Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A survey on Cryptoagility and Agile Practices in the light of quantum resistance

Lodovica Marchesi [a,*], Michele Marchesi [a,b], Roberto Tonelli [a]

[a] Department of Mathematics and Computer Science University of Cagliari, Cagliari, 09124, CA, Italy
[b] Netservice spa, Galleria Guglielmo Marconi, 2, Bologna, 40122, BO, Italy

ARTICLE INFO

Keywords:
Agile methods
Cryptographic agility
Cryptographic algorithms
Quantum resistance

ABSTRACT

Context: Crypto-agility, a name that stems from agile methodologies for software development, means the ability to modify quickly and securely cryptographic algorithms in the event of a compromise. The advent of quantum computing poses existential threats to current cryptography, having the power to breach current cryptography systems.
Objective: We investigated whether and to what extent agile practices for software development are suited to support crypto-agility, or not. In particular, we discuss their usefulness in the context of substituting current algorithms with quantum-resistant ones.
Method: First, we analyzed the literature to define a subset of 15 agile practices potentially relevant to cryptographic software development. Then, we developed a questionnaire to assess the suitability of agile practices for obtaining crypto-agility. We performed a Web search of relevant documents about crypto-agility and quantum resistance and sent their authors the questionnaire. We also sent the questionnaire to cybersecurity officers of four Italian firms. We analyzed and discussed the responses to 32 valid questionnaires.
Results: The respondents' affiliations are evenly distributed between researchers and developers. Most of them are active, or somehow active, in quantum-resistant cryptography and use agile methods. Most of the agile practices are deemed to be quite useful, or very useful to get crypto-agility, the most effective being Continuous Integration and Coding Standards; the least appreciated is Self-organizing Team.
Conclusion: According to researchers and developers working in the field, the safe transition of cryptographic algorithms to quantum-resistant ones can benefit from the adoption of many agile practices. Further software engineering research is needed to integrate agile practices in more formal cryptographic software development processes.

1. Introduction

Agile methodologies were introduced in the late 1990s under the name of "lightweight" or "flexible" processes. The term "agile" proper was introduced in the 2001 Agile Manifesto [1], which had a profound impact on software engineering and project management. In a short time, "agile" became a buzzword used wherever the ability to respond quickly and effectively to challenges and changing requirements was highlighted.

A few years after the Agile Manifesto, the term "cryptographic agility" or "crypto-agility" was introduced in the cybersecurity community. This term was first defined in a 2006 article by LaMacchia and Manferdelli [2], which introduced Microsoft's new core cryptographic API, "Crypto Next Generation" (CNG). CNG was claimed to have "cryptographic agility".

In this context, crypto-agility means the ability to modify CNG cryptographic algorithms (CA) and related data (parameter settings, key storage, etc.) in the event of a compromise, quickly and securely. An analysis of 269 cryptographic software failures reported from January 2011 to May 2014 showed that only seven vulnerabilities were due to faults in cryptographic primitives, among which four were due to insufficient key length [3]. So, practical cases in which CA has to be changed quickly to avoid damage due to their compromise are rare.

However, the advent of quantum computing constitutes a paradigm that poses new challenges to cryptography because quantum computers (QCs) of sufficient power have been proven to be capable of breaking several important CAs currently in use [4].

There are important milestones to be achieved before there can be a marketable QC that can truly be used to solve real-world problems.

* Corresponding author.
E-mail address: lodovica.marchesi@unica.it (L. Marchesi).

https://doi.org/10.1016/j.infsof.2024.107604
Received 25 January 2024; Received in revised form 11 September 2024; Accepted 14 October 2024
Available online 21 October 2024

The most optimistic experts estimate that it will take 5 to 10 years to build a viable QC. The most prudent expect 15 to 30 years [5].

The main reason for the introduction of agile principles and practices at the end of the 90 s was the inability of traditional software engineering processes and practices to meet the needs of software production in the internet age. The need to deliver a working system in a very short time, starting from unclear and changing requirements was at odds with the traditional waterfall approach, where requirements and design of the system must be very accurate, and precede the programming, testing and release activities [6].

Agile and Lean software development were introduced to shorten development times and accommodate changes, without compromising on quality. Its time frames, depending on specific activities, vary from hours to days or weeks [7].

Crypto-agility means the ability to react to CA changes effectively and timely. However, the need to protect a system from quantum attacks is likely not to be mandatory before the end of this decade, thus its time frame is of the order of years or even decades [8]. Thus, crypto-agility to obtain quantum resistance operates in time frames completely different from those addressed by agile methods.

A recent large-scale international survey about software development methods and practices that improve or tame agility found that “Most project disciplines show a clear trend towards an agile implementation” [9]. This research shows that more than 45 percent of the overall respondents use or mainly use agile practices, whereas about 25 percent use a balanced approach between agile and traditional, as reported in Fig. 8 of the paper. The project disciplines where agile is used most are Implementation/Coding and Integration and Testing, but also in Project Management, Change Management, Architecture & Design, and Maintenance & Evolution most respondents said that they use an agile approach, or at least a balance one.

All the mentioned project disciplines are very relevant in designing, coding, testing and deploying cryptographic software, so it is crucial to assess the degree to which the relatively new agile approach is used by engineers working in CA software development, and how they feel agile practices can help to obtain crypto-agility.

In particular, we asked ourselves if and what agile principles and practices can be successfully used specifically to effectively address the development and adoption of quantum resistant software, which requires to redesign and rewrite a consistent part of present CA software, taking into account a resource usage much higher than in traditional CA software [10]. To address these issues, we formulated the following research questions, that will be answered in this paper:

RQ1. *Which are the most (and least) suitable agile and lean practices to support crypto-agility?*

RQ2. *Which crypto-agility characteristic can benefit more from the agile approach?*

RQ3. *Are there differences between how agile practices are viewed by people working on quantum resistance algorithms and people just involved in crypto-security?*

To answer these questions, we first analyzed existing work on crypto-agility, about the most popular agile practices (APs), and about the links between agility and cybersecurity. According to the study of the background, we found a precise definition of crypto-agility and selected the most common agile practices for system design and development. We did not consider neither specific agile methods, nor practices devoted to manage the agile team. We then highlighted the issues and needs related to quantum resistance (QR).

The tool devised to carry out our research is a questionnaire sent to researchers and developers active in the field of QR, and of cybersecurity in general. The main contributions of our work are:

1. A thorough literature analysis about the most relevant agile practices useful for cryptographic software development, and about the definition of crypto-agility.
2. A survey among 32 researchers and developers active in cybersecurity, divided into people active, somehow active, and inactive in quantum resistant cryptographic software development. To our knowledge, no survey on this subject has been made before.
3. Findings and discussion about the suitability of the agile approach to support crypto-agility, including the most (and least) suitable agile and lean practices to support it.
4. Findings about how the opinions about agile practices of people working on quantum resistant cryptography differ from those of people not working on that.

The rest of the paper is organized as follows: Section 2 provides a review of existing work on crypto-agility, APs, and the application of APs in the context of cybersecurity; Section 2.2 presents the concept of crypto-agility, highlighting its main properties; Section 2.5 explains the challenges of QC to cybersecurity, and the concept of QR. In Section 3 we describe how we designed and performed our research using a questionnaire; Section 4 reports and discusses the results of the questionnaires collected. Lastly, Section 6 discusses the threats to validity of our study and Section 7 concludes the paper.

2. Related work and definitions

2.1. Origin and key works on crypto-agility

The term “crypto-agility” first appeared in 2006 in [2]; after that date, the term is sometimes used in technical reports of standard working groups (IETF, ETSI, NIST). Only from 2018 onwards, also due to the efforts to choose and standardize quantum resistant algorithms, which were announced at PQCrypto conference in 2016 [11], the term has been used and discussed in many forums. The related wikipedia page also dates back to November 2018.

In 2019 Grote et al. described the strategy of post-quantum cryptography and crypto-agility. They reviewed the proposed quantum resistant algorithms and highlighted the need for crypto-agility to effectively replace non-quantum resistant CAs with quantum resistant ones [12]. In 2021 Mashatand and Heintzman recommended a crypto-agile process to assess and mitigate the exposure of organizations to quantum attacks. The proposed process includes steps such as *Determine Transition Path*, *Wait for Standardization*, *Invest in Crypto-agility, Establish and Maintain a Quantum-Resistance Roadmap*, *Implement Hybrid Cryptography* [13]. In the same year, Ma et al. proposed CARAF: Crypto Agility Risk Assessment Framework [14]. CARAF is aimed at analyzing and evaluating the risk that results from the lack of crypto agility, in order to determine an appropriate mitigation strategy commensurate with the risk tolerance. The application of this framework was demonstrated with a case study regarding QR.

Zhang and Miranskyy analyzed the threats posed by QCs and compared the related mitigation strategies to those used to address the Y2K bug [15]. They proposed a road map for software developers to address encryption-related challenges associated with quantum attacks, especially using crypto-agility.

In 2022, Holm et al. proposed the Crypto-Agility Maturity Model (CAMM) to determine the state of crypto-agility of a given software or IT landscape and to improve it [16]. CAMM consists of five levels named: (0) Initial/Not possible, (1) Possible, (2) Prepared, (3) Practiced, and (4) Sophisticated. For each level, a set of requirements is formulated based on literature review. Initial feedback from field experts confirmed that CAMM has a well-designed structure and is easy to understand.

Alnahawi et al. performed a literature survey on crypto-agility and discussed respective development efforts categorized into different areas, including requirements, characteristics, and possible challenges [17]. They included a case study on crypto-agility in the context of PQC Integration for eCards. Interestingly, these authors claim that "*we still need cryptographic libraries following crypto-agile design principles.*". They stress the need to develop a common understanding of crypto-agility, including definitions, goals, and terms in order to support the discussion and place the various contributions on common ground.

Finally, in 2023 the authors published a paper addressing the same issues of this paper and performing a preliminary analysis of the relationships between agile and lean practices and crypto-agility principles and features, and of the suitability of the former to obtain the latter [18].

2.2. Definition of crypto-agility

Crypto-agility is strictly related to cybersecurity, but its scope is much narrower. As reported above, it regards the ability to easily change the algorithm implementations used by cryptographic protocols and to provide a high level of abstraction by the API for core cryptographic operations [2].

A specific definition of crypto-agility was provided by Kerry McKay of National Institute of Standards and Technology (NIST) during the Cryptographic Agility and Interoperability Workshop held in 2017, see [19], page 19–20:

1. the ability for machines to select their security algorithms in real time and based on their combined security functions;
2. the ability to add new cryptographic features or algorithms to existing hardware or software, resulting in new, stronger security features;
3. the ability to gracefully retire cryptographic systems that have become either vulnerable or obsolete.

