Chapter 7
Security and Privacy for the Internet of Things Communication in the SmartCity

Ralf C. Staudemeyer, Henrich C. Pöhls and Bruce W. Watson

What is a SmartCity? SmartCities face the problem of growth: with an ever growing world population living in large cities, governments and municipalities must sustain the city infrastructure and to provide public services. Therefore, the amount of information that is processed and stored to successfully, and still efficiently, manage a city's infrastructure (e.g., traffic, public transport, electricity) is growing also rapidly. To manage this SmartCities are deploying truly distributed and highly scalable information and communication (ICT) infrastructures connecting a conglomerate of smart devices or 'smart things'. These smart 'things' have self-configuring capabilities and use interoperable communication protocols to seamlessly integrate. Recently, the term Internet of Things (IoT) was coined to describe constrained systems that react via sensors to physical changes and are able to influence it via actuators. While ICT generally helps to mine that information, the IoT complements this with a direct link to sensors gathering needed data or taking immediate corrective action via actuators. Achieving some convergence of physical space and cyberspace offers manifold new possibilities for the SmartCities. Examples are traffic flow sensors measure congestion and environmental sensors measure air pollution. With this the IoT enables a fine grained monitoring and control of the city.

Why is a secure and private Internet of Things communication essential? Using the capabilities of the IoT to monitor and control the SmartCity implies numerous devices communicating data about the city and about its citizens, possibly impinging on basic privacy rights if not done correctly. The communicated data

R.C. Staudemeyer (✉) · H.C. Pöhls
Institute of IT-Security and Security Law, University of Passau, Passau, Germany
e-mail: rcs@sec.uni-passau.de

R.C. Staudemeyer · B.W. Watson
Information Science, Stellenbosch University, Stellenbosch, South Africa

V. Angelakis et al. (eds.), *Designing, Developing, and Facilitating Smart Cities*,
DOI 10.1007/978-3-319-44924-1_7

is used to make decisions that will affect many citizens and if not secured correctly, attackers (or other 'errors') could disrupt the operation of the SmartCity. This chapter provides a primer on general information security, its main goals, and the basic IoT security challenges in the SmartCity. Built upon the basic IT security goals of confidentiality, integrity, and availability, this chapter additionally addresses security and privacy problems in the communication that the SmartCity is facing. We especially focus on major issues related to private communication, as privacy is a key acceptance factor for an ICT-enabled SmartCity.

7.1 Information Security in the SmartCity

If we ask where could we apply security in the Internet of Things, and specifically in the SmartCities, the answer is: *everywhere* because a system fails first at its weakest link. Thus, we briefly introduce the overall view on security, many of these are adjacent to technical security mechanisms that we cover later in this chapter. For a deeper treatment of the fundamentals of computer security consult one of numerous excellent textbooks, e.g. [1, 2].

7.1.1 Security Management and Risk Assessment

For a start all the non-technical security functions need to be in place for the technical mechanisms to be not in vain, due to misconfiguration, misuse or easy circumvention. These organisational issues span a wide range, and can be as simple as the need to inform or train human users how to operate the system securely. Good organisational security also requires regular risk assessments to evaluate the effectiveness of implemented policies and controls and keep security updated. To effectively treat and manage risks the organisation needs a model for establishing, implementing, operating, monitoring, reviewing, maintaining and improving the protection of information assets. This is what ISO 27000 calls an Information Security Management System (ISMS) [3], and its basis is a risk assessment. In this comprehensive field, often the user becomes the weakest link [4]. Find a detailed discussion on information technology and risk management in [5].

7.1.2 Software and Operating System Security

Another viewpoint is about the design and implementation of secure software down to operating system level. It is of essential importance to understand that 'security is build in, not bolted on' [6]. This is challenging to achieve and many of the IT security problems are due to badly written software. Algorithms and especially cryptographic components must not only be well designed, they also need to be implemented to not introduce new weaknesses, like bad error handling, incorrect use of memory,

broken authentication, information leakage through side-channels, prone to denial-of-service, etc. Find a detailed discussion on software security in [6–8].

7.1.3 *Middleware and Trusted Systems*

In the SmartCity environment, we also need to consider security of the middleware abstraction layer. In the IoT, middleware abstracts from the underlying deployment of the actual sensors and actuators (the devices) and allows applications to easily get data and communicate with the virtual representations of the physical world. So, instead of asking the temperature of a specific sensor via its previously known network address, the application can simply ask the middleware for the temperature in the area of a specified physical location (e.g. street address). Think of the middleware as an intermediary, and indeed this non-direct accessibility to the IoT devices is called 'mediated device access' [9].

