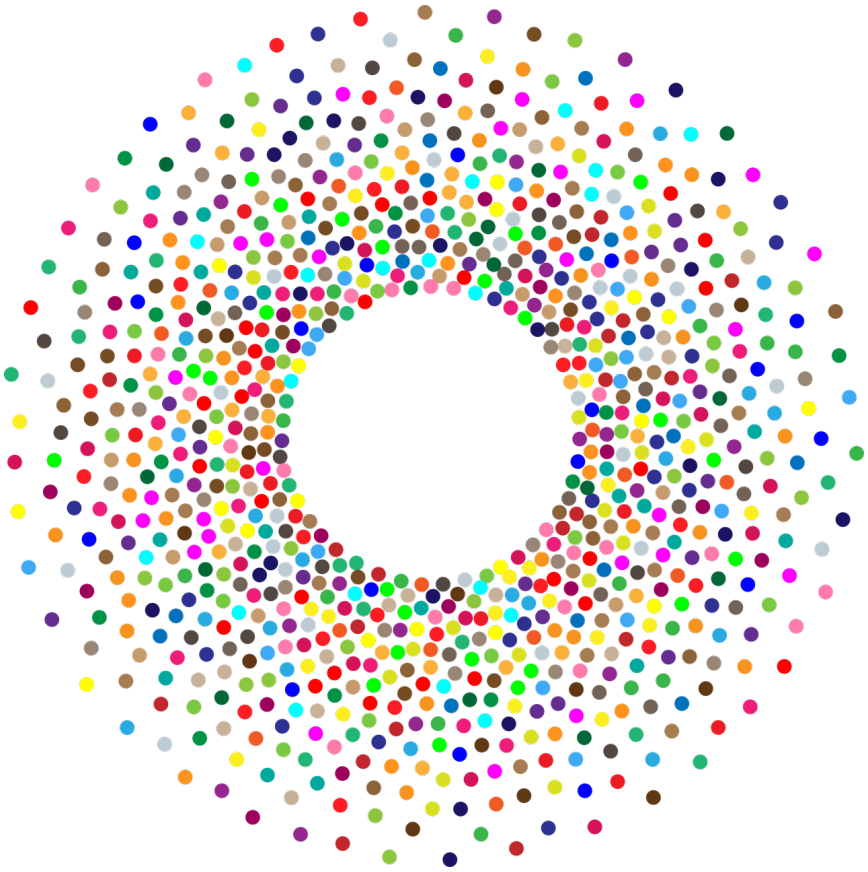
National Cyber Security Centre
a part of GCHQ

Protocol Design Principles



A white paper from the NCSC

Summary

Secure protocols that work for users

We think that user needs should be at the heart of protocol design – this means meeting their use cases and ensuring that solutions work for all stakeholders. These three principles – prioritise the use case, keep it simple, think about the bigger picture – will help to ensure that protocol designs provide the security users need and address the challenges of an evolving internet.

Prioritise the use case. First consider the context of the protocol, where it will run and what threats it will face. Within that context, make design decisions that best improve users' security for the use case. The diversity of users' needs, devices and environment, needs to be considered alongside support for multiple stakeholders. Levels of trust vary, the need for availability can be particularly important, and the threat surface differs by use case. Think about what you reveal and think about what you trust. Consider detection of misuse and consider resilience to both attack and failure. Enable defence-in-depth.

Keep it simple. Designing for simplicity leaves less space for implementation error and user error, reducing the opportunities for compromise. Making it easy for users to make secure choices, and to stay secure, has a huge impact. Avoid design complexity and make it easy to use the protocol securely. Don't reinvent the wheel, and design for easy maintenance – which will vary, dependent on the use case prioritised.

Think about the bigger picture. The security of a protocol is more than just a mathematical analysis; the system it operates in, and the impact it has on the wider environment are at least as important. Encourage diversity and think about secure implementation. Allow the use of least privilege and plan for failure of third parties. Support evolution with your protocols – and help to build a secure internet ecosystem for all users.

Using the three principles

Who are these principles for?

These principles are primarily aimed at designers of protocols, as a guide for use in the design process. They can also be used when implementing and deploying systems, by those who are deciding whether to use an existing protocol. Generally, the principles provide a framework with which to assess whether a protocol is both suitable and secure for a particular use case. While these principles were written primarily with internet protocols in mind, they may have wider applicability and utility to any protocol designer who wants to provide users with secure protocols.

Users and stakeholders should be front and centre in the protocol design process, and we believe these principles will help achieve that. These principles provide a vocabulary for protocol and system designers, administrators and many other stakeholders to evaluate protocols, to get involved in ongoing conversations around protocol design and to ensure their needs are met.

How should these principles be used?

A key first part of protocol design is taking the time to establish what the use cases are for the protocol and who its users will be. From there, it is important to consider the specific threat model that the protocol will face; achieving good security isn't possible without first understanding the threats that need to be mitigated. These principles provide a systematic but flexible guide to allow protocol designers to:

1. Prioritise the use case – by considering the context around the protocol.
2. Keep it simple – use this contextual understanding to keep the design simple.
3. Think about the bigger picture – make relevant protocol design decisions.

