Technology assurance

The NCSC's Technology Assurance activities provide a means to gain confidence in the cyber security of the services and technologies on which the UK relies.

Principles: Product development

7 principles which underpin the development of secure products.

Like all others, quantum security products must follow these principles, as their goal – security – is the same. Users should not be expected or required to have specific quantum knowledge, nor to understand the quantum operation of the device.

Furthermore, it is extremely likely that quantum communications will always be integrated with conventional communications (even within a product or system), so these principles should apply to both aspects, separately but also when together in a system. Quantum-specific comments presented here pertain to the quantum hardware/layer, but with the understanding that the principles will also apply to all conventional hardware/layers, including e.g. key management.

In publishing these principles – on Product development (this document), Product design and functionality, and Through life – so they can be implemented, it is assumed that suppliers of all security products (non-quantum and quantum), whether UK-based or not, will have to provide whatever is needed to evidence that the principles have been followed. Therefore, for all the application and deployment scenarios where historically GCHQ/NCSC would have overseen, or provided, the direct assessment and assurance for their use in the UK, in the future developers of all security products and services will need to provide arguments that they meet assurance claims that underpin the principles, backed by evidence.

Note: This provision of evidence would seem to be a new challenge for companies (particularly non-UK) wishing to supply security products to UK markets or wishing to participate in the supply chains (see later) of

companies producing security products for the UK. It will be interesting to monitor the response of these companies to this new approach from NCSC (effectively now defining principles and outsourcing the assurance, rather than undertaking or directly overseeing it).

Good security engineering means building technologies that remain usable and resilient throughout their lifetime, even in the face of cyber attack.

A high-level requirement of resilience suggests that quantum products should have a default failure mode (e.g. when the quantum channel is lost). For example, with key distribution/management products, the failure mode should be to default to use of PQC (which is likely to be a part of such products for authentication) to maintain key services. Therefore, the assurance principles must apply to this default mode as for any other non-quantum products.

These principles describe the overarching security outcomes that the NCSC would look for when assessing product development processes and practices.

Determining the degree to which a product developer meets these principles gives us a measure of how competent they are at building secure technologies. To aid this, accompanying each principle is a non-exhaustive list of the type of defensive measures which a product developer can provide evidence of, in order to demonstrate their competence.

These two points apply to quantum product developers as they do to non-quantum, and with reference to the conventional (non-quantum) aspects as well as the quantum ones.

It is worth noting that the “providers” of the quantum channels (fibre or free space) used by the quantum products when in operation do not have to provide similar assurance. The whole point about quantum communications is that the technology detects if the channel is compromised or broken. (Service can be denied by an attacker breaking the channel, but undetectable eavesdropping cannot occur.) In a sense, a successful secure operation of quantum communications provides the

assurance of the quantum channel, for operation with that product. More precisely, the quantum protocol used by a product will enable the information gained by any eavesdropper to be estimated. If this is below a protocol-determined limit, then "successful secure operation" of quantum communications can be quantifiably achieved.

Demonstrating competence

As a competent product developer, you should be able to demonstrate that you have processes and practices in place to incorporate security and usability into your whole development process and resultant products.

You should also be able to demonstrate you have secured your development infrastructure against unauthorised access, data transfer and data modification (from both internal and external sources), so as to maintain the confidentiality and integrity of your product development artefacts, including requirements and design documentation, source code and test plans.

Minimising the likelihood and impact of compromise

The security of a finished product can be compromised when a vulnerability - which may have been introduced during development - is exploited.

Product development itself can also be disrupted by a security compromise. For example, by a ransomware infection enabled by a phishing attack on one of the development team.

These compromises can result from either malicious activity or through a simple mistake.

Ensuring that a product vendor's engineering processes and practices minimise both the potential likelihood and possible impact of a security compromise plays an essential part in gaining assurance in both their overall competence and trust in the products that they produce.

Technology Assurance Principles - Developing a secure product

Quantum security products will include conventional IT (for the user interface, control, operation, post-processing, etc) and this must be assured and resilient to cyber attack in the same manner as non-quantum security products. The point here is to consider the additional impact of these assurance principles for the quantum aspects of the products.

