Wireless Personal Communications (2020) 113:1615–1639
https://doi.org/10.1007/s11277-020-07310-5

A Philosophy of Security Architecture Design

Geir M. Køien[1]

Published online: 30 April 2020

Abstract
Digital systems are almost always vulnerable, yet we increasingly depend on these systems. There will be many threats towards these system. In a fully networked system, the vulnerabilities will literally be exposed to the whole world. The exposed vulnerabilities may be transformed into attacks. To counter this state of "vulnerability", the standard remedy is to conduct security requirements analysis and security threat modeling. Threats are assessed, and various countermeasures are devised. The totality of these measures may be described as a security architecture. The goal of a security architecture will largely be to make the system robust and resilient in the face of an adversary. However, we shall argue that this is not enough. Security architecture designs should go one step further, and actually improve the defenses when faced with hostile actions. That is, the security architectures must become antifragile.

Keywords Security architectures · Vulnerability · Exposure · Scalability · Intruder · Threats · Threat model · Risk · Robustness · Resilience · Antifragile

1 Introduction

The requirements for a security architecture is very much about the level of uncertainty one wants to endure and the risks one is willing to take.

1.1 Why Philosophy?

There certainly are many technical aspects of modern information and communications technology (ICT) systems and the associated security architectures. Indeed, most of the aspect of *how* to achieve the goals tend to be of a technical nature. However, questions concerning *why* need not be technical at all. That is, on a systems level, the end goal of a security architecture are normally not technical in nature. Instead, the goals tend to be more philosophical. They may be framed in a context of moral and ethics, and sometimes in the framework of legislation and societal rules.

✉ Geir M. Køien
geir.koien@usn.no

1 University of South-Eastern Norway (USN), Campus Vestfold, Horten, Norway

The distinction between the technical and concrete aspects and the philosophical aspects can be conceptualized as the difference between *verification* and *validation*. Verification is largely about checking that something is done the correct way, whereas validation is concerned about whether one is doing the right thing. It is of little use to do something correctly, if one is indeed doing the wrong thing in the first place.

Modern ICT infrastructures are becoming integrated into our lives in many ways, and our society is poised to become even more dependent on these ICT systems. This means that safe and secure operations of these critical infrastructures literally becomes a matter of life and death. No system can be made totally safe, but how safe should we try to make it? There may be efficiency penalties and incurred costs if security is to be improved, but then there may be human casualties if the security is inadequate. There are economical aspects to this, but also moral and ethical ones. There are costs to having an inadequate system, but there are also costs to not having a system. What can be justified, and what cannot? These are complex matters. This is why there is a need for a philosophical stance when it comes to security architectures.

1.2 Security Versus Safety and Privacy

The need for security, safety and privacy is in many ways self-evident. Large-scale critical infrastructures is essential to society, and so the level of security, safety and privacy becomes a question about what kind of society one wants to have. We shall not dive into safety and privacy in this paper. However, we argue that strong security is a necessary condition for both safety and privacy. This puts emphasis on the importance of an effective and comprehensive security architecture. Informally, the differences and relationships between security, safety and privacy can be stated as follows.

- *Security is about protection of the system.*
  Security is about the system, and protection of system entities and assets against threats. Security specifically encompasses protection directed towards deliberate (malicious) threats.
- *Safety is about protection of people.*
  Safety is about being protected from harm, including actions that may cause harm. This includes risk control. Safety is often concerned about unintended consequences: that is, causes without any ill-intent behind.
- *Security versus Safety.*
  For critical infrastructures, one cannot have credible safety without having strong security. Strong security is therefore a condition for safety. Lack of security implies lack of safety, but strong security does not automatically lead to credible safety.
- *Privacy is about information control (related to persons).*
  Privacy is always related to individuals. It pertains information related (or linkable) to persons. This encompasses concepts such as freedom from unwanted/unauthorized intrusions, and to keep information private (secret). It also encompasses a right to have information deleted, not spread, etc.
- *Security versus Privacy.*
  For ICT systems, one cannot have credible privacy without having strong security. Strong security is therefore a condition for privacy. Lack of security implies lack of privacy, but strong security does not automatically lead to credible privacy

Given the above, it should be clear that strong security is a necessary, but not sufficient, prerequisite for both safety and privacy. Strong security will therefore need to be a system imperative.

1.3 The Incerto

The Incerto is a set of books and essays on uncertainty. Touted as an investigation of opacity, luck, uncertainty, probability, human error, risk, and decision making, the main body of the Incerto consists of five books by Nassim Nicolas Taleb. The books contains autobiographical sections, stories, parables, and philosophical, historical, and scientific discussions. The books are:

- Fooled by Randomness [1].
- The Black Swan: The Impact of the Highly Improbable [2].
- The Bed of Procrustes [3].
- Antifragile: Things That Gain From Disorder [4].
- Skin in the Game: Hidden Asymmetries in Daily Life [5].

These books forms an edifice of scientific thinking, statistical finesse, empirical advice, philosophical thinking and ethical attitude. It truly represents a multidisciplinary approach. And it must be said, it also represents Taleb's personal views and style (which can be idiosyncratic and quite irreverent). The Incerto is the inspiration behind this work on security architecture designs.

1.4 Key Concepts: Vulnerable, Robust and Antifragile

In [4], Taleb defines an axis where he lists the traits *fragile*, *robust* and *antifragile*. We shall prefer to use the term *vulnerable* instead of *fragile*, to keep closer to security parlance. Similarly, one may prefer to use *resilient* instead of *robust*. These changes does not alter the essence of the classification.First, we need to define the three archetype states for our target system. The arrow in Fig. 1 indicates that the space from fragile to antifragile is a continuum. Furthermore, a system will have many components, and these should be classified individually. Security system-state archetypes:

- *Vulnerable (and exposed)*
  There exists serious vulnerabilities and these are exposed externally.
- *Robust (resilient)*
  There may exist vulnerabilities and there may be exposure, but many steps are taken to mitigate weaknesses. The system is well defended.

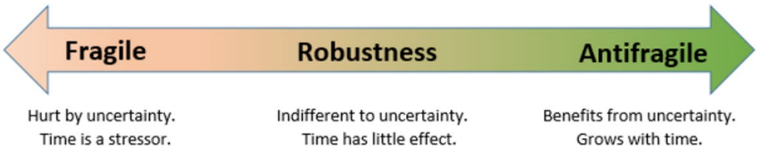

Fig. 1 Taleb's classification of fragile versus robust versus antifragile

- *Antifragile (becoming stronger in the face over adversity)*
  There may exist vulnerabilities and there may be exposure, but steps will be taken to mitigate weaknesses. Upon experiencing adversities, a host of measures will be taken to strengthen the system. One will learn from the incident, and improve on the system in as many ways as are practical.