A more detailed definition of the key properties of crypto-agility were provided by Mehrez and El Omri [20]. They state that *"crypto-agility is the ability of a system to migrate easily from one CA to another, in a way that is flexible, scalable, and dynamic"*. The most important crypto-agility properties among those reported by them are:

- **Extensibility**: ability to add new algorithms or new parameters to the system as efficiently as possible. This is property 2 of McKay's definition.
- **Removability**: ability to gracefully retire cryptographic systems that have become vulnerable or obsolete. This is property 3 of McKay's definition.
- **Fungibility**: ease to swap security components; also, the ability for machines to select their security algorithms in real time and based on their combined security functions. This is property 1 of McKay's definition.
- **Interoperability**: Crypto-Agility solutions must be interoperable between independent implementations based purely on the information provided in the specification.
- **Updateability**: support of secure updates or patches of CAs in the system.
- **Compatibility**: if we replace software on a system, the new software modules and patches should be able to operate on the same hardware.
- **Reversibility**: if any software update fails, the system should be able to return to the previous working software version.

The object of crypto-agility is to facilitate the development or updating of software modules and systems that use CAs. These systems are typically used for secure data transmission and for the storage and retrieval of encrypted data [21]. Another field where CAs are massively used is that of blockchain or digital ledger technology (DLT) [22].

The CAs which are actually used follow standards enacted by various organizations, among which the most prominent is NIST. There are standard CAs for symmetric and asymmetric encryption, hash functions, key management. The standards include detailed specification of the API of CA libraries, so that systems using different libraries implementing the same CA can exchange encrypted data, provided that these libraries follow the API specifications [23].

Typically, large, regulated organizations are more impacted by changes in CAs than small ones. Since CAs are largely regulated, including their APIs, the software that actually needs to be written is the software that uses these CAs to encrypt, send or store data, and decipher them. An example of *ad hoc* cryptographic development is the choice of applying more than one CA in sequence, to get a higher security level, to send the encoded data together with the information on how it was created, and to decode it [24].

Summarizing these considerations, our definition of CA builds on the three characteristics of McKay's original definition reported above, adding as fourth characteristic the ability to securely upgrade CAs:

4. the ability to upgrade already installed cryptographic features or algorithms without introducing unwanted side-effects or vulnerabilities.

2.3. Agile practices

Agile software development is defined by the Agile Manifesto, which describes 4 general values, that can be put into practice by applying 12 principles [1]. These principles, in turn, can be applied by using a set of software development practices called "Agile Practices" (APs).

Every agile methodology prescribes, or suggests, a set of APs. Overall, at least 35 APs have been proposed, as reported in 2011 by Sletholt et al. [25]. In their paper, these authors summarized the known APs and then performed a literature review, resulting in the examination of 10 software projects employing APs. Among the 35 APs, 11 were adopted by the majority of the teams, whereas Pair Programming was explicitly excluded by most projects but one.

In 2016 Licorish et al. surveyed 184 practitioners working in Brazil, Finland and New Zealand, with the aim of understanding the adoption and use of software development methods and practices [26]. They found that most respondents used agile methods, as for instance Scrum, Kanban, Feature Driven Development and Extreme Programming. They also surveyed the use of 17 APs among the practitioners, finding that only Pair Programming, On-Site Customer and Planning Game were used by less than half of respondents.

Henriksen and Pedersen published a paper in 2017 aiming to find the most important APs a team should use to succeed in an agile software project [27]. They interviewed the members of two software development teams in Norway. The considered APs, all aimed to project management, were divided in three groups – "Scope" focusing on the management of the functional elements delivered, "Time" referring to the amount of time required to complete the project, and "Quality" regarding APs designed to increase product quality. Overall, they considered 53 APs,of which 23 were found to be heavily used. 15 APs were considered especially relevant to get project success and were classified by the authors in three groups in decreasing order of importance. The especially important APs are: *Iterative development, Sprint review, Incremental design, Sprint retrospective, Improve collaboratively*. The second group of APs in order of importance is composed of: *Stories, Product backlog, Sprint planning, Task board, Visualize workflow, Daily stand-up, Product owner*. The three less important (but still relevant) APs are: *Sprint backlog, Team based estimation, Whole team*.

Vallon et al. published in 2018 a systematic literature review on agile practices in global software development, that is in projects carried on by multiple teams located in various part of the globe [28]. They considered 145 studies published from 2002 to 2016. From these studies, they extracted 21 APs that have been successfully applied in at least five cases, in decreasing order of usages.

Sandstø and Reme-Neiss conducted two literature reviews, the first aiming to identify commonly reported APs, while the second focusing on these Agile practices' reported effects [29]. They clustered similar APs, and filtered out rarely mentioned APs. In the end, they found 12 APs representative of one or more original clustered APs, and discussed they relevance to project success.

The last paper we considered is a recent study carried out in 2022 by Ghimire and Charters [30]. They collected and analyzed data from Agile software development teams and identified the APs that could have higher impact on the communication in the team, project requirements and project priorities Seventy-three responses were analyzed representing fifty organizations. They identified 19 APs, among which 6 are the most used.

2.4. *Agility and cybersecurity*

There are several studies on how agile practices can be made compatible with security assurance, starting from the seminal work of Bezsonov [31] on merging XP and security practices. Among these studies, two main areas emerge: updates to Scrum and in general to agile processes to manage security aspects, and user stories and other specific APs to specify security requirements.

Regarding the first area, Fitzgerald et al. identified the main incompatibility issues between agile characteristics and constraints imposed by regulated environments and illustrated through a detailed case study how an agile approach can be implemented successfully in a regulated environment [32]. Ghani et al. suggested adding Security Backlog and the role of a Security Master in Scrum [33]. Othmane et al. proposed a method to ensure the security of software increments, integrating security engineering activities into the agile software development process [34].

Two papers propose additions to Scrum method to integrate security activities into Scrum process. ScrumS by Esteves Maria et al. adds to Scrum lifecycle-specific security techniques for project management and risk analysis, in order to make it able to generate safer software [35]. The second, a secure Scrum process by Maier et al. orchestrates security engineering activities within the Scrum development process to achieve both security and agility [36].

Lastly, let us mention a paper by de Vicente et al. proposing a New Secure Software Development Life Cycle (S-SDLC) which prioritizes security aspects at any phase of the software life cycle and takes advantage of the benefits of the agile models [37].

Regarding security requirements, Kongsli proposes the concept of "misuse story", derived from the "misuse case", and reports on experiences with the use of misuse stories and automatic security tests in the development of web applications [38]. Williams et al. suggested the Protection Poker game to find the relative security risk of each requirement [39]. For a recent and comprehensive survey of security in agile software development, we refer the reader to the work of Rindell et al. [40].

The proposed security processes and practices primarily aim to protect software systems, and in particular Web systems, against multiple forms of cyber attacks. In this paper, we are interested to practices related to the development and integration of cryptographic methods and tools related to asymmetric cryptography, document encryption and decryption, digital signatures, key storage and the like. This is the field of interest of crypto-agility, which is only one of many aspects of cybersecurity, and not even the most important.

We can add that almost all studies and proposals on agility and cybersecurity are quite theoretical. A recent systematic mapping study performed by Moyon et al. addressing security compliance in agile software development sorted 11 articles from a starting set of 2,383 papers and analyzed them in detail [41]. None of these papers dealt with actual experiences in real development projects!

Moreover, none of the papers on crypto-agility reviewed in Section 2.1 makes use of or refers to secure agile methods or practices, as proposed in the papers of Section 2.4.

For these reasons, we decided not to include agile security practices in the survey about the relationships between agility and crypto-agility reported in the followings.

2.5. *Quantum computing and quantum attack*

Quantum computing represents a significant breakthrough in computer science and will have a strong impact on many fields, such as science, finance, artificial intelligence, pharmacology, and many others [42]. Unfortunately, it also has the power to breach current cryptography systems that enable secure Internet communications, digital signatures, digital currencies, and DLT.

The cryptography used in these systems is composed of:

1. *Hash functions*, which guarantee immutability of data, and in the blockchain realm are also applied in proof of work algorithms, and to generate blockchain addresses from a public key.
2. *Symmetric cryptography*, used to encode and decode information sent through a public channel, or that is stored in a repository but must remain confidential.
3. *Asymmetric cryptography*, which is behind SSL/TLS protocol ensuring secure Internet communication, and in DLT guarantees the property of the assets linked to an address, or the identity of the person who sends a transaction, thus ensuring trustfulness.

No classic computer in existence is capable of performing calculations fast enough to reverse this math in any usable time frame. However, the advent of QC constitutes a new paradigm which poses new challenges to cryptography because it has been proven that robust QCs will be able to break several important CAs currently used. Important milestones still remain before a marketable and truly usable QC can exist to solve real-world problems. Experts estimate that it will take 5 to 30 years from now to build a viable QC [5].

When QCs will become operational, the currently understood main menace they will pose to cryptography is based on Shor's algorithm for quickly factoring the product of two very large primes [43]. This algorithm will be usable when a sufficiently powerful QC is available and will allow one to unhinge the RSA algorithm, and with a small variant also the ECDSA algorithm. These algorithms are currently the most used for asymmetric cryptography. Today, a digital computer that uses the most efficient algorithm known would carry out the factoring of a number of 300 digits (1024 bits) in about 150,000 years [44]. A QC using the Shor algorithm would find the solution in seconds.

Another threat to unhinge symmetric encryption algorithms (AES, DES, hash algorithms, and many others) is Grover's algorithm, which allows you to speed up the search for possible solutions. Grover's algorithm can reduce the difficulty of any search problem from n to $\sqrt{n}$ [45].

However, the performance of Grover's algorithms is not as innovative and dangerous as for Shor's one because it can be easily countered by doubling the length of the encryption key. Additionally, hash functions are continually evolving for increased security. For example, if QCs evolve to the point of posing a threat to SHA-2, then SHA-3 is already standardized as an alternative that offers a higher level of security in the NIST standard FIPS 202 [46].

2.6. *Quantum resistance*

Luckily, several CAs that are quantum resistant already existed in the nineties, and more have been introduced after the discoveries of Shor and Grover. Standardization bodies such as NIST and ETSI (European Telecommunications Standards Institute) have been working on standard quantum resistant algorithms for almost a decade, and a set of international standards is expected in 2024 or 2025 [13].