Quantum security products have the potential to be attacked through quantum side channels[1]. A conventional cyber attack on the conventional IT parts of a quantum security product is to be regarded as a conventional side channel and should be addressed through the assurance processes as applied to all security products. Therefore, this specifically quantum discussion considers only the quantum side channels, that arise in the quantum layer.

[1] For more details and terminology, refer to the "Introduction_Quantum Assurance" document.

Developing and building products which are resilient to cyber attack can be broken down into seven areas of concern. These are:

- **1. Design for user need**
- **2. Enable your developers**
- **3. Manage your supply chain risk**
- **4. Secure your development environment**
- **5. Review and test frequently**
- **6. Manage change effectively**
- **7. Build for through-life**

1. Design for user need

Appreciating how your product will be used and maintained throughout its life, is a crucial aspect of ensuring that security will endure in practice.

You must frequently and consistently capture and record requirements (including security requirements) for all of your product's intended operational uses, and the different people that will be involved in installing, using, and maintaining it. This will help to ensure that your resultant product's functionality meets the user's needs whilst also being both usable and secure.

Understanding evolving user requirements and operational uses will require a security expert, who does not necessarily need quantum expertise. Their job is to use this information to create specifications that quantum-expert designers and constructors can work to. Installers and users should not need quantum expertise, whereas hardware and software/firmware maintenance may require (assured) quantum expertise. Modifications to functionality will likely require quantum expertise to implement and thus at minimum sign-off by supplier quantum experts, in order to maintain security assurance.

Examples of Defensive Measures

- A process for discovering requirements on a continuous basis should be in place. Requirements should be validated and documented in a consistent and auditable way. This helps to ensure that the goals of the product are visible, remain current, and that changes are easily identified and tracked.
- Requirements should be sought from a sufficiently representative group of stakeholders. Without this understanding, products are unlikely to meet the intended users' needs and this can have a direct security impact (for example, if usability is not adequately considered a device may be difficult to configure, leading to a prospective customer purchasing a less secure but easier to use product instead).
    - Quantum expertise should not be required to configure, use and (as required) reconfigure products. Or if it is, this expertise needs to be from an assured expert or the product producer/supplier. Otherwise, end users might inadvertently introduce new quantum side channels.
    - In the case of QKD, expert members of ETSI, ISO/IEC, ITU-T and CEN/CENELEC technical committees on QKD come from vendors, service providers (who also act as proxy users), academia, and national and international laboratories, and can provide expert advice on requirements. The BSI “ICT/4, Quantum Technologies” technical committee (previously “ICT/1/1/2, Quantum technologies” panel) aims to co-ordinate

UK activity on quantum technologies and could be a forum for gathering UK-centred advice on QKD security requirements.

- Requirements need to be usable for the developers that implement them. That is, they should be understandable and easy to access, their purpose and reasoning should be clearly articulated. They should be readily and easily absorbed into the product's existing development and implementation processes from the outset.
- Security requirements should be derived from an understanding of the potential threats to a product.
    - Quantum security products have the potential to be attacked through quantum side channels. Responsibility for identifying currently known side channels and understanding and addressing these implementation security matters must rest with the product producer/supplier. Therefore, the producer/supplier needs to be up-to-date on known quantum side-channel attacks and address those relevant to their product (through modifications to the security proof, or counter-measures, or both). Exploration to invent or discover new quantum side channels can be left to academia and laboratories specialising in crypto-attacks. Otherwise, this exploration could be an unaffordable cost to the producer.
- All requirements should be used to inform and evolve the product's verification regime.

2. Enable your developers

Developers are not always security experts, so it is essential they have access to appropriate training, expertise and useable, up-to-date development tools, such as compilers, static code analysers and simulators.

An organisational culture that promotes and values developer contributions to security is another crucial factor in enabling 'secure by design' outcomes.

This suggests that a development team for quantum security products must have both quantum (hardware and protocol) and security expertise.

The defensive measures discussed below seem to be framed from a software perspective. However, conventional security products contain (i) crypto/algorithms, (ii) software and (iii) hardware. The security principles must apply to all three. (E.g. secure crypto could be implemented correctly in software but the hardware that runs this could radiate information and thus the overall product is insecure.) Therefore, the defensive measures must cover all aspects of the product.