The concept of archetype states is a tool for discussion, and not something to be taken as absolutes. We shall return to this topic in Sect. 4.

1.5 Normal Security Architecture Design Requirements

1.5.1 Requirements Capture

One normally carry out a requirements capture stage during the design phase. There will be a number of basic requirements. These will include requirements for identification schemes, entity authentication, and authorization and access control arrangements. There will also be requirements for confidentiality- and integrity protection for data in transit and for data at rest. During the last decade, requirements for privacy has also become prevalent.

High-level requirements for system hardening and server protection may be mentioned and recommended, but one tend to shy away from concrete recommendations. The same can be said about security measures for detection and response. This is to be expected. These parts of a security architecture are much more dynamic and reactive in nature, making attempts at concrete recommendations difficult since it will be a moving target.

1.5.2 Not Everything Will Be Standardized

Large-scale security architectures, as exemplified by organizations like the 3GPP (https://www.3gpp.org/), will tend to capture the above mentioned requirements. There will, however, be a number of optional features and aspects that simply is not standardized. Typically, the standardized security architecture will be (mostly) silent and agnostic about requirements concerning the system development phase and about the operations phase. Likewise, it will probably also be agnostic when it comes to system management aspects. There are also aspects that is related to business priorities, and those will normally not be part of any standardized specification.

To complete the security architecture, the security conscious operator will perform its own risk modeling, decide on a security policy and capture requirements correspondingly.

1.5.3 Proactive- Versus Reactive Measures

The design of the proactive parts of a security architecture will be explicitly specified. These are "design phase" requirements. It will be possible to have a complete and consistent design for the proactive measures. The dynamic/reactive parts of the security architecture, which will be dealing with incident detection and response, recovery, etc., will likely not be completely specified (if at all). What one can hope for is that the high-level requirements be precise, consistent, complete and clear. The other phases of a system life-cycle is less likely to be captured.

1.6 Threat Analysis Philosophy

Threat modeling schemes seem, in general, to have a preference for practicality over a strong theoretical foundation. That is, they are designed to be pragmatic and efficient in finding threats. Completeness is a non-goal, while cost-effectiveness is important. The goal is not limited to picking the "low-hanging fruit", but it certainly facilitates this aspect well. The STRIDE approach to threat modeling is one such a scheme [6] and the Center-of-Gravity (CoG) approach another [7, 8]. The pragmatic approach is not only about cost-efficiency, but also about feasibility. More complex and comprehensive schemes would require system specialists and tool experts in order to carry out the modeling and analysis. For many projects, this would be too costly and/or it would lead to unacceptable delays.

1.6.1 Basics Steps

The basic STRIDE process consists of four steps, roughly defined as:

- Define the model (the system architecture).
- Find threats related to the model (STRIDE classifications).
- Handle the threats (prevention/mitigation/...).
- Revise the model (and assess the threat countermeasures).

The STRIDE mnemonic is a somewhat contrived acronym, defining the following threat types: **S***poofing*, **T***ampering*, **R***epudiation*, **I***nformation disclosure*, **D***enial of Service (DoS)* and **E***levation of Privilege (EoP)*. Central to STRIDE is the "What can go wrong" ethos. One would normally prioritize analysing data flows that is directly initiated by the actors of the system.

1.6.2 Center-of-Gravity

Center-of-Gravity (CoG) analysis has it's background in military thinking, and traces its roots to Carl von Clausewitz book "On War" [9]. Clausewitz likely conceived of COG just subsequent of the Napoleonic wars. The first edition of the book dates back to 1832. A CoG item is some aspects that is deemed important and even essential to the system owner, and CoG threat modeling will focus on those aspects. This provides for a directed strategic focus.

1.6.3 Threat Management

Threats can be handled in many ways. Some common responses are:

- Prevention (remove the threat).
- Mitigation (lessen the impact of the threat).
- Transfer the risk (insurance, let the end-user take the risk, ...).
- Accept the risk (some risks are worth taking).
- Schedule handling for later.

All of these approaches may be appropriate for some type of threat. Importantly, all actions are supposed to be informed actions. That is, one presumably understands the risks and uncertainties involved. We shall return to the topic of threat modeling in Sect. 7.

1.7 Related Work

There does not seem to be much work done on security architecture designs for large-scale critical infrastructures. Frederick Brooks, of "The Mythical Man-Month" fame, has written extensively about system designs in "The Design of Designs" [10]. This book has served as a useful background for the paper. Other works also cover system designs (Chapter 10 in [11]) and [12]. Buinevich et al. [13] provides an account related to IoT systems and Monperrus [14] provides principles for antifragile software development. There are a couple of papers on security architectures for mobile systems [15, 16], which are relevant. The classic work on computer security architecture may be the 1975 paper "The Protection of Information in Computer Systems" [17]. There are quite a few papers on topics related to organizations, various related to public utility services (water, electricity, etc.), but these are not very relevant for the present paper.

When it comes to antifragile system designs, there have been a few academic papers the last few years, but not many that cover critical infrastructures or security designs. One notable effort is the book "Anti-fragile ICT Systems" [18]. This book has been an inspiration for this paper, and in particular chapter 4 "Principles Ensuring Anti-Fragility" has influenced this paper.

1.8 Paper Layout

The rest of the paper is organized as follows:

- Chapter 2 discusses attitudes of system designs, and the system life-cycle.
- Chapter 3 discusses uncertainty and practical limits to predictability.
- Chapter 4 covers the triad *Vulnerable*, *Robustness* and *Antifragile*.
- Chapter 5 covers threats and vulnerabilities, and exposure/attack surface.
- Chapter 6 investigates possible defense strategies a system may utilize.
- Chapter 7 is an argument for a threat modeling mindset.
- Chapter 8 includes a summary and an overview.
- Chapter 9 is the conclusion, which tries to put the achievements of the paper in perspective.

2 Attitudes of System Designs

What should one normally keep in mind when designing systems? Or, for that matter, when one designs the security architecture of the system. In the following, we address some of the concerns, aspects and issues that one need to be aware of and take into account.

2.1 Process Goals

Systems are dynamical entities. Whatever high-level designs there are, the actual state of the system will tend to be constantly in flux. Indeed, the high-level designs will also be dynamical entities. That is, all concrete and specific system goals, including security architecture goals, will be subject to constant evolution. The concrete goals will therefore not be fixed and static, but rather be process goals.

