Valenta, "A look at the latest post-quantum signature standardization candidates — blog.cloudflare.com," https://blog.cloudflare.com/another-look-at-pq-signatures/, Nov. 2024, [Accessed 2025-03-27].

APPENDIX

A. Ethical considerations

We follow community best practices by exclusively working with anonymized and aggregate data. Anonymization of the data was performed by SURF prior to sharing, and none of the URLs processed for our dataset contain identifiers of users, sessions or organizations connected to SURFCONEXT. Otherwise, the study does not raise any ethical concerns.