The same must apply to quantum (or part-quantum) security products. Developers will need to identify and specify any discrepancies between the developed product and the assumptions in its security proof. They should maintain an up-to-date catalogue of attacks, countermeasures and security proofs for the relevant protocol. Particular attention needs to be paid to the potential for introducing side channels in the hardware and its control software. This will enable side channels to be identified and analysed, so countermeasures can be considered and devised.

Examples of Defensive Measures

- Developers should only use supported tools and tool extensions. If this is not possible, the reasons why not should be understood and, where applicable, mitigations put in place to minimise any associated risks. Keeping tools up-to-date ensures that the developer has access to the latest functionality and also helps reduce the risk of compromise.
- Tool choice should be based on a balanced consideration of usability and functionality. This increases the likelihood of successful tool adoption whilst decreasing the possibility of incorrect usage. Ideally, the configuration settings employed by the developer should be the most secure possible. This reduces the risk of residual product defects. Both contribute towards timely product delivery.
- An appropriate, up-to-date coding standard should be followed for every implementation language used. This will ensure that all of the code has a consistent 'look and feel', which can aid readability and future maintainability. More importantly, coding standards also

constrain the features of the language a developer can use to just that of a limited subset. This could be for reasons such as performance or perhaps to avoid potentially ambiguous syntax or common programming errors.

- Developers are often not security or usability experts, so training and access to the necessary skills and security expertise should be provided to them on an ongoing basis.
- Leadership should invest in the establishment of a positive security culture. There are often many competing requirements and priorities during product development - security should be given sufficient credibility and importance by seniors, so that it is considered and included from the outset.

3. Manage your supply chain risk

It is almost inevitable that third-party goods (including open source components) and services will be incorporated into your product during its development. Whilst providing many benefits (such as cost and time savings), such inclusions can potentially introduce additional risks that need to be identified and managed.

Supply chain security should be in place to ensure third-party components are not compromised before they are incorporated into the build.

> These supply chain considerations clearly apply to all security products, quantum included. In the quantum case there will be additional components/devices (e.g. sources, detectors, optical and electronic components) that are particular to the quantum aspect. All these need to be identified and considered, with regard to all of the supply chain defensive measures listed below. At present, it is very likely that a supply chain for quantum security hardware will include non-UK suppliers. Therefore, focus will need to be on non-UK quantum-related suppliers.
>
> This important issue of "technology sovereignty" applies across the hi-tech and cyber sectors[1], and so it is not a unique challenge for quantum tech. Nevertheless, quantum supply chains requiring non-

UK suppliers will need to consider from where within the three global heavyweights (the EU, the USA and China), or elsewhere, any non-UK quantum components are to be sourced[2].

[1] https://www.theregister.com/2022/09/29/arm_founder_uk_tech_sovereignty/

[2] There are some quantum positives. For example, an entangled source can be assured by measurement of its violation of a suitable Bell inequality, independent of its origin or supplier.

Examples of Defensive Measures

- Adhere to the NCSC's Supply Chain Security Principles.
- Maintain an accurate inventory of all third-party goods, services and suppliers being used.
- Identify who is responsible for the security of third-party supplied goods and services and ensure the controls in place are proportionate to the level of risk.
- Establish how long third-party supplied goods (such as components or development tools) will be supported for, and whether this aligns with your product's intended lifespan. Where there is misalignment, put appropriate mitigations in place (such as using an alternative).
- Determine what the normal update cycle is (daily, weekly, etc.) for third-party supplied goods (such as components or development tools) and how these updates will be advertised and obtained.
- Consider how often a third-party supplier checks for publicly known security vulnerabilities in their products and what actions they take if any are discovered.
- Determine if the level of testing that has been performed on a third-party component by its original supplier is appropriate for its intended usage. If it is not, apply suitable mitigations (such as performing additional testing or using an alternative).
- Apply updates as quickly as possible to original vendor-supplied, third-party components and development tools. This will help to reduce the likelihood of vulnerabilities.

4. Secure your development environment

Protecting your development environment from cyber attack enables you to maintain the confidentiality and integrity of your product data.

Having a regularly tested disaster recovery plan will enable you to get things back to normal as quickly as possible, if something does go wrong.

Again, this principle emphasises the cyber security (as opposed to the physical access security) of the development environment. These considerations and defensive measures all map to the case of quantum security products. However, given the physical aspects of quantum security products, this principle of a secure development environment should also be considered from the physical and access perspectives, not just cyber. Physical intruders could interfere with (or gain information about) quantum hardware, even if they have no cyber or electronic access to anything. Therefore, physical security of the environment should be added to the defensive measures.