2.2 Self-Evident and Obvious Aspects

A lot of things may be evident with hindsight. This also goes for systems design. Some of those things may indeed be predictable, but systems are complex and system designs are hard to do right.

In the “Black Swan” [2], the author points out that many Black Swan events seem predictable in hindsight. It is, of course, a fallacy. The same can be said about “self-evident” facts, tautologies, etc. One will often have no problem accepting these facts, but somehow it often does not lead to increased understanding or concrete actions to address the case.

2.2.1 Explicitness

As a corollary, if there are aspects that is assumed to be so obvious that it does not need to be explained, then one can be reasonably sure that there will be some that misunderstand. It is therefore important that crucial aspects are explicitly given. This is also in line with the explicitness principle in the paper “Prudent Engineering Practice for Cryptographic Protocols” [19]

2.2.2 Plan for Success

It ought to go without saying, but systems are inherently planned to be successful. One simply does not plan for a system to fail. Yet, there are implications to having a successful large-scale system that sometimes are not fully appreciated. One of those, ironically, is that a successful system will exists for so long that it will also experience large failures.

2.2.3 Success Means Long-Term

A successful system will be long-lived. Corollary, if the system isn’t successful, it will probably have a short life-time. In fact, successful systems routinely seem to be a lot more long-lived than anticipated by the designers. That is, systems design needs to flexible, extensible and able to accommodate whatever changes there will be.

2.2.4 Long-Term Means Change

If one accepts the premise that a system will be long-lived, then logically one should be prepared to accept that the system will experience substantial changes. Time is the ultimate stressor, and technology is fast moving and highly dynamic. A technology-based system will therefore have to be able to cater for change. One must assume that almost all parts of a system will be affected. Even when properly planned for, this will be a difficult process.

2.2.5 The Red Queen's Race

The so-called *Red Queen's Race* takes place in Lewis Carrol's "Through the Looking Glass". It puts emphasis on the need for change, and indeed for the necessity of change.

> "A slow sort of country!" said the Queen. "Now, here, you see, it takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!" – *Through the Looking Glass* [20].

To stay ahead on the technology curve, or at least to stay relevant, one needs to "run all you can do" just to keep current. This emphasises the need to embrace change on a system level. And, if one indeed wants to "get somewhere", one needs to be aggressive in embracing change.

2.2.6 Long-Term Means Dependencies

Systems do not operate in a vacuum, and there will be systems of systems. To paraphrase: *No system is an island.*

For an ICT system, there will be some obvious dependencies. One of those are a dependency on electric power. For a non-critical system, this dependency will have a limited impact on the system users. However, many systems starts out being non-critical, like the cellular systems in the early phases, and then become critical over time. So, one need to acknowledge that:

- The target system is, or will be, dependent on other systems.
- Other systems are, or will be, dependent on our target system.
- Large systems may turn into critical infrastructures.

If, like the cellular systems, the target system evolved into becoming a critical infrastructure, then there will profound societal dependencies. Obviously, one will want to reduce these dependencies, but that has proven hard to do.

2.2.7 There Will Be Risks

Whatever security measures one takes, there will necessarily be residual risks. Given this, one needs to learn to live with uncertainty and make plans accordingly. That is, risks must be managed.

2.2.8 There Will Be Failures

There are a huge number of possible causes for failures, and for any useful and long-lived system one will inevitably experience failures. This makes it necessary to "plan for failure". There needs to be contingency plans that addresses situations where the system fails.

Detection and response capabilities are perquisites, and it includes well-practiced full-scale recovery from backups.

2.2.9 Kill Your Darlings

The phase “kill your darlings” was reportedly a remark made by William Faulkner, and he intended it as advise for prospective authors. For an evolving system architecture, it can be thought of as advise to deprecate components, protocols and schemes ruthlessly.

A component that is found to have flaws and vulnerabilities may need to be deprecated even though one has yet to experience any actual exploits or incidents. In fact, even if the component is well-functioning and without any apparent flaw, it may still need to be deprecated. A technology will inevitably reach its end-of-life, and one should emphatically not be dependent on outdated technologies.

The grand implication is that every technology, every component, every protocol and every cryptographic algorithm inevitably will have a “best before” date. This also means that one must regularly schedule an audit of the applicability and validity of deploying the technology, component, etc.

2.2.10 Delusion and Self-Delusion

During the life-time of a system, there will be many decisions affecting the future trajectory of the system. For a security architecture, one must be able to deal with active and resourceful adversaries. These will seek to influence and weaken the system in any way they can. On a meta-level, this may extend to influence decisions about the future directions of the security architecture.

There will be an arms race in terms of capabilities to defend the system versus capabilities to attack the system. A large part of this will be a “shadow war”, where elements such as threat intelligence and deception will play a significant role. Perception of weaknesses and strengths comes into play, as well as whatever real capabilities there are. There may be “security theater” acting from the defence perspective [21]. Likewise, the adversary may be using various non-technical approaches, including social engineering, to fool the system, its users, its managers and its designers. Even if one imagines a perfect security scheme, it would matter little if one is able to circumvent the security by means of deceit and deception.

It is therefore important that decisions concerning the system be made based on facts. This means that one needs to have accurate situational awareness. The problem is of course that decision making can be influenced. People are susceptible to social engineering, coercion, etc. In the book “The Folly of Fools” [22], the author goes into the subject of deception and self-delusion in great detail. We shall not dive deeper in this fascinating subject, but suffice to say that awareness of these aspects is imperative in order to manage the long-term future of large-scale critical infrastructures.

2.2.11 The Weakest Link

Phising scams illustrates the fact that people will often be the weakest link in a system [23, 24]. If one can trick an authorized individual to carry out some task, then it may be difficult to counteract this with purely technical schemes. In this way, people are the weakest link. Modern system will increasingly employ automated decision system (machine learning, etc.), which will bring speed and efficiency to the decision making. Provided of course, that the automated system is not fooled. Loyal, vigilant and knowledgeable people can make security decisions that automated decision systems simply cannot do. In this way, people are invaluable.

An awareness of what actually constitutes the weakest links is a critical part of the overall security situational awareness. It should be coupled with cyber threat intelligence (CTI) capabilities.

2.2.12 On Universality, Adequacy and a Holistic Worldview

Security is very much a weakest link game. In order to find such a weakest link, one will need to see the system as a unity. That is, one needs to take a holistic approach. This will facilitate a focus on weak spots, missing elements and outright omissions. To have a front door made of reinforced hardened steel is all very well, but may come to naught if the backdoor is left open. One need to ensure adequate defenses that are *comprehensive and consistent* through the system.