These are principles, not directives. Under every principle are a few prompts and explanatory text that protocol designers should read and consider. Not every prompt will be relevant to every protocol and use case, but they should all be considered. They are all important topics, and if a protocol does not meet any of them then it should be a conscious and thought-through decision by the designer. Further, the impacts of that decision should be outlined to users, implementers and deployers.

Contents

Introduction

Over the last twenty years, the internet has changed beyond recognition, and it will continue to change.

The number of internet users has grown massively, and the number of endpoints has grown even faster. The types of user (and the variety of devices they use to connect to the internet) are both much more diverse than twenty years ago. We've moved from a world of desktops talking to servers, to one where children have smartphones and a group of kitchen appliances can launch a Distributed Denial-of-Service (DDoS) attack.

This white paper defines a set of design principles for protocols, created to address these fundamental changes in internet use and the continually developing threat landscape. As threats and use cases evolve, internet protocols should too. The principles in this white paper put user needs at the heart of the design process to make sure that protocols provide functionality and security for users. This set of principles doesn't attempt or claim to cover every aspect of protocol design, but it does cover some of the most vital issues.

These principles are aimed at the designers of protocols, to help them produce secure protocols standards and implementations that are suitable for today's and tomorrow's internet. While these principles were written primarily with internet protocols in mind, they may have wider applicability and utility to any protocol designer who wants to provide users with secure protocols that meet their needs. The principles should be considered early in the protocol design process, but can also be used to review existing protocols.

How has the internet changed?

A full examination of how the internet has changed during the last twenty years and continues to evolve is beyond this document's scope. What follows is a non-exhaustive summary of the more substantial changes. These are:

1. The richer multi-stakeholder environment.
2. The architectural changes as the use of Content Delivery Networks (CDNs), cloud computing and edge computing grows.
3. The massive rise in cyber attacks, threat actors and their sophistication.
4. The growth of an increasingly complex security ecosystem.
5. The increase in diversity of user and device profiles.

Multi-stakeholder environment

The internet is a rich ecosystem that is enabled by a range of stakeholders – their roles are changing, becoming increasingly important and need to be considered. Even in a relatively simple case, like an Internet of Things (IoT) device in a home, there are many stakeholders:

- the owner of the data who needs their personal data to be protected
- the owner and operator of a device who wants to ensure its secure operation
- the manufacturer of the device who wants to make sure it's not malfunctioning
- a provider of a service that the device uses, who wants to confirm they're getting and providing all of the data they need to
- the ISP or telecommunications provider that needs to provide a reliable and performant connection
- a company performing data analytics for the device manufacturer

All of these parties and more have important roles in ensuring a secure service and providing the functionality the user expects, where they didn't before. In addition, traditional roles are shifting, with more previously lower level responsibilities being undertaken at the application layer. As a consequence, some of these stakeholders may now be liable to provide aspects of the service, and hence need to ensure a good user experience for the aspects they supply. The interests of the end user are paramount[1], and should be favoured over those of other parties, noting that for the end user to get the functionality they require, other stakeholders need to be able to perform their roles.

[1] https://www.rfc-editor.org/rfc/rfc8890.html

Architectural changes

The architecture of the internet has changed. A big part of this is the growth in CDNs; global networks of servers that serve content on behalf of their customers. CDNs aim to provide a consistent, fast and reliable service to their customers' users. In practice, this means that those users won't connect directly to the server they might expect to; instead they connect to a server owned by the CDN that delivers the content on behalf of (or routes the user to) that server. The routing of large amounts of traffic is therefore controlled by CDNs, rather than chosen by the user's ISP, representing a shift of architecture and stakeholder role.

CDNs have also driven a trend towards edge computing, where more processing is performed nearer to user devices - at the 'edge' of the network. In the CDN case, this means delivering content to the user from a server geographically close to them, to reduce latency. The growth in edge computing has also meant increasing geographical distribution of processing power, enabling CDNs to improve performance.

The growth of cloud computing has also been a major architectural change. This has provided convenience for everyday users, supported growing numbers of IoT devices, and encouraged a dramatic shift in enterprise IT from local to cloud-based solutions. For enterprises, cloud computing provides access to agile, convenient, robust and scalable solutions. These could be in the form of SaaS (Software as a Service) products such as multi-user messaging and video apps or IaaS (Infrastructure as a Service) products that allow enterprises to add and remove servers with ease, as they are needed. While edge computing has begun to diversify processing power geographically, increased use of cloud computing means that more and more hosting is done in large data centres owned by cloud computing services, routing traffic to a relatively small number of locations owned by a small number of providers.

Rise in cyber attacks

Cyber attacks are increasing in scale, ingenuity, and variety. There are a range of reasons for this increase, but two key reasons are the increased opportunity for malicious actors to conduct cyber attacks and the increased impact of those attacks.