Unfortunately, substituting traditional CAs with quantum resistant ones is a non-trivial problem because QR algorithms are much less efficient than traditional algorithms, requiring much more memory and more computation time. So, adopting quantum resistant CAs is not a simple replacement of an algorithm with another, but the replacement of one with another that has different characteristics, and which requires many more hardware resources.

This problem is typically not as important for traditional computers, which have large amounts of memory and very fast CPUs. Instead, the problem arises for devices with limited computing power, such as mobile terminals, IoT devices and network management hardware, which usually work using programs embedded in the firmware and have limited and not easily expandable memories. Everything that is installed on firmware risks requiring machine replacement.

One of the main issues faced by organizations to be prepared for quantum menace, is the fact that CAs are ubiquitous in the software and hardware systems used. Therefore, migrating to quantum resistant solutions will not be an easy journey. To this end, an organization can take huge advantage of a process that exhibits the properties of crypto-agility as defined in Section 2.2.

The road-map to systematically upgrade existing computer systems to obtain quantum resistance is still in its initial phases, because the standardization of CA has not yet fully ended. The steps involved are [47]:

1. Study the existing and the pre-selected QR algorithms and the available software libraries implementing them. This preliminary step is crucial, also because during the QR algorithms selection process, some of those already selected have been compromised, like for instance CRYSTALS-Kyber [48] and Rainbow [49].
2. Once the standard CA will be selected by NIST, which is forecasted by Summer 2024[1] it is needed to develop/tune a set of QR and traditional CA modules, able to be used interchangeably, including software modules able to run on firmware, using as few resources as possible.
3. The software using these libraries and QA must be modified, adding the ability to transparently and securely switch among CAs, depending on the security level needed.
4. This new software must be installed with the minimum possible impact on functioning systems.
5. We must be prepared to intervene promptly and efficiently to resolve any problems and unforeseen side effects.

Agile practices that can be useful to get crypto-agility, specifically for upgrading CA software to quantum resistance are all the practices that can improve software reliability, security and ease of updating. Among these APs we may cite the use of an automated testing suite, simple design, refactoring, coding standards, continuous integration, and others.

The following sections are aimed to shed light on how people involved in QR software development, and generally in security-related software consider the use of APs to obtain crypto-agility, especially in the context of quantum resistance.

3. Method

The context is the development of software that must strictly follow standards regarding its CA and API, and that uses standard libraries to send, store, and receive encrypted data, to decode them, to manage keys, and to combine CAs to obtain stronger security or to manage digital signatures and secure ownership of digital assets. Accordingly to a crypto-agile characteristic, this software should also be able to automatically choose the "right" CA, and manage the updating of CA libraries.

In detail, the research was designed and conducted accordingly to the following steps:

1. Setting of the research goals and specifying of the research questions, as reported in the introduction.
2. Designing the research, identifying the sending of a questionnaire to researchers and developers active in the field of crypto-agility and in general in the development of cryptographic software as the tool to find if and how agile practices can be used to obtain crypto-agility, answering the research questions.
3. Performing a literature review – already reported in Section 2 – to assess:
   (a) how crypto-agility was introduced and how it is defined;
   (b) what are the APs most used and reputed among developers;
   (c) what are the relationships between APs and cybersecurity, and in particular between APs and the development of cryptographic libraries;
   (d) how critical is the threat posed by quantum computers to cryptography, and what are the strategies to address and mitigate this threat.
4. Designing the questionnaire to be sent by email, trying to keep it as short as possible to encourage people to fill it out. More specifically this led to the choice of the demographic questions needed to assess the experience and activity of respondents, and to the definition of the most popular APs to be presented in the questionnaire.
5. Designing the strategy to find people with knowledge of both cybersecurity procedures and algorithms, and of software development processes.
6. Defining exclusion criteria for questionnaires.
7. Defining the criteria to analyze the answers, discriminating among various groups of respondents, including choosing the one-tailed Mann–Whitney U-Test [50] to assess the statistical significance of results.
8. Answering the research questions, and discussing the threats to the validity of the answers.

3.1. The questionnaire: demographics and processes

We prepared a questionnaire to be answered by professionals and researchers active in cybersecurity. At the beginning, we ask some demographic questions to differentiate among respondents according to their age, location, employer, experience, and possible involvement in quantum resistant CA studies.

Then, we ask which is the software development process they follow, allowing more than one answer. This to understand the degree of "agility" of the processes they follow to develop software.

Overall, we considered the following processes:

- Waterfall (sequential steps), covering traditional approaches.
- Iterative (Spiral, RUP, …), covering more recent, yet traditional approaches introducing iterations to correct and improve the strictly sequential approach of the waterfall.
- Scrum/Secure Scrum. Scrum is by far the most used agile methodology; Secure Scrum is a variation explicitly taking in account security requirements, see Section 2.4.
- Kanban board, another very popular Lean/Agile method.
- Other agile/lean process, to cover agile processes which cannot be clearly defined as Scrum or Kanban.
- Microsoft SDL, one of the most popular security-related process.
- OWASP/CLASP, another popular approach which includes an activity driven, role-based set of security-related guidelines and practices for building security into software development.
- BSIMM, which is not a process but a set of guidelines to assess the maturity level of an organization from the cybersecurity point of view.
- No defined process, to consider also organizations not having or not reporting a defined process for producing software.

[1] See: https://csrc.nist.gov/Projects/post-quantum-cryptography/workshops-and-timeline

3.2. The questionnaire: agile practices

Regarding the APs to consider in the questionnaire, we had to choose among all practices mentioned in the six studies cited in Section 2.3. Note that paper [27] cites 53 APs, but provides an evaluation of only the 15 APs considered especially relevant, so we considered only these 15 APs for paper [27].

First, we standardized the names of APs, choosing just one telling name for each AP. For instance, the AP "*Development using time-boxed iterations*" was called "*Time-boxed sprints producing potentially shippable output*" in [25], "*Iteration*" in [26], "*Iterative development*" in [27], "*Sprint/iterations*" in [28], "*Sprints*" [29] and "*Sprint/Time box*" in [30].

We also put together as a single practice guidelines, actions or roles intrinsically linked to it. For instance, the AP "*Incremental development driven by requirements expressed as features or user stories*" includes the two practices "*The Project Velocity is measured*" and "*User stories are written*" of Ref. [25], the three practices "*Product backlog*", "*Iteration backlog*" and "*Iteration planning*" of Ref. [26], and the four practices "*Incremental design*", "*Stories*", "*Product backlog*" and "*Sprint backlog*" of Ref. [27].

After these standardization and clustering, we had 35 APs mentioned in at least one of the above quoted papers. For each of these AP we collected the evaluations of their popularity and/or effectiveness in the quoted studies, according to a scale ranging from -XXX (not useful at all) to 0 (neutral), to XXX (very useful). For positive scores, we also considered "X+" and "XX+" scores.

This evaluation led to find 18 APs with the highest scores. Among them, we discarded the following three APs:

- "*Release planning to release product increments*", because it has to do more with delivering software applications to a customer rather than to develop cryptographic libraries or security modules.
- "*The customer is always available*", because it has to do with getting requirements from future users of the system, which is not the case for CA development or security.
- "*Give the team a dedicated open work space*", which we considered not relevant to our case, and which is less and less applied in times of teleworking and global software development.

All remaining 17 APs were quoted only by one of the six studies considered, and were consequently discarded.

In the end, we identified fifteen key APs reported and appreciated in most, if not all, of the studies cited in Section 2.3. Table 1 shows these APs and their description.

We include as supplementary material a spreadsheet where all APs reported by the 6 studies quoted above are shown, and are compared with the fifteen APs chosen in this paper.

In addition to the literature review on agile practices, we also consulted some cryptographic software developers, to find out if and which agile practices are actually used by them. Note that most cryptographic library development projects are currently focused on QR algorithms, and declare to pursue crypto-agility.

Some projects, such as Bouncy Castle [51], and Open Quantum Safe [52] use many of the APs mentioned, while others such as Botan [53] and WolfSSL [54] use only some of them. The Mozilla Network Security Services group [55], instead, only declares the use of TDD.

The most used APs in these projects are automatic testing, continuous integration, collective code ownership. The APs used very little, or not at all, are time-boxed iterations, pair programming and daily meetings. All other APs are used just by a couple of the mentioned projects.

For each of the 15 chosen APs, the questionnaire gives 4 options to evaluate their effectiveness to support crypto-agility, from a minimum of "Irrelevant" to a maximum of "A lot". The intermediate options are "Not much" and "Quite". We also allow the respondent to choose "I don't know" option, and to add comments.

The second and last part of the questionnaire somehow reverse the problem, and asks how much obtaining the four definitions of crypto-agility according to McKay presented in Section 2.2 can be helped by APs. The respondent has the same four options quoted above, plus the "I don't know" option and the capability to add comments.

The questionnaire was conceived according to the following guidelines: (i) The questionnaire was kept as short as possible, to mitigate respondents' time limitations. (ii) In the case of answers that give a merit score, we used a four-point Likert scale, avoiding the "neutral" option. (iii) As reward to increase the wish to compile the questionnaire, we offered to send a copy of the pre-print of this paper. (iv) We guaranteed the anonymity of the questionnaire to limit social desirability bias.

Once the questionnaire was completed, we conducted a survey pre-test, by having three Ph.D. students and two developers from partner companies complete the questionnaire, using their feedback to make adjustments and corrections. We also asked two colleagues expert in software engineering and security to evaluate and discuss with us the questionnaire. The above quoted guidelines to conceive and validate the questionnaire are consistent with some of those reported by Ghazi et al. [56]).

3.3. Choosing the sample

We looked for a sample of researchers and professionals with knowledge of both cybersecurity procedures and algorithms, and of software development processes. It is practically impossible to get a "representative" sample of these people, also because it would be difficult to define what "representative" means. So, we resorted to *convenience* sampling, that is selection based on accessibility [56]. The population search plan was designed according to the guidelines reported by De Mello and Tarassos [57].

After publishing the questionnaire on the Web[2] we looked for email addresses which to send an email to, asking for answering the questionnaire. To find these addresses, we followed two approaches. The first was to find the emails of the authors of articles and technical reports on the topic, the second to invite to answer to the questionnaire developers working on cybersecurity in 4 companies with which we collaborate on research projects.

We downloaded PDF files with relevant documents about crypto-agility and quantum resistance listed on Google Scholar. We excluded the papers published before the year 2016, that is the year when the need to find and standardize quantum-resistant CAs was formalized. As selection criteria, we formulated the following queries that yielded the number of documents shown:

- "crypto agility" OR "cryptoagility" OR "crypto-agility": 179 results;
- "crypto agility" post-quantum cryptography: 111 results;
- post-quantum cryptography phyton: 3 results;
- quantum resistance standard: 24 results
- agile "quantum resistance": 1 results
- agile quantum resistance: 4 results
- crypto agility quantum: 25 results

The search found 171 documents. The sum of the reported results above is greater than 171 because several documents were found in more than one query.