We shall tend to interpret “adequate” in a strict way. One may retain a measure that isn’t adequate if its cost effective, but one cannot depend on such measure. If cost-effective, it *may* be useful as a defense-in-depth feature, but its just as likely that it should be scheduled for deprecation.

2.3 The 5D Lifecycle Phases

The following is the so-called 5D life-cycle of a large-scale system. We note that one may define a “devising” pre-phase too, and that there will be multiple iterations and cycles for the first 4 of the phases.

For a large-scale system, it should be no surprise that there will be parallel developments of different subsystems and components. This will make it hard to have consistent security levels over all components, and it is a kind of tension one needs to deal with. The implication of this is that the system will always be in a state of transition. Our goals of comprehensiveness and consistency will therefore be hard to meet, and must be seen as process goals.

1. Design phase
2. Development phase
3. Deployment phase
4. Delivery (of services; operational phase)
5. Decommissioning

2.4 The 5D Degrees of Freedom

There exists various degrees of freedom to make changes in a large-scale system, and these varies over the 5D life-cycle.

2.4.1 Design Phase Freedom

One has a great deal of freedom to change a system during this phase. The design group may even be able to make unilateral changes. There will inevitably be restrictions, but restrictions may actually be beneficial (Chapter 10 in [11]).

The freedom will be limited to designs within a specific version of the system. Past history will limit the freedom, as well as a host of other aspects ranging from legal consideration, economics at large and whether or not there are viable alternative system available. Requirements for fallback and backwards compatibility will impose additional restrictions.

2.4.2 Development Phase Freedom

For large systems, there will normally be many parties involved in the implementation of the system. In the development process, one will be limited in the changes one can make on the design. Design corrections will be possible, but these will need to be acknowledged and fixed by a design authority.

The basic design choices to be made are concerned with *how* to implement the various features. The individual parties may also have some freedom in *what* they choose to implement and not. That is, a system will often have many optional features and components, and it is common to only implement optional features that are actually requested.

2.4.3 Deployment Phase Freedom

Deployment is often conducted as a joint operation between the vendors and the system operator(s). There is a lot of logistics, as well as pressures like timeliness (time-to-market) and cost containment. The freedom to make changes is limited during this phase. That is, this is largely true for physical infrastructures and hardware related aspects.

For software-only functionality, the deployment may be fast and flexible. Technically, the updates can be almost as frequent as one wants.

On the physical side, there are aspects concerned with scale and supporting infrastructures (electricity, etc.). How many servers are needed, how many routers should there be, what kind of topology is to be used, etc.

2.4.4 Delivery (Operational) Phase Freedom

The operational phase is often characterized by one entity that has most of the authority. Many aspects concerning use of optional features and configuration of functionality may be unilaterally decided by that entity. For instance, Amazon decides most aspects of the www.amazon.com host(s) and T-Mobile decides for it’s networks. The freedom is limited in cases where there are external interaction.

When it comes to changing design features of the system, there is normally very little freedom at this level. Pure design errors and/or implementation error can of course be dealt with. However, the operating entity is not normally at liberty to deal with this, and must forward requests to the design authority and to the vendors. There may be substantial lead times for all but the most urgent cases. For security aspects, urgency will normally be decided by the Common Vulnerability Scoring System (CVSS) rating (or similar) and contractual requirements concerning lead times for a fix. This pertains to implementation flaws, but there may also be workarounds for design errors.

2.4.5 Decommissioning Phase Freedom

One might naively assume that the operating entity will have full unilateral decision authority concerning decommissioning of the system. Alas, this is not the case. There will be contractual and other legal aspects to be considered, and for critical infrastructures there will be a societal context to consider.

For instance, nowadays many mobile operators are considering to decommission their 2G and/or 3G services. To do this, the operators would have to ensure that spectrum licensing permits this, and to ensure that contracts with end-users are honoured. Finally, the operator will need permission from national regulatory authorities. Then, the operator will need to carefully examine all dependencies, etc., before proceeding with the process to actually decommissioning the system. Privacy and data control consideration will apply to equipment that has dealt with sensitive information.

2.5 Fallback and Backwards Compatibility Considered Harmful

During transition periods, and there will inevitably be a need to retain old functionality for a while. That is, one may have to live with different versions of core components and protocols. One will normally have a preferred version, but inevitably one may be forced to support multiple versions. This creates two large potential problems: *Fallback* and *Backward Compatibility*.

2.5.1 Fallback

Fallback is when a system entity attempts to use the newest version of a scheme, and it turns out that it is not supported by all interacting parties. For non-security purposes, this will likely lead to loss of some feature of the service. It is often the case that it is worse not to have any service than to have a somewhat limited and restricted service. Thus, one normally wants to support fallback to older versions of a service.

The problem, as seen from a security point of view, is when there is fallback to older versions that causes a downgrade in the provided security. The fallback mechanisms may then be exploited by an adversary to force a security downgrade. This is of course highly undesirable. For one, it means that one is still exposed to whatever weaknesses there were with the old security scheme. Secondly, one is forced to provide backwards compatible solutions in the system, leading to increased cost and complexity. This includes added security management complexity, which is never a good thing.

2.5.2 Backwards Compatibility

Backwards compatibility is related to fallback. For security procedures, it means deliberately permitting use of sub-optimal security. Needless to say, but this reduces the overall security level and it reduces the utility of the new schemes. In particular, backwards compatibility may be exploited by an adversary. Backwards compatibility is also costly to maintain. It is therefore harmful, and should be avoided whenever one can.

2.5.3 Containment

There should be explicit management decisions concerning permitting fallback and backwards compatibility. A threat analysis should be part of this process. There should also be monitoring of the usage of these solutions, and one should ensure that the usage patterns are acceptable. Extra logging may be necessary, as well as applicable use of cyber threat intelligence. Finally, there should be defined expiration dates for fallback and backwards compatibility solutions.

2.6 The Stoic Philosophical Attitude: "Keep Calm and Carry On"

The slogan "Keep Calm and Carry On" originates with a motivational poster produced by the British government in 1939 [25]. The backdrop was the preparation for World War II. The slogan clearly is a play on a belief in British stoicism and perceived British traits such as the "stiff upper lip", self-discipline, and remaining calm in stark adversity. When it comes to a philosophy of security architecture design, the Stoic approach inherent in "Keep Calm and Carry On" can serve as guiding principle. In a similar vein, the phase "Weather the Storm" also captures a mindset of endurance and perseverance. The Stoic attitude is useful for the whole of the 5D lifecycle.