Increased opportunity. There are far more internet-connected endpoints than twenty years ago, and hence there are more devices that are poorly secured and vulnerable to attack. Broadly available 'off-the-shelf' attacker tools have increased in sophistication, number and potency. This means that it's easy for criminals with little experience to use existing malware to conduct cyber attacks with widely felt impact. Attacker tools are widely available, reducing the bar to entry, but also encouraging waves of similar attacks as tools are incrementally changed to defeat mitigations and increase target space. Increased consumer bandwidth has augmented the power of botnets, such as Mirai[2] and its successors, to conduct massively disruptive DDOS attacks.

Increased impact. A breach of a single entity can affect the privacy of millions of users. Each of those users likely has accounts with many other services, so the loss of personal data or passwords can lead to further accounts being breached from otherwise secure services. It's not only privacy, however, that is impacted by cyber attacks; the impact on availability of services can be equally significant. The world is increasingly connected – compromised industrial control systems and IoT devices can lead to physical consequences, and the compromise of internet-connected endpoints relied on by vital services can have massive effects. The WannaCry ransomware famously impacted the UK NHS[3], as vital systems were disrupted, putting patient health at risk. The large and ever increasing variety of attacks can make it hard for defenders to keep up, and the quick turnaround of attacks means new mitigations are needed all the time.

For most users, the most likely threat to their privacy online is the breach of a poorly secured system at an airline, hotel chain or social media platform - leaving an attacker with access to all of their personal details. Fixing these vulnerable systems is therefore a priority, but the security of the protocols they interact with is just as critical.

Increasingly complex security ecosystem

Increasingly, security means more than just encryption, not least because authentication is just as important. Looking out for a padlock in your browser's address bar gives no security if you're communicating directly with an attacker. An attacker using a password obtained in a breach appears identical to a user accessing their account legitimately. Many devices have no way to provide even simple trust indicators to users, nor assess the legitimacy of connections. Though the burden of assessing legitimacy should not be on users, and devices should be secure by default, in practice, this is often not the case. This highlights the growing need to consider the environment the protocol is operating in – the increasingly complex ecosystem means that assumptions about security are very hard to make.

[2] https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/

[3] https://www.bbc.co.uk/news/health-39899646

Protocols form just one part of the security ecosystem. In the same way that using an encrypted protocol to communicate with an attacker misses the bigger picture, using secure protocols that undermine security elsewhere (for example by hindering malware detection or concentrating user data in the hands of a few service providers) can be counterproductive. No protocol operates in a vacuum, and they need to support security, in the present and future, in all aspects of that ecosystem.

Diverse users and devices

With the massive growth in user and device diversity - as well as the multiple stakeholders in even the simplest interaction - the variety in use cases for internet protocols is now vast. Advances in technology, and its affordability, mean that everyday consumers have many more internet-connected devices than they used to, from mobile phones to smart fridges. Not every protocol has the same job, not every protocol operates in the same ecosystem and not every protocol has the same user base. These are self-evident statements but, together, they lead to the conclusion that each protocol has a unique threat surface and different requirements on how it should address those threats. Some protocols have a diverse range of use cases - considering all of those use cases is important, and improvements made to support one use case could be detrimental when the protocol is used for another.

Principles for an evolving threat landscape

Against the backdrop of these changes, trying to define an entirely new threat model to make a 'one size fits all' solution is not the right approach. Instead of conducting threat modelling tickbox exercises, protocol designers should develop threat models unique to each protocol, supported by a framework of key principles that every protocol designer should consider. The NCSC believes that these principles will help designers produce protocols that are not just secure against these new threats, but also meet user needs.

Motivation for the principles

In 2019, the NCSC published its Secure design principles for designers of secure systems[4]. That guidance, summarised below, provides the motivation for the protocol design principles described in the following section. By considering the three principles defined later in this document, protocol designers will be able to deliver towards each of the following five goals.

Establish the context

Each protocol will be used differently, by different users and for different purposes. It's important to take users, use cases, risks and threats into account when designing the protocol. Consider the unique threat model of the protocol, defining that threat model clearly, and ensuring that the protocol addresses the most important identified threats. It's also important to identify those threats which aren't addressed to allow users to take them into account when deploying the protocol.

Make compromise difficult

This goal protects against data compromise through unauthorised violation of data confidentiality, data integrity or data authenticity. A large part of this is communications security, which is obviously key to secure protocol design, and more generally protecting the privacy of users by keeping personal information secure. The different types of compromise that could be made difficult are determined by the context of the protocol, with decisions driven by how to minimise the attack surface most effectively. Preventing some methods of compromise can increase the vulnerability of the system to other methods, so it is vital to take the protocol's whole threat model into account, considering the impact of design decisions on use cases and the ability to protect against other threats.

Make disruption difficult

Considering how best to deal with both malicious attempts to disrupt (like a DDoS attack) and non-malicious disruption (such as a server failing) are important aspects of secure protocol design. This goal is even more important for the many protocols used in critical contexts where disruption (which could be gaps in communication, high latency or unreliable service) cannot be tolerated.