We then wrote a Python program able to scan all .PDF files in a folder, generating a .CSV file with all the email addresses found in these files. We checked and corrected some wrong or generic email addresses

[2] https://forms.gle/aEcSM2ZK8ScbnCG36

Table 1
The Agile Practices considered in our study.

AP	Description	Detailed description	Refs.
AP01	Development using time-boxed iterations	The project proceeds through short, time-boxed iterations, which in Scrum are called Sprints. A planning meeting is held at the beginning of the iteration.	[25]-[30]
AP02	Incremental development driven by requirements expressed as features or user stories	The features are also called Stories or Minimum Marketable Features (MMF). They constitute the product backlog, and are elicited through requirements workshops or estimation meetings. The project proceeds by implementing a set of features at each iteration.	[25]-[30]
AP03	Continuous integration (performed daily or at regular intervals)	Code changes from multiple contributors are merged to an integration branch using automated configuration system. Usually this is after each commit or periodically like once a day. Once built, all tests should run.	[25] [26] [28]-[30]
AP04	Team members volunteer for tasks (a self-organizing team approach)	Work is not assigned to developers by a project manager, but is chosen by them; the PM intervenes only in case of disagreement.	[25] [27] [29]
AP05	Use of boards (task/kanban boards, burndown charts, etc.) to visualize the project's status	This AP includes the use of Kanban boards, burndown charts, and other visual tools, so that the team is made aware of the status of the project. In case of distributed development the board can be virtual.	[25] [27]-[30]
AP06	Use of an automated test suite	Unit and functional tests are written in code and can be run as many times as needed.	[25] [28] [30]
AP07	Writing tests before writing the code (Test Driven Development, TDD)	Automated unit tests are written even before writing the tested code. This is also a powerful design tool, and ensures that unit tests are always written.	[25] [26] [28]
AP08	Collective code ownership	Each team member can update any portion of the code; the code is not owned by a specific developer.	[25] [26] [28]
AP09	Prioritization of simple design and simplicity	Implement only the features that are strictly needed. Do not plan for future possible uses. Less software means fewer defects and less cost.	[25] [26] [28]
AP10	Regular refactoring to improve code quality	Code refactoring is the process of restructuring existing computer code without changing its external behavior. It is intended to improve the design, structure, and implementation of the software, while preserving its functionality.	[25] [26] [28] [30]
AP11	Adherence to coding standards and creation of intention-revealing code	All team members must approve coding standards and adhere to them. The code should be self documenting, and comments should be kept to a bare minimum.	[25] [26] [28]
AP12	Pair programming, particularly for critical sections of code	All or part of the code is produced by pairs of developers working together.	[25] [26] [28]-[30]
AP13	Daily meetings for progress assessment and issue resolution	Also called *Stand-up meeting* or *Daily Scrum*. These are short meetings (of about 15') to assess project status and highlight issues and blocks.	[25]-[30]
AP14	Iteration review meetings	Meetings held at the end of each iteration to evaluate and accept iteration results.	[25] [27] [28]
AP15	Conducting retrospectives	Retrospective meetings held after the review meeting, aimed to inspect and adapt work process. These meetings are typically held every 1-3 iterations.	[25]-[30]

and found 377 unique email addresses related to the downloaded documents.

We sent an email to all of them inviting them to respond to the online questionnaire we prepared. 83 emails – about 22% – was rejected from their mail server because the address was no longer valid, or because of the spam filter. In the end, we received 23 valid responses from 294 emails delivered, with a response rate of 7.8%. This rate is quite low, but it is not uncommon. A recent paper on survey response rates reports that about 7% of the studied surveys gets a rate lower than 10% [58]. Considering the way in which we obtained the email addresses to send the questionnaire, the response rate was within our expected range for non-probabilistic sampling.

To enlarge the sample, we also sent the email to the cybersecurity officers of four Italian firms our Department cooperates with, asking them to make their developers answer the questionnaire. We got 10 more answers to our questionnaire, for a total of 33.

The exclusion criterion was to discard questionnaires with too few answers, or whose authors explicitly stated they are not knowledgeable on the subject. In practice, we had to discard just one questionnaire with 14 over 15 "I don't know" answers to question 7, and 3/4 "I don't know" answers to question 8. In the end, we analyzed 32 valid questionnaires.

4. Results

4.1. Demographics of respondents

We have 13 answers from academics and researchers, and 19 from the industry, so there is a good balance between them. Most answers coming from industry come from ICT firms. Two of them come from "other firms", and in one the respondent defines her/himself as "professional/freelance".

In the following, we will denote with "***researchers***" the respondents whose affiliation is a university or a research body, and with "***developers***" all other respondents.

Table 2
Age (years), experience (years) and affiliation of respondents, as a function of their involvement in QR software. U= University, F= Firm.

Involvement in QR	Age				Experience				Affiliation		Total
	20–29	30–39	40–49	50+	0–1	2–5	6–10	11+	U	F	
Active	0	5	4	3	1	2	3	6	7	5	**12**
Somehow active	2	1	4	4	2	4	2	3	6	5	**11**
Inactive	2	3	0	4	5	0	0	4	0	9	**9**
Total	4	9	8	11	8	6	5	13	13	19	**32**

Table 3
Number of respondents for each answer option, for all considered agile practices.

	AP01	AP02	AP03	AP04	AP05	AP06	AP07	AP08	AP09	AP10	AP11	AP12	AP13	AP14	AP15
QR-active															
Don't know	5	2	3	2	3	1	2	5	0	0	0	4	0	0	2
Irrelevant	0	0	0	4	3	0	2	3	1	1	0	0	3	2	1
Not much	5	5	2	9	4	6	5	7	3	4	5	3	9	1	3
Quite	12	11	7	4	8	6	9	4	10	7	6	12	4	14	11
A lot	1	5	11	4	5	10	5	4	9	11	12	4	7	6	6
Average score	1.78	2.00	2.45	1.38	1.75	2.18	1.81	1.50	2.17	2.22	2.30	2.05	1.65	2.04	2.05
QR-inactive															
Don't know	0	0	0	1	2	0	1	0	0	1	1	2	0	0	1
Irrelevant	2	1	0	1	0	0	0	0	1	0	1	1	2	1	1
Not much	0	2	4	4	0	1	1	3	1	2	0	1	3	2	3
Quite	5	3	0	3	3	2	4	3	2	4	5	2	3	2	3
A lot	2	3	5	0	4	6	3	3	5	2	2	3	1	4	1
Average score	1.78	1.89	2.11	1.25	2.57	2.56	2.25	2.00	2.22	2.00	2.00	2.00	1.33	2.00	1.50
Total average	1.78	1.97	2.34	1.34	1.96	2.29	1.93	1.67	2.19	2.16	2.23	2.04	1.56	2.03	1.90

The ages of the respondents vary, but most of them are quite senior. About one third of them are over 49, and about 60% are over 39. Only 12.5% are under 30.

Almost all the respondents live in Europe, with only two in North America, and two in Asia (East and South). However, we deem that this should not impact on the validity of the results, because Europeans' attitude toward crypto-agility is likely similar to that in every other part of the world.

About 41% of the respondents have more than ten years of experience in cryptographic software development or cryptography. Only one fourth have less than 2 years.

As expected, cryptography experience is strongly correlated with age, with a correlation coefficient equal to 0.755.

23 respondents (about 72%) are active, or somehow active, in quantum-resistant cryptography. All inactive ones (9 answers) are from ICT firms.

In the followings, we will denote with "***QR-active***" the respondents who declared to be "active" or "somewhat active" in QR cryptography, and with "***QR-inactive***" the other respondents.

Table 2 shows in detail the demographics of the respondents, as a function of their involvement in QR software research and development.

People active in QR are mainly between thirty and forty years old. People somehow active are more equally distributed, with a preference of over 40. Inactive people are either quite young, or senior.

With regard to experience in cryptographic software development or cryptography, people active or somewhat active in QR tend to be quite experienced. Inactive people either have very low experience or are very experienced.

Regarding affiliation, QR-active people are evenly distributed between researchers and developers. On the contrary, all inactive people work in a firm. This is due to the fact that the questionnaires sent to emails obtained from scientific and technical articles on crypto-agility were largely sent to researchers also active in QR research. On the other hand, the questionnaires directly sent to the companies were filled in by cybersecurity experts who were not necessarily involved in QR research.

4.2. Software processes

The survey asks what software development processes are used, and allows to answer more than one process. Consequentially, the average number of processes mentioned by each respondent is 1.8.

The considered processes were of five different types: waterfall processes, iterative processes such as RUP [59], agile/lean processes, security-related ones, and no process. We marked as " "Agile" the answers "Scrum/Secure Scrum", "Kanban board", "Other agile/lean process"; we marked as "Secure" the answers "OWASP/CLASP", "Microsoft SDL" and "BSIMM". 11 answers only mention "No process", whereas two other answers mention "No process" together with other processes.

Fig. 1 shows pie charts of the percentages of adoption of the various types of processes among all respondents, and by their involvement in QR research.

Considering the percentages of reported processes over the total number of mentioned ones, about 47% of all respondents use agile methods, whereas 21% do not use any process. Iterative, waterfall and secure processes are used in decreasing percentage, from 13.1 to 11.5 to 8.2 percent, respectively. Nobody declared to use BSIMM model.

One third of university/research respondents answers that they do not use any process, one fourth that they use agile methods. A secure process is used by 16.7% of researchers, whereas iterative and waterfall processes are mentioned in 12.5% of the answers.

Most firm respondents use agile methods (59.5%), the percentages of them declaring to use no process, and to use an iterative one are the same (13.5%). 10.8% use Waterfall, whereas a secure method (OWASP/CLASP) is mentioned only in one answer.

The two bottom pie charts shows the answers of people involved in QR research, and of those who are not involved. Respondents who are QR active, or somehow active, in quantum-resistant cryptography tend to use agile methods less than those who are inactive. However, the overall use of agile methods is quite popular both among QR-active and QR-inactive respondents. A few QR-active respondents use a secure process (11.6%), and about one fourth do not use a process. The remaining use Waterfall and iterative processes (11.5 and 13.1 percent, respectively).