Examples of Defensive Measures

- An IT asset management system should be used to maintain an accurate inventory of all the hardware and software used within your development environment. Records should also be kept of where all sensitive data is being stored, to ensure it is adequately protected.
- Systems storing sensitive data should be patched and updated to the latest version as soon as possible. This helps protect them against the latest known vulnerabilities.
- An access control system should be in place to ensure that only authorised users can access sensitive data. A suitably robust logging and auditing regime should also be adhered to. This will help detect unauthorised or unusual accesses, or data transfers.
- Core business services (such as email and document management) should be either logically or physically separate from development environments. You should determine the degree of separation on a case-by-case basis. This will help to ensure that a successful attack

on one system does not necessarily lead to the compromise of another.

- Sensitive data, credentials and secret keys used to access and trigger the build should be protected and handled securely. This ensures that only authorised users have access to the build pipeline.
- The development team should be aware of the impact of their digital footprint and what to do if they suspect they have received a phishing email or text message. This helps to reduce the likelihood that product data can be compromised through a social engineering attack.
- A disaster recovery plan should be in place and adhered to. This should include the requirement that critical data be regularly backed up to a separate location and the process of restoring from it be frequently tested to ensure it works.
    - This back-up should be via a quantum secure link.

5. Review and test frequently

You should have a rigorous verification regime in place, which uses multiple approaches, such as testing and peer review.

For the specifically quantum aspects of any security product, there is a role for an independent, quantum-expert, institution here. It should at least peer review the hardware tests that are used; and could consult/collaborate on the establishment of the tests. Companies cannot peer review each other and using academics, paid as consultants, is also not a good approach.

This helps you find defects and ensures that security works in practice. It will also tell you whether the product is achieving its end goals. The verification process should not get in the way of delivery and should be automated where possible.

ETSI has recently released a Protection Profile, which specifies high-level requirements for the physical implementation of prepare-and-measure QKD systems. Although written to help manufacturers submit pairs of QKD

modules for evaluation under the Common Criteria security certification scheme, it can also be a useful resource for evaluations under the PBA scheme.

Examples of Defensive Measures

- A comprehensive verification regime should be in place that will be used to both locate defects and determine whether each of the product requirements has been met. This should involve both Static Verification (e.g. personal and peer review, static code analysis and formal verification) and Dynamic Verification (all forms of testing).

  - In QKD, vulnerabilities can arise not only from product defects (incorrect implementation and manufacture of hardware and software), but because the physical hardware does not possess the properties assumed in the security proof. This introduces quantum side channels[1] which can be exploited by an attacker. Countermeasures to quantum side channels would be: (i) to accommodate the true properties into the security proof; (ii) to implement additional protective measures which can be accommodated into a security proof; (iii) where (i) and (ii) are not currently achievable, implement protective measures which can bound the potential for information leakage to an attacker. Refer to ETSI White Paper for an overview. For recent discussion of "A security framework for quantum key distribution implementations", refer to https://arxiv.org/abs/2305.05930.

  [1] For more details and terminology, refer to the "Introduction_Quantum Assurance" document.

- Verification should be made as easy as possible through the creation of consistent, well-structured and understandable product development artefacts. This should include design documents, models and source code. Adherence to simple and unambiguous procedures should help the process to run smoothly.

  - Verification applies to security proofs in the quantum case. For hardware, verification involves physical test procedures, i.e. measurement of security-critical physical properties.

- The location, independence, coverage, frequency and repeatability of verification activities should be optimised. Automation can greatly support this and also aid efficiency.
- Verification should be a continuous process that is performed during both product development and post release (to ensure a product remains secure throughout its life). The specifics of its constituent activities, as well as their coverage, should be frequently reviewed and, if appropriate, updated. For example, the contents of test suites, or what is looked for during peer review should be kept current and regularly exercised (test runs, reviews performed, etc.) so that they protect against the latest threats.
    - Hardware needs to be designed to permit testing post-release – either at discrete times during its lifetime, or by real-time testing within the hardware security perimeter during operation – without compromising performance subsequent to testing. Next-generation device-independent products may provide lifetime testing as part of normal operation.
- When defects are discovered, they should be promptly addressed. Root causes should be analysed to determine if they are endemic and ensure that similar mistakes are avoided in the future.
    - When quantum side channels are discovered, they should be promptly addressed and published. Root causes should be analysed to determine if they are endemic and ensure that similar side channels are avoided in the future.