3 Limits to Predictability

It is hard to truly understand the behaviour of complex systems like an ICT infrastructure system. It may appear tractable and it may seem that one can predict the system behaviour. However, it is folly to believe that one can fully understand such a system. That is, one may be able to predict "normal" behaviour, which is what you see when the system is operating within its basin of stability. The problem is of course that we are looking for predictable behaviour when the system is under stress and/or duress.

3.1 Non-linearity

When Edward Lorenz published his paper "Deterministic nonperiodic flow" [26] in 1963, it was not immediately embraced by the scientific community. However, today that paper is widely recognized as the starting point of *chaos theory*. It basically features a set of three seemingly quite simple equations (see Fig. 2), which one may even be inclined to think that one can easily solve. However, that would be a false assumption.

The Lorenz system is nonlinear, non-periodic, three-dimensional and deterministic (see https://en.wikipedia.org/wiki/Lorenz_system for more details). It is a system of ordinary differential equations. It is not important by itself in this paper, but serves to illustrate that seemingly simple systems can be amazingly complex. Such systems can be extremely sensitive to the initial configuration and can fluctuate wildly. Non-linear systems may also reach basins of stability, but you can not easily tell how close you are to leaving that basin.

Fig. 2 The Lorenz equations (an example question)

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x),$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y,$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z.$$

That is, such systems tend to be impossible to predict for all but the most trivial cases and the shortest time horizons.

3.2 Large-Scale Systems are Non-linear

To exhibit non-linear behaviour only means that the output of the system does not scale linearly with input. Large-scale ICT systems are non-linear and they inevitably have a huge number of more-or-less free variables. This means that prediction of dynamic aspects of such systems will often be intractable. Obviously, that is not to say that there is nothing one can know about the system. Far from it, but it does limit the certainty we can have.

3.3 Normal Accidents

In 1979, the Three Mile Island nuclear reactor experienced a tragic partial meltdown. It was the largest nuclear accident prior to Chernobyl. Subsequent to the Three Mile Island accident, there was a large investigation that sought to find the root causes. Charles Perrow gives an account of this in the book “Normal Accidents” [27]. The term “Normal accidents” is given to system accidents that are deemed inevitable in extremely complex systems. For long-lived systems, there will be cases where multiple innocuous failures can interact with each other. This may trigger non-linear feedback mechanisms. Each of the individual failures make be largely inconsequential and appear trivial, but may lead to chaotic and unpredictable cascading throughout the system. Perrow identified three conditions that make a system susceptible to these kinds of failure modes. These are:

- The system is complex
- The system is tightly coupled
- The system has catastrophic potential

If we modify the third point to include inter-system effects, Perrow’s failure modes will apply to any sufficiently large system. Non-linear behaviour is clearly at the heart of the model. We note that all real-world system will necessarily be complex, although not all complexity is necessary. We additionally note that by segmenting the system and by having fewer and only well-defined coupling, one will reduce the complexity and also reduce the impact of cascading effects.

3.4 The Certainty of Uncertainty

Our systems will be non-linear, and even very good designs cannot escape this. Thus, there will be uncertainty. This has ramifications, and one needs to plan accordingly.

3.5 Estimating Risk

In cybersecurity settings, one tend to define risk as the product of impact and probability. That is, $Risk = Impact \times\ Probability$. In a non-linear system, there will be a limit to how closely on can estimate impact and probability. That is, it may be perfectly possible to predict mundane occurrences with some precision. And, similarly, for frequent run-of-the-mill events, the risks can normally be estimated. However, due to the possibility of scalability and cascading effects, there will be a degree of uncertainty. This pertains both to probability of an incident and to the effects of an incident. Risk, as defined above, is therefore largely a useless concept. We shall much prefer to deal with *threats*, *vulnerabilities* and *exposure*. We shall return to those concepts in Sect. 5.

4 Vulnerable, Robust and Antifragile

We have already defined the archetype key concept in Sect. 1.4. Figure 1 illustrates the continuum from fragile, via robust, and to antifragile.

4.1 Fragile (Vulnerable)

A system is said to be fragile when small changes can lead to large (negative) consequences. A delicate wine glass may be said to be fragile, while an anvil is not. We typically want to measure the fragility in the face of adverse conditions and hostile actions.

Principle 1 A state of vulnerability is unacceptable.

Vulnerable components cannot be retained as-is. A system design action point (SDAP) must be made to either deprecate or refactor the vulnerable component. Then, as a principle, any vulnerability must be put on the to-do list. There will be another principle concerning vulnerabilities (Principle 13).

4.2 Robust (Resilient)

A system can be considered robust when it is largely unaffected by adverse conditions and hostile actions. The above mentioned anvil, is a good example of a robust entity. Robustness is necessary, but robustness, like the anvil, is a static concept. We do need to capture the dynamic reality of ICT systems, and in this sense robustness is insufficient. In a dynamic system, a state of robustness is not inherently stable and cannot be expected to last forever. However, it is still a necessary requirement that all components be robust.

Principle 2 Robustness is a prerequisite for a security architecture.

Only robust designs and components should be retained. If a component does not meet this criteria, an SDAP must be set to remedy the situation.

4.3 Antifragile

Antifragile can be considered to be the opposite of fragile. That is, instead of being susceptible to adverse conditions and hostile actions, an antifragile mechanisms will improve from an ordeal. There are numerous biological examples of antifragility, ranging from a muscle becoming stronger from heavy use to your immune system becoming better at fighting infections after having successfully fought an infection. What doesn't break the system, will improve it. There are of cause boundary conditions to the possible improvements. Too much abuse, and the system will break or become permanently damaged.

Principle 3 Only antifragile components are future-safe.

Antifragility builds on a foundation of robustness. Seek out robust components and develop these to become antifragile. The only way to go from robust towards antifragile is by embracing change and improve. In essence, to actively learn and be vigilant. This is a never-ending process.

Principle 4 Embrace change: Adopt, Adapt and Improve.

Actionable learning is a key aspect of antifragility. Continuous learning and improvement must be supported. This requires infrastructures, organizational support and management support for active learning. The learning must be based on and supported by data/information. Amongst others, this means that detection- and logging capabilities must be designed into the system.

Principle 5 Failures provide invaluable information.

To learn from failure is essential. Failures provide vital information about how the system actually works. Which means that one must capture and analyse failures. This requires technical support and knowhow. It also requires organizational structures and management support.

To learn from security failures in other systems is also important. Which means that one must actively seek out this kind of information.