Make detecting compromises easier

At some point, an endpoint or application will be compromised. Protocols should facilitate detection of an intrusion and not aim to prevent such detection, either during or after the event. Protocol designs should consider how the protocol may be misused when an endpoint is compromised, and how that misuse could be detected, for example by exfiltration of stolen data. Compromise of an endpoint may also mean any security mechanisms it has are compromised, and could become attack vectors themselves. Though security methods are evolving, relying on host-based security alone has limitations and is not always an option, so mechanisms which allow non-endpoint components to detect and thwart compromise should be considered. This is particularly important when designing for environments such as IoT, where devices may have only limited or no host-based security.

Similarly, when profiling past malicious behaviour to detect future compromise, it is important that the user (and appropriate admins) are able to analyse what has happened to their own equipment and their own data in a way that preserves user privacy; protocols should enable this analysis to be performed. If a protocol hides this kind of information, then defenders may have to rely on logs held on a compromised endpoint (or may have no logs at all).

Reduce the impact of compromise

Once a compromise has been detected, the user (or their network administrator, service provider or other stakeholder) needs to be able to reduce its impact. This includes minimising the amount of data an attacker can access or exfiltrate, minimising an attacker's impact and use of data, pre-emptive action to minimise the scope of any future compromise, minimising the impact on service availability and taking active measures to protect users. Limiting an attacker's access to data is a key consideration, for example by using short-lived credentials to reduce the window during which an attacker can exploit those credentials if compromised. Designing protocols so that data exfiltration or command and control communications can be blocked and signatured is not necessarily an easy problem, especially while protecting the content of communications, but is nonetheless vital to reduce impact.

[4] https://www.ncsc.gov.uk/collection/cyber-security-design-principles

Protocol design principles

The principles described in this section will help protocol designers to create protocols that achieve the motivational goals outlined above. These principles capture aspects of secure protocol design that are important in the evolving threat landscape. Within each principle are several prompts for protocol designers to consider; these prompts cover a range of ways to apply the principle to protocol design.

Principle 1: Prioritise the use case

User needs should be at the heart of the protocol design process. Prioritising use cases and users, while also considering the protocol's context, when making design decisions will ensure that a protocol provides functionality and security for users.

1.1 Consider the context

Before beginning protocol design, it is important to take a step back and consider the protocol's context. Establishing the context of a protocol is essential to good protocol design. This means thinking not only about where the protocol will be used and what threats it will face, but also dependence on any stakeholders, who will use it and how. A big part of this is considering the unique threat model for the protocol - only once this is clearly laid out is it possible to decide whether the protocol is secure.

Protocols with different user bases or operating environments will likely have very different threat models and very different security requirements. For example, when writing protocols that will operate in high latency or constrained environments, designers will consider the importance of protecting against certain threats balanced against operational concerns. The speed needed, battery constraints and processing power requirements for protocols running on (for example) IoT devices will contrast greatly to those running in data centres. The assumption of physically secure endpoints is also often made implicitly but, for many use cases, this assumption is unwise. Different user bases will also assign conflicting priorities to different security concerns, for example absolute confidentiality of communications weighed against ease of malware detection, leading to what may be contradictory pressures on protocol designers.

Some protocols will have a variety of use cases and operate in a variety of contexts. All of these should be considered in the design process if the protocol is to be secure for all its users. It won't always be possible to meet all of the desired use cases, in which case the design should make it clear when the protocol should and should not be used. As they become successful, it is likely protocols will be considered for a range of use cases beyond those they were designed for – this clear assessment in the design should give potential users enough information to decide if it is appropriate for their use case, or whether a new protocol is needed.

1.2 Think about what you reveal

The more information that can be seen in the clear, the more privacy is lost, and the less work an attacker must do to see user data and to compromise other user data. For example, unencrypted Transmission Control Protocol (TCP) traffic can allow an attacker to read packets, whereas unencrypted Domain Name System (DNS) requests can allow an attacker present on the network to build up a picture of a user's browsing habits. Both attacks can be mitigated by encrypting the relevant data, but they carry very different risk profiles, with severity to be judged according to the use case and threat model.

Metadata also matters. Many protocols need metadata to provide functionality (e.g. routing) and metadata is often used by trusted tools to improve security. Additionally, some protocols need metadata to discover other systems or indicate their presence to them. However, metadata can also leak information about users. If this is done gratuitously, then attackers have a head start for future attacks; at a minimum, they could begin to build a picture of a user's habits or of vulnerable software present on a particular device.

1.3 Think about what you trust

Traditionally, system designers have placed trust in a specific network, but, with movement toward 'zero-trust' architectures[5], this is becoming less common. It is, therefore, rarely sensible to assume that protocols operate in a wholly trusted environment. Trust is not a constant, and not all or nothing - legitimate users can become adversaries, devices can be stolen or compromised. Endpoints can also be unpatched but not exploited; malware can be cleaned-up and trust restored. These examples illustrate why network administrators cannot trust all nodes absolutely in a trusted network. Detecting a compromised node exfiltrating data (or communicating with command and control nodes) is vital to network defence.