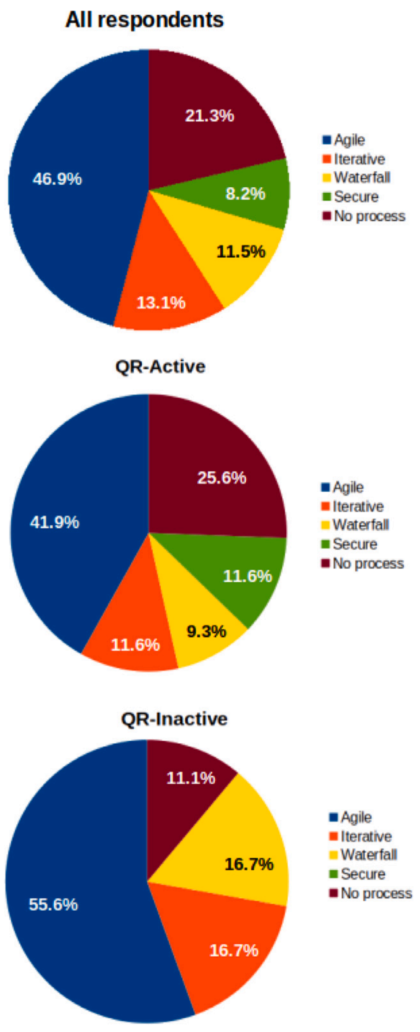

Fig. 1. Adoption percentage of various types of processes by all respondents (top), QR-active respondents (center) and QR-inactive ones (bottom).

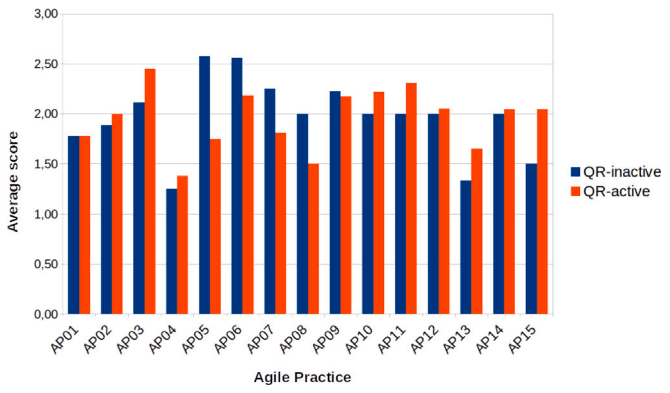

Fig. 2. Average score of the 15 agile practices, given by respondents depending on their involvement in Quantum Resistant cryptography.

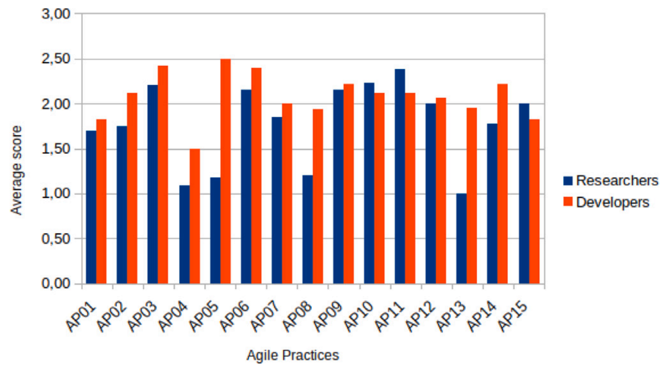

Fig. 3. Average score of the 15 agile practices, given by researchers (blue bars) and industry employees (red bars).

55.6% of the respondents who are inactive in quantum-resistant cryptography (all from firms) use agile methods. 17% of them use waterfall or iterative processes. Only two answers declare “No process”, and nobody uses secure processes.

The aforementioned results are influenced by the fact that the majority of active people work in research, whereas the majority of inactive people are employed in industry, so they somewhat mirror the processes used by researchers versus those used by developers.

4.3. Results on crypto-agility and agile practices

To measure the respondents’ opinion about the effectiveness of agile practices to support crypto-agility, we give a numerical score to the answers in ascending order, namely: 0 (Irrelevant), 1 (No. much), 2 (Quite) and 3 (A lot). We do not give a score to “I don’t know” answers.

Table 3 reports the codes of main Agile and Lean practices as defined in Table 1, with respondents’ judgment on their relevance to crypto-agility. Respondents are divided into QR-active and QR-inactive. You can see that most of the agile practices are deemed to be quite useful, or very useful.

Fig. 2 graphically shows the average scores of APs for QR-active and QR-inactive respondents; Fig. 3 shows the average scores for researchers and developers. Most APs score over 1.5, meaning closer to “Quite” (score = 2) rather than “Not much” (score = 1).

Overall, the most effective APs to support crypto-agility, with an average score above 2 as shown in the last row of Table 3, are AP03 (Continuous Integration), AP06 (Use of an automated test suite), AP11 (Coding standards), AP09 (Simple design and simplicity), AP10 (Regular refactoring), AP12 (Pair programming) and AP14 (Iteration review meetings).

The least appreciated AP is AP04 (Self-organizing team), with an average score of 1.34. Also AP13 (Daily meetings) and AP08 (Collective code ownership) are not deemed very supportive to crypto-agility.

All other APs have a score close to 2, meaning that on average their effectiveness is considered quite good by respondents.

For some APs, there is a big difference between the average answers of respondents who are QR-active and those who are not. The use of boards, automated testing, TDD and collective code ownership are considered more effective by respondents inactive in QR research than by QR-active ones. Retrospectives meetings are considered more important by respondents active in QR research than by QR-inactive ones. All other APs do not exhibit evident differences.

The first result is clearly influenced by the fact that the majority of inactive people are employed in industry, where use of boards, testing and collective code ownership are very popular. On the other hand, the importance of retrospectives meetings stressed by developers active in QR research could be due to the novelty and frequent changes in the QR algorithm field, which require continuous evaluation and updating of the work done.

Wishing to assess whether the answers of two sampled groups (QR-active and QR-inactive respondents) related to these practices are statistically distinguishable, we applied one-tailed Mann–Whitney U-Test [50]. This test is a non-parametric technique that can be used with unequal sample sizes, which is the case of our sample groups, and

Table 4
Number of respondents about how agile practices can help crypto-agility characteristics for each answer option, separately for QR-active and QR-inactive respondents.

QR-active	Automatic selection	Adding new features	Retiring features	Upgrading features
Don't know	1	1	2	1
Irrelevant	1	0	0	1
Not much	6	4	2	1
Quite	8	8	9	7
A lot	7	10	10	13
Average score	1.95	2.27	2.38	2.45
QR-inactive				
Don't know	3	2	2	2
Irrelevant	0	0	0	0
Not much	1	1	1	1
Quite	3	2	1	2
A lot	2	4	5	4
Average score	2.17	2.43	2.57	2.43
Total average	2.00	2.31	2.43	2.45

allows to check whether the null hypothesis that two populations are the same can be rejected. Mann–Whitney U-Test is the non-parametric test most widely used in software engineering studies, as reported in a recent work [60].

We applied the test to all 15 APs, to check the probability that the null hypothesis (there is no difference between QR-active and QR-inactive groups) is rejected. We found that the null hypothesis is rejected with a probability $> 96.5\%$ for AP05 (Use of boards), and with a probability $> 92.2\%$ for AP15 (Retrospective meetings). These differences may be considered as statistically significant.

All other APs cannot reject the null hypothesis with a probability greater than 90%.

The same Mann–Whitney U-Test was also applied to assess whether the answers of researchers and developers show statistically significant differences. In this case, we found that the null hypothesis is rejected with a probability $> 99\%$ for AP05 (Use of boards) and AP13 (Daily meetings), and with a probability $> 96\%$ for AP08 (Collective code ownership), so these differences may be considered statistically significant. For AP14 (Iteration review meetings) the null hypothesis is rejected with a probability $> 90\%$, to the limit of significance. No other AP rejects the null hypothesis with a probability $> 75\%$.

Note that all APs whose scores given by researchers and developers are statistically significant show a higher score given by developers.

This is probably due to the regulated environment of cybersecurity software development, where developers follow strict rules and are supervised and coordinated by project managers. This environment is quite different from the environments aimed to develop information system and Web applications, where APs are mostly applied, including self-organizing teams.

Regarding the last question of our questionnaire "How much crypto-agility characteristics can be helped by agile practices?", Table 4 summarizes the results of our study. Note that most respondents think that APs can help "a lot", or "quite" to achieve all crypto-agility characteristics, while the number of negative answers is very low. 10 answers were "I don't know".

5. Discussion and answers to research questions

In this section we provide a more comprehensive interpretation of the research findings, addressing the implications of the results for the adoption of agile practices in achieving crypto-agility and the challenges of transitioning to quantum-resistant algorithms. We also answer the research questions asked at the beginning of the paper, complementing the discussion with the most relevant comments sent by some of respondents.

Let us first report the least favorable comment on using agile practices to achieve crypto-agility, made by an experienced Eastern European developer working in a company, who does not use any specific development process. This developer stated that "*The development of software that implements cryptographic algorithms is a specific area. This is a strictly licensed activity. The implemented algorithms are subject to requirements in accordance with the security model. It is very difficult and sometimes harmful to implement agile development here. There is a danger of allowing the possibility of attacks*".

This opinion has some arguments in favor, but virtually all other respondents do not share these concerns, according to the answers to the questionnaire. These clearly indicate that, among the 15 APs mentioned, some are considered very suitable to support crypto-agility. The actual average scores were already reported in Table 1.

In decreasing order of importance according to the average scores given by QR-active respondents, the APs deemed most suitable to get crypto-agility are:

5.1. *AP03: Continuous integration performed daily or at regular intervals*

This AP got the highest average score (2.34), quite balanced between QR-active and QR-inactive respondents, with a statistically non-significant preference given by active ones. It is an AP aimed to prevent the issues due to late integration of software components developed separately. It works well when complements with an automatic test suite (AP06), so that possible faults or unwanted side-effects introduced with the code just developed and integrated are found and fixed early. Most respondents consider it to be very helpful to ensure a rapid response to a cryptographic threat, without the issues due to late integration.

5.2. *AP11: Adherence to coding standards and creation of intention-revealing code*

This AP got an average score of 2.23, with a slight preference given by QR-active respondents.

A respondent working in an European university commented on this topic that "*The adherence to software coding standards and coding practices should result in keys or algorithms becoming modular components. ... This software would then allow to change algorithms or re-key a system without having to intervene with business logic and user-intended functionality* ".

This practice should be applied in every software process, agile or not. It is considered "agile" because it is one of the 12 original practices of Extreme Programming.

In the context of crypto-agility, when one needs to upgrade or change existing CA software previously developed software by other teams, working on intention-revealing code will greatly ease the programmer's task. Conversely, writing standard and intention-revealing code will enable future maintainers to work quicker and with a lower fault rate, so crypto-agility will be enhanced.