Principle 6 Cyber threat intelligence (CTI) is an essential learning tool.

4.4 On Existential Risks and the Precautionary Principle

There is one type of risk that one simply cannot take. This, of course, is existential risks. Taleb is co-author of two papers about the precautionary principle, black swans and antifragility that highlights this [28, 29].

The basics of the precautionary principle is simple enough. Think twice, and do not take chances that may incur losses you cannot afford. It does not matter if the potential upside is large; if you cannot afford to loose, then you cannot afford to take the chance. Any action that may permit losses you cannot afford must *never* be taken. If one accepts these kinds of risks in a system, then it may actually jeopardise the future of the system. The only acceptable existential risk bet one may take is when it is even worse not to take the bet.

Principle 7 Precautionary principle: Existential risks must be avoided.

We also note that even if one can afford the losses, one is generally advised to avoid actions that have a huge downside. That is, one must at least be fully prepared to realize the losses.

Principle 8 Risks with a large or unknown downside must be avoided.

When things go wrong, it is sound advise cut losses at the earliest possible time. That is, we shall have a propensity for the “fail fast” philosophy. That is, if a component shows signs of being broken and/or vulnerable, accept that fact as soon as possible and take appropriate action. There may be a cost to this, but it may be a lot more expensive to procrastinate.

Principle 9 Fail fast. Cut losses when they are still affordable.

4.5 Sustainable Risks and Scalability

Relatively speaking, risks that have a well-defined and properly understood maximum penalty, are harmless. If one engages in a silly bet with a friend, where the loser will have to buy the other party a fine bottle of red win, then the maximum penalty should be acceptable. The bet is also symmetric, which means that the maximum positive outcome exactly matched the penalty.

Well-defined does not automatically mean fully understood. The story about the “chess board reward” illustrates this. In this story, the inventor of chess was to receive a reward from a king for inventing the chess game. Innocently, the inventor asks for one grain of wheat for the first square, then two grains for the second square, four grains for the third, etc. The king grants this, as he thinks it was a very modest request. However, the doubling does of course result in exponential growth and adds up to a total of $2^{64} - 1$ grains.

What one ideally wants is for the potential benefits to be unbounded and for the downside to have an acceptable and definitive bound.

Principle 10 Opportunistic risk taking is generally to be condoned.

This pertains to any aspect of improving and enhancing the security architecture. Of course, whether or not the security architecture is improved or enhanced by a change must be measured holistically, while keeping a long-term perspective. If the new change introduces inconsistencies or if it conflicts with other mechanisms, it may turn out not to be worth the effort. Then it will matter little if the change is beneficial in isolation. Also, it may be the case that a highly beneficial cheap fix will preempt better and more complete solutions. This will be tradeoffs between the here-and-now versus the future. One needs to ensure that longevity is a priority.

When it comes to systems and security architectures, a key property is *scalability*. One decidedly wants the security measures to scale effortlessly (friction free scaling), but modest linear scaling penalties may be acceptable. Likewise, one wants, at almost all costs, to avoid pathological scaling of attacks. That is, one must, as far as is practical, ensure that there is a growing cost to any type of attack on the system. One may think of this as adding friction to the attack. Ideally, the attack cost is at least linear, and ideally with a cost-benefit profile that makes attacks uneconomical.

Principle 11 Scalability must be handled with great care.

5 Threats, Vulnerabilities, Exposure and Attack Surfaces

There will always be threats towards a system. In order to convert a threat to an actual attack, the perpetrator (threat agent) will need to find a vulnerability to exploit. That requires an exposed vulnerability. Together the sum of all exposed vulnerabilities are known as the attack surface.

5.1 Threats

A threat is a statement of an intention to inflict damage or to carry out an adverse action. Threats are by means of threat actors (intruder/adversaies), and is usually directed towards an asset of some kind. As long as there are opposing interests, one is safe to assume that there will be threats (uttered or not). In order to take appropriate measure, one must know what kind of threats the system is likely to face. One must therefore conduct threat landscape investigations, and this must be a continual process.

Principle 12 Threat landscape awareness is a prerequisite.

5.2 Vulnerabilities

A threat may materialize into an attack. For this to happen, a threat agent would need to exploit some kind of vulnerability in order to successfully carry out an attack. This implies that there must exposed vulnerabilities.

Principle 13 Known vulnerabilities must fixed.

This is not to suggest that every vulnerability is equally important or need urgent attention. It simply means that all known vulnerabilities must be given an SDAP entry, and seen to in due course. Sometimes it may suffice to reduce the exposure to provide an effective stop-gap mitigation.

5.2.1 Incompleteness

There may be vulnerabilities that stem from omission and lack of protection.

5.2.2 Inconsistencies

There is a generic security principle called the "Principle of least surprise". This suggests that one should be systematic in the approach towards the whole of the 5D process to avoid inconsistencies that may be exploited.

5.2.3 Flaws and Errors

There are some vulnerabilities that simply are due to errors.

5.3 Exposure and Attack Surfaces

Exposure and Attack Surface are more or less the same thing. A perpetrator needs to interact with the system somehow, and all functionality that is visible to the perpetrator is exposed. The more functionality that is exposed, the larger the attack surface. Systems do of course need to expose their functionality to the users. Still, one is better off with a ‘Least Privilege” kind of approach (“minimal exposure”). This requires explicitly defining what functionality to expose. It must also be noted that different users will have different access rights, which implies that they have different views of the system.

The advise is simple: Be explicit about intended exposure. To be explicit about intended exposure does not guarantee that the attack surface is well-contained, but it will at least indicate that the problem has been considered.

Principle 14 System interfaces and exposure should be explicitly defined.

5.4 Cyber Kill Chain Considerations

Advanced persistent threat (APT) actors will tend to follow a certain set of steps to attack a system. These steps are called the “kill chain”. There are several kill chain models. A prominent kill chain model is the so called Lockheed Martin Kill Chain model from 2011 [30]. It identifies 7 steps that the APT intruder will take to carry out its nefarious actions. Later, other and more complete models has been suggested, including the Unified Kill Chain [31] and the MITRE ATT&CK framework [32]. Kill chain knowledge is no silver bullet, but kill chain awareness is nevertheless very important.

Principle 15 Awareness of the Cyber Kill Chain is necessary.

6 Defense Strategies

Defense is difficult. It needs to be comprehensive and consistent, while still being efficient. Band aid measures will tend to be neither efficient nor effective. To steer the efforts, a strategy is called for.