6. Manage change effectively

Inevitably, throughout the development process, there will be all kinds of change. From changing requirements, through to product, technology and threat evolution.

Having processes and practices in place that allow agility but ensure coherence and consistency aids cyber resilience. That is, it enables a response to new information about threat, a potential attack or an

implementation change that will reduce the harm to the system into which your product is deployed.

This concept of change needs to be extended to include the physical quantum aspects of any security product and systems in which these might be deployed. Thus, the emergence of a new quantum side-channel threat can be handled. All this also suggests that the approach of "composable security" [2] (already viewed by quantum security folk as generally desirable) is a good tenet for quantum security. Then change to, or reconfiguration of, part of a quantum security product will not require a whole new security proof. The defensive measures below should extend to the quantum hardware.

[2] For more details and references, refer to the "Introduction_Quantum Assurance" document.

Examples of Defensive Measures

- At the start of product development, identify all classes / types of item that will be subject to configuration management. These should include (but should not be restricted to) anything required to recreate and maintain a specific product version, post release (including the original development and build tools).
- Configuration management, that tracks changes, implements version control, and enables reproducibility should be applied throughout the lifetime of a product and not just during its initial development. Post release support of specific product versions is not possible otherwise.
- Configuration items should be version controlled, with full details of any modification (including the author and time / date) recorded. This will allow root cause analysis to be performed when defects are discovered.
- The product build process should be repeatable and, where possible, automated. Should a defect be discovered in a previous product release, a reproducible build will allow you to revisit the exact build scenario in order to develop a fix. Automation can also reduce the risks of errors and aid efficiency.

- Should a quantum side channel be discovered in or since a previous product release, a countermeasure should be implemented for the next product release.

7. Build for through-life

Given the increasing interconnectivity of the technology we rely upon day-to-day, and the continuously evolving cyber security threat landscape, it is essential to maintain and support products throughout their lifetime, if they are to remain secure.

This general principle clearly applies to all security products, whether quantum or not.

As stated earlier, there should be no reliance on users or customers having any quantum knowledge. Therefore, any upgrades, reconfigurations, fixes, etc., that require such knowledge have to rest with and be undertaken by the product producer/supplier. Quantum security should default to non-quantum security whilst this is undertaken.

Software aspects of quantum security products should be addressable through all the same approaches as used for non-quantum products (such as the defensive measures below), but hardware matters will additionally require engineers with the appropriate skills (which is also usually the case for non-quantum hardware). These engineers could be provided by the producer/supplier, or be suitably assured/certified third-party engineers.

Examples of Defensive Measures

- Product development artefacts, such as design documents, source code and test scripts, should be clear and consistent. This will ensure they are easily understood and maintained in the future.
- Implement a suitable mechanism by which externally discovered product defects (including vulnerabilities) can be reported easily and responsibly. Security related defects should be acted upon as quickly as possible.

- Maintain up-to-date knowledge of publicly known security vulnerabilities and, if necessary, augment a product's verification activities to protect against them. Users of at-risk products should be promptly notified and, where possible, a fix deployed as soon as reasonably practicable.

    - Hardware can suffer from degradation due to ageing or the environment. Lifetime testing should be implemented to reveal vulnerabilities introduced by this process.

- Make it easy for a customer to determine which particular version of a product they are using.
- Publish a product support lifecycle that explicitly states the minimum length of time for which a product will receive security updates and the reasoning behind the stated duration. Customers also need to be provided with suitable notice of when product support will cease, so there is time for them to develop contingency plans, which could include sourcing a suitable alternative, accepting and managing the risk of using an obsolete product.
- Products should be actively supported, with safe and easy to implement updates. These should be released in a timely manner and advertised to supported customers via a suitable mechanism. The reasons for the update should be clearly communicated.
- Provide customers with clear guidance on how to securely configure, use and update a product.
- Employ the concept of ‘Secure by Default,’ so that a product’s default configuration settings are the most secure possible.