[5] https://www.ncsc.gov.uk/blog-post/zero-trust-architecture-design-principles

Most protocols will operate in a constantly changing trust landscape. An endpoint could be compromised at any point, so it is important to consider how a protocol should operate if one or more endpoints are not trusted. Revocation of trust is a valuable tool in these cases, but has proven complex to implement successfully, leading to hard failures and UI warnings or even acting as an attack vector itself. Considering ease of revocation (or reduction) of trust should be key for designers.

The problem of establishing trust in an endpoint is also increasingly important. Bootstrapping trust is a hard problem, and being able to restore trust to an endpoint is vital for resilience. Allowing use of mature, proven services for identity management, key exchange and revocation, such as those provided by an operating system, is a good way to manage trust. Finding the right balance of trust is important and should take account of a range of factors, including levels of trust in the endpoints, levels of trust in the network and the impact of an undetected compromise.

1.4 Consider detection of misuse

A key method for detecting compromise is to detect activity linked to that compromise. This means distinguishing it from expected activity – metadata for malicious behaviour will usually look quite different to the normal activity from that device. Certain protocol artefacts, such as IP addresses or domain names linked to malware, are known as Indicators of Compromise (IoCs)[6]. These IoCs are a vital part of the cyber defence strategy for many organisations and individuals, and are easy to share, helping to limit the scale of compromise. This detection, using IoCs or otherwise, cannot take place if a protocol is designed such that distinguishing different types of traffic is impossible. A protocol can take different routes to allow detection of misuse – either a built-in ability to detect misuse or making information (such as IoCs) available to other services – but whatever solution is chosen should meet the use case and any limitations should be made explicit in the design.

There are also use cases that prioritise integrity (for example non-sensitive control messages), and in these cases, sending data over an encrypted tunnel can mean that a security device is unable to decrypt the traffic and so cannot detect misuse. However, taking an approach of providing only integrity to the messages allows security appliances to do their job without compromising the integrity required by the recipient device. Integrity is often closely linked with confidentiality, so can be difficult to do well in isolation, but an integrity-only solution should be considered if it meets the security requirements of the use case.

The value an attacker gains from knowledge of an artefact might be less than the value of an admin knowing what is traversing their network - implementing blanket encryption could prevent a defender from spotting malicious activity. There's strong asymmetry here; large scale defensive monitoring must be very efficient while an adversary can choose to focus massive effort on a single packet. Logs are also a vital tool for defenders, and protocols should support logging that allows detection of misuse. It's important to consider the use case and threat model when deciding what should be logged and when developing error messages.

Preventing potential data breaches, both immediately and in the future, through detecting compromise needs to be balanced against the ability of attackers to access traffic - but for many users and use cases, enabling defence will be of prime importance. Finding the right balance is important to both ensure the security of users and to protect their privacy.

1.5 Consider resilience to both attack and failure

Not every security problem is due to a malicious attacker, and not every threat happens purely in the digital realm. A system - comprising networks, machines and protocols - can be compromised without malicious intent, and it is important that recovery is possible in these situations. Such a compromise could be a loss of timing synchronisation, a software bug, a corrupted hard disk or even a natural disaster resulting in infrastructure outage. Ensuring data remains secure if an endpoint becomes unavailable should be an important consideration, as should the operation of endpoints when they are offline. Protocols designed with lack of availability in mind can help this recovery. Of course, a situation where all hardware conducting the protocol is destroyed cannot be fully accounted for, but the level of resilience should be determined by the use case and made explicit in design.

Another way that a protocol design can help a system to be resilient is to outline possible failure cases, and what mitigations should be taken if they occur. If an event should not happen, it may well still do so, and, if possible, a user should be equipped with a well-defined, secure mechanism with which to recover. Considering these failure states during the design process will help produce a resilient protocol that is able to recover properly, and retain security, in a range of situations.

[6] https://datatracker.ietf.org/doc/draft-paine-smart-indicators-of-compromise/

1.6 Enable defence-in-depth

Defence-in-depth is a cornerstone of modern cyber defence. The goal is simple; use layers of defence with several different mitigations at each layer. This approach provides multiple opportunities to detect compromises at different parts of the system, to prevent them causing further damage, and to account for different points of failure. Indicators of compromise, as discussed in 1.4, can be detected both at the network level and on endpoints, meaning they are a perfect fit for this defence-in-depth resilience.

Different use cases will place different priorities on the CIA triad of confidentiality, integrity, and availability. Providing the resilience across components required when using a defence-in-depth approach means that certain information must be made available to the relevant components to allow them to perform their functions. This means that even if one or more security appliances fail, or are compromised, then defence is maintained and privacy is protected. Protocol designers should consider overall systems, and how they would apply and allow defence-in-depth, when considering how to mitigate different attacks and how to combine confidentiality, integrity, and availability.