5.3. *AP10: Regular refactoring to improve code quality*

This AP got a balanced average score of 2.16. Refactoring is the habit to restructure the software to improve its design, without adding new features. It is one of the tools to achieve a better, simpler architecture, and requires an automated test suite in place, to easily perform regression testing after the updates.

Also this AP clearly enhances the ability to change and upgrade software modules developed in this way, thus promoting crypto-agility. However, this comes at a higher cost because consistently and frequently performing refactoring can increase significantly the development effort.

5.4. AP06: Use of an automated test suite

This AP got an average score of 2.29, with QR-inactive respondents giving an average score of 2.56, a difference that is not statistically significant. This AP prescribes developing a set of tests written as code together with the system, which can subsequently be easily run to validate the software after each update (regression testing). As reported before, AP06 is instrumental to AP03 Continuous integration.

A specific comment on automated testing, made by an European developer is: *"Automated tests take the spotlight since well executed tests in simulated environments before deployment can prevent most of the issues arising from the critical changes described by the principles"*.

Testing is very important in every software project, and even more important in the case of software that must meet stringent security criteria. So, the high score got by AP06 was predictable. It is clearly a practice very useful in the field of CA software development, and is instrumental to get crypto-agility.

5.5. AP09: Prioritization of simple design and simplicity

This AP got a balanced average score of 2.19. Simplicity is basically avoiding adding to the software features not strictly required, thinking that they might be useful in the future – an eventuality that usually will not happen. A simpler software is easier to understand, cheaper, and less prone to bugs.

Creating only the strictly required software should be obvious in the regulated environment of CA libraries with the aim of obtaining and enhancing security. This AP has effects similar to those of the previous quoted AP11 practice, so it too looks very important to achieve crypto-agility.

5.6. AP12: "Pair programming (PP)"

This AP was praised in about the same way by both QR-active and inactive respondents (average score 2.05 and 2.00, respectively). Regarding to it, an European firm developer commented that "*Pair programming is more useful than ever since working not alone on critical features such as these can prevent bugs and trivial mistakes*".

PP is a practice that can be very useful when developing critical software in order to have two people knowledgeable on it, or when a new team member is trained on the job to become proficient in managing the software already developed by the team.

Regarding crypto-agility, PP can help produce software with fewer errors and higher quality [61], and thus can help crypto-agility.

5.7. AP15: Retrospectives as meetings to inspect and adapt work process

Retrospective meetings consists of periodically holding a meeting, usually every 1–3 iterations, where the team reflects on what happened in past iterations and identifies actions to improve things in the future. AP15 got a higher score by QR-active respondents (2.05 vs. 1.5), a difference whose Mann–Whitney U-test significance is above 92.2%.

The need to periodically pause the development and reflect on past iterations is valued by QR-active respondents. Retrospective meetings are a chance for all team members to share what went well, what did not, and what could be improved upon for next time. When developing security-critical software, systematically improving the process and managing risks through periodic meetings can be very useful.

5.8. AP14: "Iteration review meetings", AP02: "Incremental development driven by features", AP01: "Time-boxed iterations"

These three APs are strictly related, and implement iterative and incremental development. All of them have similar average scores (2.03, 1.97 and 1.78, respectively), with very small differences between QR-active and inactive respondents.

AP02 prescribes to express requirements as independent features, AP01 prescribes to run the development process using short, time-boxed iterations each implementing a subset of the features, AP14 is about reviewing the work completed after each iteration, accepting the features that pass acceptance tests and moving forward to the next iteration the incomplete features.

An European developer wrote commenting AP02: "*Introducing new cryptographic features could be managed on the broader level with user-stories, that would help define precisely what the feature should do. Also, TDD would help turn into code these requirements and then the development could proceed until the requests are satisfied*".

Using iterative and incremental development with features specifying the formal requirements of the CA software to implement or upgrade, and controlling the development process and the quality of the produced software after each iteration is clearly useful to obtain crypto-agility.

5.9. AP07: Writing tests before writing the code (TDD)

Also this AP has a higher score by QR-inactive respondents with respect to QR-active ones (2.25 versus 1.93). TDD is an AP highly considered among developers who seriously apply agile methods. We got some interesting comments about TDD.

Among them, an Italian developer working in an ICT firm commented "*A great contribution would be given especially by the test-driven programming conventions, that would help building a compact, expandable and solid codebase*".

Another European developer working in an ICT firm commented "*TDD can help in particular to define in advance the common mechanism and behavior of an encryption algorithm to be adopted by the system. If the automated tests are well implemented, they can then again be used when an algorithm is changed to check that the same requirements are still satisfied, and to prevent regression (and/or security issues) in the system*". The same developer added *"If the implementation of a new feature would end up causing side effects, this could be verified in advance through the automatic test suite and help intervene accordingly"*, and "*Retiring old features could be done after all of the previous features' duties are now performed by newly developed code. An automatic test suite would be crucial in this regard, and a Kanban board could help keep track of all the necessary tasks that have to be completed before the obsolete feature's phaseout*".

From these comments, it is not surprising that "AP06 Automated testing" described above had the second highest average score of all the APs considered, and that the "AP07 TDD" also achieved a high average score.

However, while the availability of an automated test suite (AP06) is clearly important to be able to change quickly and effectively CA software, thus improving crypto-agility, TDD (AP07) is a good design practice but is not specific to the development of CA software, and to crypto-agility.

5.10. AP05: Use of boards to visualize the project's status

This AP prescribe the use of *"Information Radiators"*, that is boards, diagrams, cards to visualize the state of the project, to make the whole team aware of it, and to highlight possible problems early.

A comment by an Italian developer about boards was: *"Regarding retiring old features, . . . a Kanban board could help keep track of all the necessary tasks that have to be completed before the obsolete feature's phaseout"*.

QR-inactive respondents highly consider this AP (average score of 2.57), whereas QR-active ones give it an average score of 1.75. The difference is statistically significant, as reported in Section 4.3.

To dig further in the meaning of this difference, we applied the Mann Whitney U test to statistically assess the difference between researchers (academics) and developers (non-academic) respondents. The average score for the 11 researchers and the 16 developers who answered the question are 1.2 and 2.5, respectively, resulting in a probability $p \geq 99,95\%$ that the difference is statistically significant.

Remembering that all QR-inactive respondents are developers, the conclusion is that the difference between QR-active and QR-inactive respondents is due to the fact that researchers seem not to be impressed by this practice whereas developers appreciate it a lot.

In general AP05 is a practice used in software development, much less in scientific research. As every "good" development practice it can help getting crypto-agility, but it is not specifically aimed to it.

5.11. AP13: "Daily meetings for progress assessment and issue resolution"

The average score of this AP is 1.56, with QR-active respondents giving a slightly higher average score than QR-inactive ones (1.65 versus 1.33).

The low score is probably due to the fact that cryptographic software development is typically performed on time scales longer than development of Web applications or other more common types of software, thus not requiring daily assessments of the project status.

5.12. AP08: "Collective code ownership"

This AP has an average score of 1.67; the average score of QR-active respondents is 1.5, that of QR-inactive ones is 2.0. This AP looks quite ad odds with software security development, where every piece of code must have someone responsible for its correctness. Its low score therefore appears well justified, as well as the lower score given by people active in CA development for obtaining quantum resistance.

5.13. AP04: "Team members volunteer for tasks (a self-organizing team approach)"

This AP got the lowest average score of 1.34, balanced between QR-active and inactive respondents (1.38 versus 1.25).

As noted before, this is probably due to the regulated environment of security software, where developers are managed by their project managers, and cannot self-organize themselves.

5.14. Answer to RQ 1

The first research question is: ***RQ1: "Which are the most (and least) suitable agile and lean practices to support crypto-agility?"***.

From the scores and comments of the respondents, in particular considering those of people involved in CA software development to get quantum resistance, among the 15 considered APs the most suitable to support crypto-agility are, in decreasing order of importance:

1. *AP03 Continuous integration performed daily or at regular intervals*
2. *AP11 Adherence to coding standards and creation of intention-revealing code*
3. *AP10 Regular refactoring to improve code quality*
4. *AP06 Use of an automated test suite*
5. *AP09 Prioritization of simple design and simplicity*

Note that these APs are exactly those we assumed to be important specifically considering the challenges and issues posed by the transition to quantum resistance, reported at the end of Section 2.6

On the contrary, the APs considered less suitable to support crypto-agility are:

1. *AP04 Team members volunteer for tasks (a self-organizing team approach)*
2. *AP08 Collective code ownership*
3. *AP13 Daily meetings for progress assessment and issue resolution*

Other APs considered useful, but not as useful as the previous ones are *AP12 Pair Programming*, *AP15 Retrospective meetings*, and the iterative-incremental approach to develop software: *AP14 Iteration review meetings, AP02 Incremental development driven by features, AP01 Time-boxed iterations*.

AP07 Test Driven Development and *AP05 Use of boards to visualize the project's status* are less considered, but not with an average score as low as the three APs listed above.

We believe that the above discussion of the most and least suitable APs for achieving crypto-agility, according to the opinion of the respondents to our questionnaire, has satisfactorily answered our first research question.

5.15. Answer to RQ 2

The second research question is: ***RQ2: "Which crypto-agility characteristic can benefit more from the agile approach?"***.

This RQ can be answered quite easily. From Table 4 it is evident that most respondents answered that all four crypto-agility characteristics, as defined in Section 2.2 can benefit from the agile approach.

The lowest score is given to the first crypto-agility characteristic, "*Ability for machines to select their security algorithms in real-time and based on their combined security functions*". Perhaps this characteristic is considered by a minority of respondents, both in research institutions and in industry, to be too critical to be developed using APs. However, most respondents do not agree and consider APs quite, or a lot relevant to implement this characteristic.

All other three questions about the suitability of APs to help obtaining crypto-agility characteristics got very high scores, with only one answer "Irrelevant" and 10 answers "Not much" over 46 answers "A lot" and 29 answers "Quite".

So, the answer to this RQ2 is that all four crypto-agility characteristics can benefit from the agile approach.

5.16. Answer to RQ 3

The third research question is: ***RQ3: "Are there differences between how agile practices are viewed by people working on quantum resistance algorithms and people just involved in crypto-security?"***.

Recalling the above discussion and referring to Fig. 2, we can state that for seven considered APs there is almost no difference. These APs are AP01, AP02, AP04, AP09, AP10, AP12 and AP14.

Among the remaining APs, four are appreciated more by people who are QR-active, namely AP03, AP11, AP13, AP15. Among these, only AP15 (Retrospective meetings) appears to have a statistically significant difference, with probability greater than 0.922.