6.1 Clean Designs, Transparency and Explicitness

A sound design is normally also a more secure design. A clean and transparent design will contribute towards this goal. However, it is under most conditions also an unattainable goal. Still, this is not the kind of goal that one expects to reach, it more a guideline for direction.

One factor that contributes quite a lot is explicitness. Do not assume anything. If it is indeed important, then state it explicitly. This is sound advise for system designs at large, and even more so for security designs. In the paper "Prudent Engineering Practice for Cryptographic Protocols" [19], being explicit was elevated to a principle. We shall not hesitate to do the same.

Principle 16 Be explicit. Do not assume.

6.2 Fallback and Backwards Compatibility Revisited

We have already discussed the problems associated with fallback and backwards compatibility. In a dynamic system these problems cannot entirely be avoided, and so one needs to manage them.

Principle 17 Fallback and Backwards Compatibility must be managed.

6.3 Redundancy by Design

Efficiency may seem like a natural goal for a system architecture. And, while it is a natural goal, it should not be adopted without caution. What one obviously wants is for all individual component to be as efficient as possible.

However, one also wants robustness and this means that one needs spare capacity and capabilities for the rainy day. This translates into extra storage, extra processing and redundancy in all vital components. Redundancy must also be applied to security designs. One cannot allow security schemes to become a single point of failure without serious consideration of the drawbacks.

Principle 18 Single-point of failure must be avoided (and planned for).

One cannot always avoid single point of failure cases, but then one must at least identify those instances. Contingency plans should be made for handling such failures.

6.4 Segmentation and Coupling

To segment, or modularize, the system design is sound advise. Likewise, to reduce coupling to the minimum is a sensible goal. What we shall note is that while segmentation and loose coupling is necessary at a system level, the system access methods and access rights, etc. should still be seen as universal and consistent across the various segments.

Principle 19 Compartmentalize and de-couple whenever possible.

6.5 Defense-in-Depth

Defense-in-depth is an approach in which a series of defensive mechanisms are layered in order to protect valuable data and information. This may be according to segmentation boundaries, etc. If one mechanism fails, the perpetrator must very soon face another security mechanism. This will make an attack more complex to conduct, and it will incur a greater cost in the attack. This will effectively make the attack less scalable, and may even thwart the attack.

Principle 20 Defense-in-Depth is a requirement.

6.6 Strong Proactive Security Measures

Systems will typically have a host of security features as part of the baseline. This typically encompasses identification and authentication schemes, security protection for data in transit and data at rest, and security schemes for authorization and access protection. Typically, one will have guidelines about cryptographic strengths, passwords lengths, etc. These functionalities needs to be there, and be as consistent and comprehensive as possible.

Principle 21 Security coverage must be comprehensive and consistent.

They will also need to be upgradable and replaceable, which can pose a lot of challenges for security functionality.

Principle 22 All security functions must be upgradable and replaceable.

The key difficulty is that changes to security functionality needs to be absolutely incorruptible. We shall not go further into this. While these are hard problems, they are also run-of-the-mill aspects of security system design.

6.7 Capable Reactive Security Measures

These features are for the dynamic cases, and often for unexpected events. They typically encompass all detection and response capabilities, including the full gauntlet of recovery and incident investigations. We shall not go further into any of the specifics, save for a firm recommendations to have strong capabilities in detection, logging and analysis. It is an obvious requirement if one wants to learn from intrusion attempts.

Principle 23 There must be strong detection and response capabilities.

7 Threat Modeling Mindset

Murphy's law is a popular adage that states that "whatever can go wrong, will go wrong". A number of variants on the rule have been formulated, as have several corollaries. It catches the threat modeling mindset quite well.

7.1 Threat Modeling

Threat modeling is an activity normally associated with the design phase of a system. The threat modeling process may be a complex one, with multiple iterations and various degrees of impact on the system design. The process is roughly as outlined below:

1. *Define the system.* Define assets, actor types, system interfaces, data flows, system nodes and entities, possible intruders, etc. For reference purposes, everything identifiable aspect should be named and enumerated.
2. *Identify the threats—"What can go wrong".* The threats are subjective to the assets and key operational characteristics of the system. Every threat should be identified and enumerated.
3. *Propose counter-measures (prevention and mitigation).* This is the "What are you going to do about it" part. Every threat must be classified and handled according to severity. Many threats will just be noted.
4. *Validate the model and the outcome.* The first part pertains to the question of scope and depth of the model. The second part is about whether or not the threats are handled in an adequate way.

There may be multiple iterations of this cycle. The threat modeling approach should also be extended to the whole of the 5D life-cycle. In terms of mindset, we like to highlight these two aspects:

- What can go wrong?
- What are you going to do about it?

Principle 24 A Threat Modeling Mindset must apply to the full life-cycle.

8 Summary

We have presented a case for taking a philosophical stance to security architecture designs. Large scale systems, including our ICT systems, will often grow to become critical infrastructures. How we protect these systems therefore affects individuals and society, as well as the system itself. The security architecture design philosophy rules collected in this paper is an attempt to provide guidance in the design process.

8.1 Living with Uncertainty

If there is one thing that should be learnt from the Incerto, then this would probably be that one needs to learn to live with uncertainty. This means that one needs to acknowledge, both intellectually and by gut response, that one really cannot predict the future. It may appear to be perfectly clear, but somehow the human psyche prevents us from internalizing this in full. We design systems that are non-linear, tightly coupled and very complex, and yet we somehow still seem to believe that we can understand these systems. This is why one needs to explicitly state apparently self-evident principles.

8.2 Time Implies Change

The passage of time almost directly translate into change. And, invariably, any successful large-scale systems will be long lived. This literally translates into requirements to embrace change.

Principle 25 Plan for success and a long-term future.

8.3 The Design Principles

The following is a non-exhaustive and non-exclusive list of security architecture design principles. Adhering to these is not a requirement, but one ought to have carefully considered the principles before choosing not to adhere to them. One also needs to dwell on how to comply with the principles, as it may be difficult to operationalize the principles. This process is by itself beneficial.