Principle 2: Keep it simple

Ease of use is a virtue - designs that enable users to operate securely with minimal intervention (or technical knowledge) make a vast difference to the security of many ordinary users. Protocol designs that are simple to implement and use previously established components can allow for easier evolution and avoid vulnerabilities that arise from design complexity.

2.1 Avoid design and deployment complexity

Vulnerabilities often arise due to complexity in both products and protocols. Designers should aim to avoid unnecessary complexity, so the protocol can be well understood at design time, modelled, and receive a full security analysis. Open review is a crucial part of the protocol design process, allowing issues to be found and addressed before they cause issues for either implementers or users. Taking design decisions that allow for easy, open review, by limiting complex logic, therefore builds both the security and confidence in the security of the protocol.

Protocols should also be designed to allow for simple implementations that reduce the risk of non-deterministic states and scope for errors at implementation time. Simple implementations made possible by the protocol design, such as those with modularisation or code re-use, enable meaningful analysis and reduce the scope for complex code with opaque justifications. Designing protocols to build on existing mechanisms, such as hardware roots of trust, can also reduce implementation complexity. Providing lots of intricate configuration options or modes of operation can make designs more complicated and much harder to test; prioritise the use case when deciding which options and modes are necessary. Limiting options and modes makes it easier to verify that the protocol has the claimed protections against its threat model, ensuring the user has the security they need.

Complexity is to be avoided in deployment of protocols too. Overly complicated, or ill-defined methods of deployment for a protocol mean it is more likely to be deployed incorrectly or not deployed at all. Interoperability is an important property of protocols that can be impacted if deployment is not done well. While not an inherent part of protocol design, deployment is heavily influenced by protocol design decisions, and should therefore be considered during that process. A secure protocol must be deployed securely to protect user privacy.

2.2 Make it easy to use securely

Most users don’t know how the internet works or what protocols they’re using, and nor should they need to. They want to use products or services, not configure them. Default settings should be configured in a way that balances user needs with security. This means that users should have secure choices made for them by default. Further, these defaults do not obstruct a user from being protected by other common security measures and do not allow an attacker to disable such choices easily. Sensible defaults will help users improve privacy of communications and protect endpoints from compromise.

Protocols are also often designed with security built-in, but without user-centric security in mind. This can result in protocols that lead users to misunderstand the security they're getting, be it belief in the identity of the 'From' field in an email, or the level of trust they should place in an 'https' website. It's also common for protocols to be designed with the assumption that all users are equivalent, and have the same security, privacy, understanding of threats, and safety needs and wants. In reality, users vary greatly, with very different levels of vulnerability to different types of threats.

Protocols should be designed with users and deployment models in mind, such that they meet users’ basic security expectations by default, and allow for those with more specific requirements to be catered for.

2.3 Don't reinvent the wheel

Use existing, tested and well-established technologies where possible. Using proven building blocks makes it easier to have confidence in the constituent parts of a protocol. Not only will security analyses likely exist for those standard components already, but so will tried and trusted implementations. While use of existing components is no guarantee of security, making things easier for implementers, by facilitating use of well-tested libraries, is more likely to result in secure implementations. A prime example of not reinventing the wheel is to use standard, well-understood cryptography in protocols. If an algorithm has been in wide use, and has robust implementations available, it is less likely to:

- have any undiscovered security flaws
- be implemented incorrectly (when compared to a bespoke implementation of a new algorithm)

A good example of this is authentication - a key component of secure communication. Many good authentication solutions already exist, so where a protocol implements its own authentication, it could be taking extra risks. Techniques like single sign-on (SSO) allow a protocol to use an existing, robust authentication method. Clearly it is important to consider the use case and use a method that is operating to a similar threat model to the protocol being designed; using an established but inappropriate method may introduce (rather than reduce) weakness.

2.4 Design for easy maintenance and upgrades

Patching is important; it prevents systems remaining vulnerable to old attacks that should no longer be effective. Patching needs to be made simple or many admins won't have the time (or inclination) to do it. A non-savvy user can't be expected to make a sensible choice between a seemingly small chance of compromise and a convoluted update process or a compatibility-constraining patch. For critical systems that cannot afford any down time, a protocol that is not easy to patch or maintain can become insecure as operational priorities may override security considerations.

Protocols will often be used in ways not anticipated by their original use case - if there is a way to provide support for new use cases through easy extensibility, this is usually valuable. When extending a protocol, any new use cases must be compatible with the original threat model to ensure that security guarantees are maintained. Providing simple extensibility and flexibility in the design from the beginning can boost security by protecting against insecure use of the protocol or the need for workarounds that are inconvenient or inefficient. Well-designed extensibility can also remove the need for patches that damage interoperability and prevent deployers needing to switch to a whole new protocol – helping to promote a healthy ecosystem.