Four APs, namely AP05, AP06, AP07 and AP08, look to be appreciated more by people who are QR-inactive. Only AP05 (Use of boards) is statistically significant, with probability greater than 0.965.

The significance tests were again made using Mann–Whitney U-Test [50].

We recall that all QR-inactive respondents are industrial developers, whereas QR-active ones are well distributed between researchers and developers (see Table 2). As reported in Fig. 3, the only significant differences between researchers and developers, all in favor of the developers, concern AP05, AP08 and AP13. So, the advantage of AP05 (Use of boards) and AP08 (Collective code ownership) might be explained by the higher percentage of developers among QR-inactive respondents.

The prevalence of the average score of AP13 (Daily meetings) given by QR-active respondents cannot be explained in the same way. The score difference of AP13 between QR-active and QR-inactive respondents, however, is not statistically significant.

We can summarize the answer to RQ3 by saying that, among all the APs, only AP15 “Retrospective meetings” is clearly preferred by QR-active respondents.

On the other hand, only AP05 “Use of boards” is clearly preferred by QR-inactive respondents.

For all other APs, there is no statistically significant difference between the two groups.

6. Threats to validity

In the following, we discuss the threats to validity to be considered when applying the method presented in the paper at hand.

6.1. Internal validity

Internal validity threats refer possible unknown and hidden factors that could affect the results [62]. In our survey, the factors affecting the internal validity of our work are coverage bias and item errors. The first might be the fact that the professional level distribution of the survey participants may not be balanced.

The second accounts for survey respondents who may misunderstand some questions. To mitigate the latter factor, all authors discussed and refined the survey questions, also keeping to a minimum the number and complexity of the questions, and made some exterimentations before finalizing the questionnaire, as reported in Section 3.2. However, interpretation differences among respondents could still introduce some errors.

6.2. Construct validity

Construct validity is concerned with whether one can justifiably make claims at the conceptual level that are supported by results at the operational level. In our case, the observed results of our survey might not correspond to the effect we think we are measuring. For instance, we defined “agility” as the use of 15 APs, chosen after an analysis of several more APs found in the literature. There is still the possibility that choosing other APs would have produced a different result. A similar issue might be related to the choice of the definition of “crypto-agility”.

6.3. External validity

External validity is the degree to which results of one experiment can be generalized to a larger population. The results of our study are only based on a data sample of 32 respondents, who were chosen through an email sending campaign, and business contacts.. This was an example of convenience sampling. So, the sample size might not be big enough, and the population sample might not be significant to generalize the study findings. However, several relevant software process studies are conducted in the similar sample size range,*e.g.* [63,64]. The data resulting from our study provide valuable insights about the suitability of agile practices to gain crypto-agility. Generalizability to the entire population may be limited, but the findings offer meaningful insights into this recent topic.

A similar observation can be made about the Mann–Whitney Q test performed to assess the significance of the difference between answers given by different populations (QR-active versus QR-inactive, or researchers versus developers). The small dimension of the considered samples does not bears enough statistical power to avoid Type I errors, that happen when the null hypothesis is rejected when it is true [60,65]. Consequently, the significance of the differences found cannot be generalized, but simply conveys useful information to discuss and interpret the results of our survey.

6.4. Reliability

The research results may be influenced by the quality of the questionnaire. This threat was addressed by verifying the questionnaire before publishing it. Regarding statistical validity, the limited size of the sample and the fact that the distributions of obtained scores are not normal, led us to use the one-tailed Mann–Whitney U-Test to test whether the difference between two quantities being compared is of practical consequence. However, as quoted above, the significance levels obtained have low statistical power, and must be considered just as informative, but not reliable.

7. Conclusions

The advent of quantum computing bring about existential risks for present cryptographic algorithms, which we may have to deal with within a decade. Being prepared to quickly and safely migrate security software into using quantum resistant algorithms will not be simple task, due to the wide diffusion of software and firmware relying on cryptographic algorithms.

In this article, we have analyzed the relationships between traditional agility, that proved itself to be very effective in helping to develop traditional software, and crypto-agility, that aims to ease the transition from a cryptographic algorithm to another. After a discussion of crypto-agility and quantum resistance issues, we discussed what are the software development practices that best represent agility, targeting the field of cybersecurity. Then, using a questionnaire answered by people active in cybersecurity and QR, we investigated whether and to what extent these leading agile and lean practices are suited to support crypto-agility. Most APs have been rated useful or very useful, with an emphasis on continuous integration, adherence to coding standards, refactoring, automated and massive testing, and simple design.

Some APs are held in less consideration, especially by people active in QR software development. These are daily meetings, collective code ownership, self organizing teams.

Two APs are viewed very differently by people who are active in QR and by those who are inactive. These are the use of boards to visualize the project’s status, that is preferred by people inactive in QR development, and the conduction of retrospective meetings to inspect and adapt the process, that is preferred by people active in QR development.

Regarding the suitability of agile practices to support the four main features of crypto-agility, we found almost unanimous support to the idea that they can take advantage of using an agile approach. However, further software engineering research is needed to integrate agile practices in the more formal cryptographic software development processes.

Potential areas for future investigation in the rapidly evolving domain of quantum-resistant cryptography are the followings:

- *Software processes to ensure crypto-agility, and software processes specifically aimed to support the transition to QR.*
  This field is still in its infancy. There were proposals for a maturity model for crypto-agility assessment [16], and of a framework to assess the risk that results from the lack of crypto-agility [14]. A recent paper presented a case study of the application of a road-map called “7E” to address encryption-related challenges associated with quantum advantage, in IBM’s Db2 database system [66]. However, we still need to perform substantial research about agile and non-agile processes and practices, tools to compose and integrate CA software effectively, easily and securely, tools for static code analysis and automated testing specifically aimed to verify crypto-agility. Also processes and tools specifically aimed to take advantage of hardware accelerators, including those explicitly addressing PQ algorithms, would be very appreciated [17]. These investigations will need a strong interaction between cryptographers and software engineers, which is quite new and is little explored.

- *Edge Computing and IoT interaction.*
  Nowadays, the Internet is rapidly evolving towards a paradigm where IoT devices record and send data to the net, performing computation and elaboration on these data as closely as possible to the data source. The number of IoT and Edge devices is enormous and rapidly increasing. Cryptographic primitives are often directly used on those devices to digitally sign sent data and to provide proof of correctness, such as hashing the data-frame, in order to get reliable data and correct computation on it. Within this paradigm quantum attacks will be hard to detect and to tackle, due to the enormous size of the surface of attack [47]. Future investigations will certainly help in establishing how crypto-agility can provide strategies and methodologies which can easily scale on large numbers and on different applications domains when dealing with the interaction with IoT at the Edge. This will be especially useful when managing the upgrade of already installed cryptographic features or algorithms without introducing unwanted side-effects or vulnerabilities on Edge computation and IoT devices.
- *Quantum-safe blockchains.*
  Blockchain technology is relatively recent, but its decentralized finance applications already move tens, and even hundreds, of billions of dollars. The property of these assets is guaranteed by cryptography and it would be for sure one of the preferential targets of a massive quantum attack, if and when available. There are already many studies and proposals on the design of post-quantum blockchains [67], but the role of crypto-agility in the development of QR blockchain software and applications is still poorly studied.
  In particular, according to the crypto-agility definition and to what reported in Section 3, the ability to add new cryptographic features or algorithms to existing hardware or software, resulting in new, stronger security features, and the ability to gracefully retire cryptographic systems that have become either vulnerable or obsolete, would be worth of being studied in future works, as well as the properties of updateability and compatibility for such peculiar emerging technology.

This work is part of a broader involvement of our research group in the field of quantum software engineering.

CRediT authorship contribution statement

Lodovica Marchesi: Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Michele Marchesi:** Writing – review & editing, Writing – original draft, Supervision. **Roberto Tonelli:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We acknowledge financial support under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5-Call for tender No. 3277 published on 30 December 2021 by the Italian Ministry of University and Research (MUR) funded by the European Union-NextGenerationEU. Project Code ECS0000038—Project Title eINS Ecosystem of Innovation for Next Generation Sardinia—CUP F53C22000430001-Grant Assignment Decree No. 1056 adopted on 23 June 2022 by the Italian Ministry of University and Research (MUR), and by Puglia Region, POR-POC Puglia 2014/2020-Title II-Chapter II “Aid for integrated programs promoted by SMEs and MEDIUM ENTERPRISES”. Project Code: OH1RWB4 - Project Title: KryptoSafe - CUP B86I22000060007.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.infsof.2024.107604.

Data availability

Data will be made available on request.