1. A state of vulnerability is unacceptable.
2. Robustness is a prerequisite for a security architecture.
3. Only antifragile components are future-safe.
4. Embrace change: Adopt, Adapt and Improve.
5. Failures provide invaluable information.
6. Cyber threat intelligence (CTI) is an essential learning tool.
7. Precautionary principle: Existential risks must be avoided.
8. Risks with a large or unknown downside must be avoided.
9. Fail fast. Cut losses when they are still affordable.
10. Opportunistic risk taking is generally to be condoned.
11. Scalability must be handled with great care.
12. Threat landscape awareness is a prerequisite.
13. Known vulnerabilities must fixed.
14. System interfaces and exposure should be explicitly defined.
15. Awareness of the Cyber Kill Chain is necessary.
16. Be explicit. Do not assume.
17. Fallback and Backwards Compatibility must be managed.
18. Single-point of failure must be avoided (and planned for).
19. Compartmentalize and de-couple whenever possible.
20. Defense-in-Depth is a requirement.
21. Security coverage must be comprehensive and consistent.
22. All security functions must be upgradable and replaceable.
23. There must be strong detection and response capabilities.
24. A Threat Modeling Mindset must apply to the full life-cycle.
25. Plan for success and a long-term future.

9 Conclusions

The security of large systems is a societal concern. The same goes for the related concepts of safety and privacy, which we saw are dependent on strong security.

The security level one should strive for, and the arguments for those requirements, cannot be deduced by purely scientific methods. It is ultimately a value question, and it is therefore suitable to pose this as a philosophical question. Ultimately, that question is: What kind of society is it that we want? The right to feel, and be, secure and safe should be fundamental concerns.

The security architecture designs principles derived in this paper are, ironically, neither comprehensive nor fully consistent. They are also not strictly necessary. What is hoped for is that they will be useful, and that they will provide inspiration for others to extend and refine them.

It has been a conscious decision not to derive too many principles, as this would devaluate the individual principle. There is a balance to this, and to find the proper balance will require refinements and further work.

Acknowledgements Open Access funding provided by University Of South-Eastern Norway.

References

1. Taleb, N. N. (2001). *Fooled by randomness: The hidden role of chance in life and in the markets*. New York: Random House Publishing Group.
2. Taleb, N. N. (2007). *The black swan: The impact of the highly improbable*. New York: Random House Publishing Group.
3. Taleb, N. N. (2010). *The bed of procrustes: Philosophical and practical aphorisms*. New York: Random House.
4. Taleb, N. N. (2012). *Antifragile: Things that gain from disorder*. New York: Random House Incorporated.
5. Taleb, N. N. (2018). *Skin in the game: Hidden asymmetries in daily life*. New York: Random House.
6. Shostack, A. (2014). *Threat modeling: Designing for security* (1st ed.). Hoboken: Wiley Publishing.
7. Eikmeier, D. C. (2004). Center of gravity analysis. *Military Review*, *84*(4), 2–5.
8. Stevens, R., Votipka, D., Redmiles, E. M., Ahern, C., Sweeney, P., & Mazurek, M. L. (2018). The battle for New York: A case study of applied digital threat modeling at the enterprise level. In *27th {USENIX} Security Symposium ({USENIX} Security 18)* (pp. 621–637).
9. von Clausewitz, C. (2009). *On war*. Wildside Press.
10. Brooks, F. P, Jr. (2010). *The design of design: Essays from a computer scientist*. London: Pearson Education.
11. Denning, P. J., & Martell, C. H. (2015). *Great principles of computing*. Cambridge: MIT Press.
12. Saltzer, J. H., & Kaashoek, M. F. (2009). *Principles of computer system design: An introduction*. Los Altos: Morgan Kaufmann.
13. Buinevich, M., Fabrikantov, P., Stolyarova, E., Izrailov, K., & Vladyko, A. (2017). Software defined internet of things: Cyber antifragility and vulnerability forecast. In *2017 IEEE 11th international conference on application of information and communication technologies (AICT)* (pp. 1–5). IEEE.
14. Monperrus, M. (2017). Principles of antifragile software. In *Companion to the first international conference on the art, science and engineering of programming* (p. 32). ACM.
15. Køien, G. M. (2014). A best current practice for 3GPP-based cellular system security. In *Proceedings of global wireless summit 2014 (GWS'14)*. GWS.

16. Køien, G. M. (2015). Reflections on evolving large-scale security architectures. *International Journal on Advances in Security Volume*, *8*(1 & 2), 60–78.
17. Saltzer, J. H., & Schroeder, M. D. (1975). The protection of information in computer systems. *Proceedings of the IEEE*, *63*(9), 1278–1308.
18. Hole, J. K. (2016). *Anti-fragile ICT systems*. Berlin: Springer-Verlag GmbH.
19. Abadi, M., & Needham, R. (1996). Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, *22*(1), 6–15.
20. Carroll, L. (1917). *Through the looking glass: And what Alice found there*. Chicago: Rand McNally.
21. Schneier, B. (2009). Beyond security theater. https://www.schneier.com/blog/archives/2009/11/beyond_security.html.
22. Trivers, R. (2011). *The folly of fools: The logic of deceit and self-deception in human life*. New York: Basic Books.
23. McCoy, D., Park, Y., Shi, E., & Jakobsson, M. (2016). Identifying scams and trends. In M. Jakobsson (Ed.), *Understanding social engineering based scams* (pp. 7–19). Berlin: Springer.
24. Whitty, M. T. (2019). Who can spot an online romance scam? *Journal of Financial Crime*, *26*(2), 623–633.
25. Slocombe, R. (2014). *British Posters of the second world war*. Imperial War Museums.
26. Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, *20*(2), 130–141.
27. Perrow, C. (1999). *Normal accidents*. Princeton: Princeton University Press.
28. Taleb, N. N., Bar-Yam, Y., Douady, R., Norman, J., & Read, R. (2014a). *The precautionary principle: Fragility and black swans from policy actions*. Extreme Risk Initiative–NYU School of Engineering Working Paper Series.
29. Taleb, N. N., Read, R., Douady, R., Norman, J., & Bar-Yam, Y. (2014b). *The precautionary principle (with application to the genetic modification of organisms)*. arXiv preprint arXiv:1410.5787.
30. Hutchins, E. M., Cloppert, M. J., & Amin, R. M. (2011). Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, *1*(1), 80.
31. Pols, P. (2017). *The unified kill chain: Designing a unified kill chain for analyzing, comparing and defending against cyber attacks*. Ph.D. thesis, Cyber Security Academy (CSA).
32. Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., & Thomas, C. B. (2018). *MITRE ATT&CK: Design and philosophy*. MITRE Product MP 18–0944.

Geir M. Køien received his Ph.D. from Aalborg University in 2008. Before that he had worked for many years in industry, including LM Ericsson Norway and Telenor R&D. During these years he worked extensively with mobile systems and with security and privacy. He has also worked with the Norwegian Defence Research Establishment (FFI) and with Norwegian Communications Authority (NKom) on various security and communications related projects. Currently, he is a professor with the University of South-Eastern Norway (USN).