While implementation of patches is beyond the scope of protocol design, patching can be made easier in a range of ways. One way is to keep the protocol design simple, which reduces the likelihood of ossification (loss of extensibility through inextensible implementations), therefore making it easier for different versions to interoperate, removing a risk of patching. Stateless designs not only usually make for easier implementation, but also, along with designs that enable load-balancing or handing off responsibility, make it easier to patch servers without having to incur downtime.

Allowing for easy patches and updates also allows for 'crypto-agility' (updating and replacing cryptographic algorithms used within the protocol). If implementations of a protocol are hard to update, then making a change to the protocol design through patching is likely to be ineffective or even harmful to security. It could lead to the protocol's ecosystem splintering with inconsistent security guarantees between different users, compromising the privacy of all users, including those who upgraded. Designing the protocol in such a way that adding a new algorithm, or deprecating an old one, is a simple change will help its implementations, and the ecosystem it operates in, to stay secure.

Principle 3: Think about the bigger picture

No protocol exists in isolation. Protocols designed today may have unintended consequences or be influenced in unexpected ways, from security concerns directly related to the protocol (such as where cryptographic keys are stored) through implications for software and devices (such as side-channel attacks) to impacts on the wider internet ecosystem (such as market diversity, introduction of DDoS amplification attacks or other protocols). Users need more than just a secure protocol, they need a secure and robust ecosystem, now and in the future – considering the impacts of a protocol design is vital in achieving that.

3.1 Encourage diversity

Service diversity should be encouraged because it allows users to make the best choice to support their use cases and privacy. A sufficiently broad market of services would give commercial success to participants and allow new profitable entrants to the market, while providing good user choice. However, if protocols are designed in such a way that constrains the ability for services to succeed in a well-functioning market, then this limits user choice.

Design decisions, including the mechanism and architecture chosen, can contribute to a lack of provider diversity. For example, if a new protocol is hard to upgrade to or implement, then only larger stakeholders will initially have the ability to do so, enabling market capture at an early stage and potentially leading to single points of failure.

Market dominance effects, resulting in concentration (and possible commercialisation) of users' personal data are important considerations for wider society and user privacy. If a service has only a handful of providers then user choice is restricted, making it more likely that users are forced into making undesirable privacy-critical decisions – for example using a service provider they would not otherwise trust with their personal data.

Lack of service diversity also increases the likelihood of more widely shared vulnerabilities, which give attackers a better chance of disruption or compromise on a large scale. Attackers' motivations may also increase correspondingly. Diversity of service provision can therefore increase the security and resilience of the whole ecosystem.

3.2 Consider secure implementation

A protocol may use cryptographically proven methods, but traditional communications security and cryptanalytic attacks are just some of the threats that protocols may face. Further threats include side-channel attacks (where an attacker observes information about an implementation of a protocol which can be used to derive secret information) and social engineering attacks (such as phishing, where a user is tricked into revealing private information to an attacker). Through due consideration of the endpoint environment that the protocol operates in, it is often possible for protocol designers to not only make secure implementation easier, but also to foresee implementation choices that enable alternative attacks and warn against them.

Formal verification of protocol security is growing in popularity, especially for protocols that will be widely used, and hence have their security highly relied upon and regularly challenged. Conducting this sort of analysis on the state machine underlying the protocol is not cheap or easy but can be desirable for some protocols. Designing the protocol in such a way to enable this analysis (see 2.1) or making it part of the design process, is likely to be increasingly common in the future.

Protocol design decisions can also help developers implement protocols correctly. A simple design (see 2.1) is one way to help achieve this. Providing test vectors, and other metrics to measure implementations, is another way to make secure implementation easier; this enables developers to ensure that their implementation acts as expected, and also gives users confidence that an implementation they are using is secure. Thinking about how a protocol will scale is also valuable – a design that remains robust, performant and secure is not only better for deployers but can still provide security when deployed at an unintended scale.

3.3 Allow use of the least privilege

Data breaches happen every day and this is not about to change. Devices and databases will be compromised, so our motivating goal to reduce the impact of compromise is crucial. The principle of least privilege is to restrict each component to only have access to the information necessary for it to perform the functions for its job. Restricting permissions appropriately is a vital part of security - maintaining user privacy while delivering rich-enough functionality.

If a protocol design means that components in a system are given greater privilege than they really need, the consequences of a compromise of one of those components are correspondingly worse than they need to be. For example, a protocol that requires constant access to cryptographic keys in turn means that a malicious actor gaining access to an endpoint running that protocol has access to those keys. This not only compromises the protocol, but any other data secured using those keys, potentially causing a massive breach of privacy. Recovering from a compromise is also much easier if least privilege is applied, by, for example, using tightly scoped or short-term credentials which can be easily revoked, as this drastically limits the window and extent of the compromise.

Applying the principle of least privilege in protocol design therefore not only lessens the impact of a breach by limiting the data an attacker has access to but can also aid recovery by limiting the scope of the compromise to recover from. By allowing privilege to be limited in this way, a breach will compromise privacy as little as possible while ensuring that a system still functions effectively.