References

[1] K. Beck, M. Beedle, A. Van Bennekum, et al., The agile manifesto, 2001.
[2] B.A. LaMacchia, J.L. Manferdelli, New vistas in elliptic curve cryptography, Inf. Secur. Techn. Rep. 11 (4) (2006) 186–192.
[3] D. Lazar, H. Chen, X. Wang, N. Zeldovich, Why does cryptographic software fail? A case study and open problems, in: Proceedings of 5th Asia-Pacific Workshop on Systems, 2014, pp. 1–7.
[4] A. Majot, R. Yampolskiy, Global catastrophic risk and security implications of quantum computers, Futures 72 (2015) 17–26.
[5] M. Mosca, Cybersecurity in an era with quantum computers: Will we be ready? IEEE Secur. Privacy 16 (5) (2018) 38–41.
[6] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: A systematic review, Inf. Softw. Technol. 50 (9–10) (2008) 833–859.
[7] L. Williams, A. Cockburn, Agile software development: It's about feedback and change, Computer 36 (6) (2003) 39–43.
[8] M. Piani, M. Mosca, Quantum Threat Timeline Report (2023), Global Risk Institute, Canada, 2023.
[9] M. Kuhrmann, P. Tell, R. Hebig, J. Klünder, J. Münch, O. Linssen, D. Pfahl, M. Felderer, C.R. Prause, S.G. MacDonell, et al., What makes agile software development agile? IEEE Trans. Softw. Eng. 48 (9) (2022) 3523–3539.
[10] T.M. Fernández-Caramés, From pre-quantum to post-quantum IoT security: A survey on quantum-resistant cryptosystems for the internet of things, IEEE Internet Things J. 7 (7) (2020) 6457–6480.
[11] T. Takagi (Ed.), Proceedings of the 7th International Workshop on Post-Quantum Cryptography, PQCrypto 2016, vol. LNCS 9606, Springer, 2016.
[12] O. Grote, A. Ahrens, C. Benavente-Peces, Paradigm of post-quantum cryptography and crypto-agility: Strategy approach of quantum-safe techniques, in: 9th International Conference on Pervasive and Embedded Computing and Communication Systems, PECCS 2019, SCITEPRESS Lda, 2019, pp. 91–98.
[13] A. Mashatan, D. Heintzman, The complex path to quantum resistance: Is your organization prepared? Queue 19 (2) (2021) 65–92.
[14] C. Ma, L. Colon, J. Dera, B. Rashidi, V. Garg, CARAF: Crypto Agility Risk Assessment Framework, J. Cybersecur. 7 (1) (2021).
[15] L. Zhang, A. Miranskyy, W. Rjaibi, Quantum advantage and the Y2K bug: A comparison, IEEE Softw. 38 (2) (2021) 80–87.
[16] J. Hohm, A. Heinemann, A. Wiesmaier, Towards a maturity model for crypto-agility assessment, 2022, arXiv preprint arXiv:2202.07645.
[17] N. Alnahawi, N. Schmitt, A. Wiesmaier, A. Heinemann, T. Grasmeyer, On the state of crypto-agility, Cryptol. ePrint Arch. (2023).
[18] L. Marchesi, M. Marchesi, R. Tonelli, Reviewing crypto-agility and quantum resistance in the light of agile practices, in: International Conference on Agile Software Development - Workshops, Springer, 2022, pp. 213–221.
[19] A.F. Johnson, L.I. Millet, Cryptographic agility and interoperability, in: Proceedings of Workshop, Washington DC, National Academies Press, 2017, http://dx.doi.org/10.17226/24636.
[20] H.A. Mehrez, O. El Omri, The crypto-agility properties, in: Proc. of the 12th International Multi-Conference on Society, Cybernetics and Informatics, IMSCI, 2018, pp. 99–103.
[21] A.J. Menezes, P.C. Van Oorschot, S.A. Vanstone, Handbook of applied cryptography, CRC Press, 2018.
[22] S. Zhai, Y. Yang, J. Li, C. Qiu, J. Zhao, Research on the application of cryptography on the blockchain, in: J. Phys. Conf. Series, 1168, IOP Publishing, 2019, 032077.
[23] NIST, Cryptographic standards and guidelines, 2024, https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines. (Accessed: 30 August 2024).
[24] R. Canetti, Security and composition of cryptographic protocols: a tutorial (part I), ACM SIGACT News 37 (3) (2006) 67–92.
[25] M.T. Sletholt, J. Hannay, D. Pfahl, H.C. Benestad, H.P. Langtangen, A literature review of agile practices and their effects in scientific software development, in: Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering, 2011, pp. 1–9.
[26] S.A. Licorish, J. Holvitie, S. Hyrynsalmi, V. Leppänen, R.O. Spínola, T.S. Mendes, S.G. MacDonell, J. Buchan, Adoption and suitability of software development methods and practices, in: 2016 23rd Asia-Pacific Software Engineering Conference, APSEC, IEEE, 2016, pp. 369–372.
[27] A. Henriksen, S.A.R. Pedersen, A qualitative case study on agile practices and project success in agile software projects, J. Modern Project Manag. 5 (2022).

[28] R. Vallon, B.J. da Silva Estácio, R. Prikladnicki, T. Grechenig, Systematic literature review on agile practices in global software development, Inf. Softw. Technol. 96 (2018) 161–180.

[29] R. Sandstø, C. Reme-Ness, Agile practices and impacts on project success, J. Eng. Project Prod. Manag. 11 (2021) 255–262.

[30] D. Ghimire, S. Charters, The impact of agile development practices on project outcomes, Software 1 (2022) 265–275.

[31] K. Beznosov, Extreme security engineering: On employing XP practices to achieve'good enough security' without defining it, in: First ACM Workshop on Business Driven Security Engineering, Vol. 31, BizSec, Fairfax, VA, 2003.

[32] B. Fitzgerald, K.-J. Stol, R. O'Sullivan, D. O'Brien, Scaling agile methods to regulated environments: An industry case study, in: 2013 35th International Conference on Software Engineering, ICSE, IEEE, 2013, pp. 863–872.

[33] I. Ghani, Z. Azham, S.R. Jeong, Integrating software security into agile-scrum method, Trans. Internet Inf. Syst. 8 (2) (2014) 646–663.

[34] L.b. Othmane, P. Angin, H. Weffers, B. Bhargava, Extending the agile development process to develop acceptably secure software, IEEE Trans. Depend. Secure Comput. 11 (6) (2014) 497–509.

[35] R. Esteves Maria, L.A. Rodrigues Jr., N.A. Pinto, ScrumS: a model for safe agile development, in: Proceedings of the 7th International Conference on Management of Computational and Collective IntElligence in Digital EcoSystems, 2015, pp. 43–47.

[36] P. Maier, Z. Ma, R. Bloem, Towards a secure scrum process for agile web application development, in: Proceedings of the 12th International Conference on Availability, Reliability and Security, 2017, pp. 1–8.

[37] J. de Vicente Mohino, J. Bermejo Higuera, J.R. Bermejo Higuera, J.A. Sicilia Montalvo, The application of a new secure software development life cycle (S-SDLC) with agile methodologies, Electronics 8 (2019) 1218.

[38] V. Kongsli, Towards agile security in web applications, in: Companion To the 21st ACM SIGPLAN Symposium on Object-Oriented Programming Systems, Languages, and Applications, 2006, pp. 805–808.

[39] L. Williams, A. Meneely, G. Shipley, Protection poker: The new software security" game", IEEE Secur. Privacy 8 (2010) 14–20.

[40] K. Rindell, J. Ruohonen, J. Holvitie, S. Hyrynsalmi, V. Leppänen, Security in agile software development: A practitioner survey, Inf. Softw. Technol. 131 (2021).

[41] F. Moyón, P. Almeida, D. Riofrío, D. Mendez, M. Kalinowski, Security compliance in agile software development: a systematic mapping study, in: 2020 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA, IEEE, 2020, pp. 413–420.

[42] S.S. Gill, A. Kumar, H. Singh, M. Singh, K. Kaur, M. Usman, R. Buyya, Quantum computing: A taxonomy, systematic review and future directions, Softw. - Pract. Exp. 52 (2022) 66–114.

[43] P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Rev. 41 (2) (1999) 303–332.

[44] W. Stallings, Cryptography and network security, 7/E, Pearson Education, Harlow, UK, 2017.

[45] L.K. Grover, Quantum mechanics helps in searching for a needle in a haystack, Phys. Rev. Lett. 79 (1997) 325–328.

[46] NIST, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, Tech. Rep FIPS PUB 202, National Institute of Standards and Technology, 2015.

[47] N. Alnahawi, N. Schmitt, A. Wiesmaier, C.-M. Zok, Toward next generation quantum-safe eids and emrtds: A survey, ACM Trans. Embedded Comput. Syst. 23 (2) (2024) 1–28.

[48] E. Dubrova, K. Ngo, J. Gärtner, R. Wang, Breaking a fifth-order masked implementation of crystals-kyber by copy-paste, in: Proceedings of the 10th ACM Asia Public-Key Cryptography Workshop, 2023, pp. 10–20.

[49] W. Beullens, Breaking rainbow takes a weekend on a laptop, in: Annual International Cryptology Conference, Springer, 2022, pp. 464–479.

[50] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering, Springer Science & Business Media, 2012.

[51] Legion of the Bouncy Castle, Bouncy castle – open-source cryptographic APIs, 2024, https://www.bouncycastle.org/. (Accessed: 30 August 2024).

[52] Linux Foundation's Post-Quantum Cryptography Alliance, Open quantum safe (OQS) project web site, 2024, https://openquantumsafe.org/. (Accessed: 30 August 2024).

[53] Botan, Botan: Crypto and TLS for modern C++, 2024, https://botan.randombit.net/. (Accessed: 30 August 2024).

[54] wolfSSL Inc., wolfSSL software development process and quality assurance, 2024, https://www.wolfssl.com/about/wolfssl-software-development-process-quality-assurance/. (Accessed: 30 August 2024).

[55] Mozilla Foundation, Network security services, 2024, https://nss-crypto.org/. (Accessed: 30 August 2024).

[56] A.N. Ghazi, K. Petersen, S.S.V.R. Reddy, H. Nekkanti, Survey research in software engineering: Problems and mitigation strategies, IEEE Access 7 (2018) 24703–24718.

[57] R.M. De Mello, G.H. Travassos, Surveys in software engineering: Identifying representative samples, in: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2016, pp. 1–6.

[58] M.-J. Wu, K. Zhao, F. Fils-Aime, Response rates of online surveys in published research: A meta-analysis, Comput. Human Behav. Rep. 7 (2022) 100206, http://dx.doi.org/10.1016/j.chbr.2022.100206.

[59] I. Jacobson, G. Booch, J. Rumbaugh, The unified process, Ieee Softw. 16 (3) (1999) 96.

[60] F.G. de Oliveira Neto, R. Torkar, R. Feldt, L. Gren, C.A. Furia, Z. Huang, Evolution of statistical analysis in empirical software engineering research: Current state and steps forward, J. Syst. Softw. 156 (2019) 246–267.

[61] A. Begel, N. Nagappan, Pair programming: what's in it for me? in: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 2008, pp. 120–128.

[62] T.D. Cook, D.T. Campbell, W. Shadish, Experimental and quasi-experimental designs for generalized causal inference, vol. 1195, Houghton Mifflin Boston, MA, 2002.

[63] M. Felderer, D. Winkler, S. Biffl, Hybrid software and system development in practice: initial results from Austria, in: Product-Focused Software Process Improvement: 18th International Conference, PROFES 2017, Innsbruck, Austria, November 29–December 1, 2017, Proceedings 18, Springer, 2017, pp. 435–442.

[64] P. Zhou, A.A. Ali Khan, P. Liang, S. Badshah, System and software processes in practice: Insights from chinese industry, in: Evaluation and Assessment in Software Engineering, 2021, pp. 394–401.

[65] T. Dybå, V.B. Kampenes, D.I. Sjøberg, A systematic review of statistical power in software engineering experiments, Inf. Softw. Technol. 48 (8) (2006) 745–755.

[66] L. Zhang, A. Miranskyy, W. Rjaibi, G. Stager, M. Gray, J. Peck, Making existing software quantum safe: A case study on IBM Db2, Inf. Softw. Technol. 161 (2023) 107249.

[67] Z. Yang, H. Alfauri, B. Farkiani, R. Jain, R. Di Pietro, A. Erbad, A survey and comparison of post-quantum and quantum blockchains, IEEE Commun. Surv. Tutor. (2023).