3.4 Plan for failure of third parties

All internet users are dependent on third parties to ensure their security and hence maintain their privacy. Using Transport Layer Security (TLS) eventually comes down to trusting a Certificate Authority (CA), while protection from Denial-of-Service (DoS) attacks is often only provided by one of a handful of big providers. If these third parties are compromised, become unavailable, or even stop functioning as expected, then the user may be vulnerable as a result. Planning for this includes mitigations like allowing expiry or revocation of credentials, if, for example, a CA is compromised. For protocols that rely on DNS this could mean considering the usability of the protocol if the necessary DNS records are unavailable. Protocols should allow a user to, at a minimum, understand what assurances are being provided by critical third parties, but ideally should provide some mitigation for compromise of those third parties.

3.5 Support evolution

Like the security landscape, protocols are rarely static; their implementations will grow and diverge with time. Bugs or undefined behaviour in one implementation soon spread to others due to interoperability concerns. These undefined behaviours may well be insecure, and can be effectively impossible to patch as, rightly or wrongly, interoperability concerns often override security concerns. Ideally, interoperability versus security is not a choice that users should have to make.

The robustness principle[7] is to be conservative in what you send, but liberal in what you accept. Unfortunately, this approach can encourage non-conformant implementations to survive and grow. In a world of intolerant receivers, these implementations would not be able to persist, improving the likelihood of only correct, secure implementations being used. Deployment of mechanisms such as GREASE [RFC8701][8], a lightweight way to prevent ossification (see 2.4) and extensibility failures caused by incorrect implementations of TLS, can help drive the creation of a consistent, secure ecosystem of correct implementations.

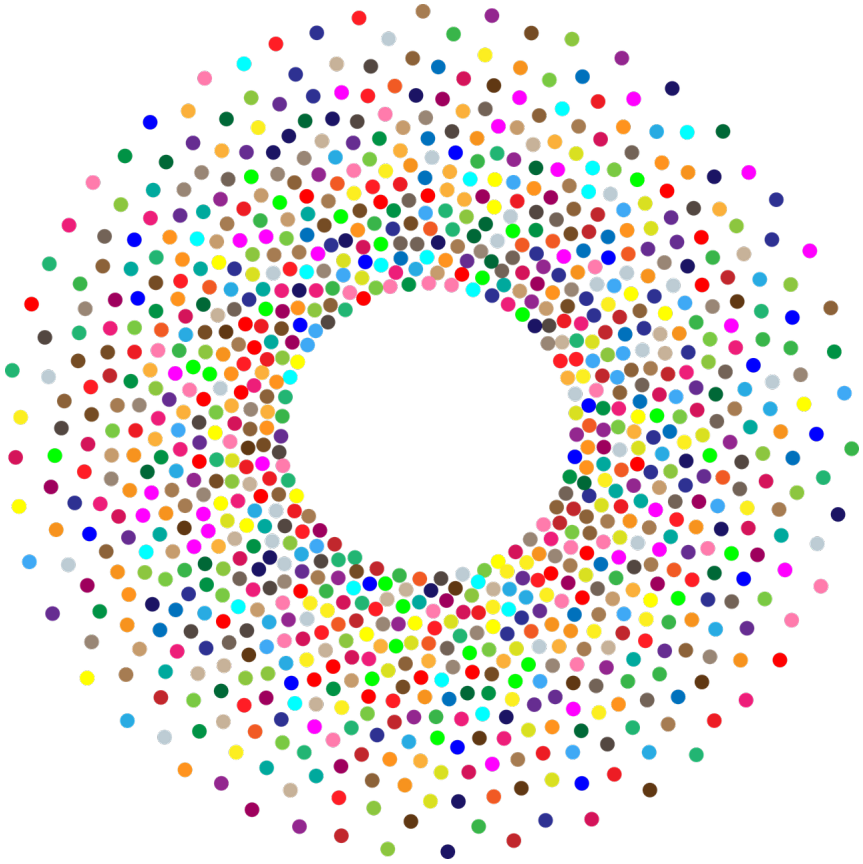
Protocols need to evolve, and designs should support this. Including extension points, and building an ecosystem so that they are preserved, supports evolution, as does easing interoperability between different versions by building in the capacity for version or parameter negotiation. How a protocol is designed will influence its deployment, which will have a massive impact on who uses it and how it is used - which in turn affects how the protocol should evolve. Considering deployment when designing the protocol will inform decisions about how best to support evolution and allow the impacts of how it will be deployed to be made clear to users.

Consider the appropriate balance between interoperability, strictness, agility and evolution for each protocol – striking the right balance will help to ensure that the protocol operates, and continues to operate, in a healthy, secure ecosystem.

[7] https://en.wikipedia.org/wiki/Robustness_principle
[8] https://tools.ietf.org/html/rfc8701

National Cyber Security Centre
a part of GCHQ

Protocol Design Principles



A white paper from the NCSC