From a networking security point of view, the middleware component introduces at least one additional communication partner. In the simplistic case, depicted in Fig. 7.1, the middleware intercepts all messages and transforms the messages to fulfil the abstraction. In terms of network, each communication link between two individual systems is called a hop. In Sect. 7.3.5 we will discuss in more detail what is meant by the term end-to-end security. In short, end-to-end security protects the message on its complete flow. If the data from the (sensor) device and the application is not protected end-to-end then the intermediate components and systems, like the middleware must be trusted. Note, this is even necessary when all communication links in between are hop-to-hop protected by so-called secure channels.

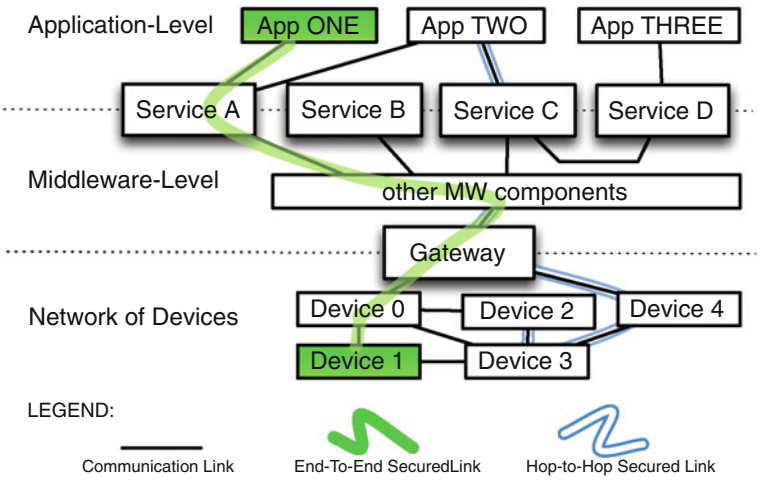

Fig. 7.1 Hop-to-hop security and end-to-end security protection in the Internet of Things levels

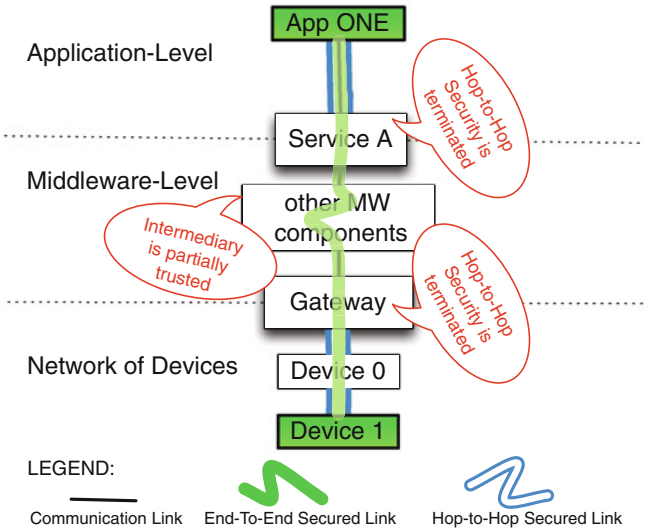

Fig. 7.2 Mixing hop-to-hop and end-to-end security for a communication between Device 1 and App ONE

In a more thought through design, the SmartCity framework shall allow important parts of a message to traverse intermediaries such as the middleware without having to trust them. In Fig. 7.2, some information (though not all) is protected end-to-end, while additional secure channels involving the middleware as one end of the communication are used to authenticate the applications on one side and the device on the other side.

Find a detailed survey on service-oriented middlewares in [10], as well as its role for end-to-end security in Sect. 7.3.5 [11].

7.1.4 On-Device Security and Trusted Hardware

We also need to discuss the general security and privacy mechanisms against a potential attacker gaining access to the device. Of course, *full protection* can only be achieved by complete physical isolation of the device, which is in a IoT context impractical. A compromise is to use *trusted hardware* which deeply embeds encryption as an intrinsic part of the processor (CPU) and memory systems—something only possible in the chip design phase, and therefore expensive. A further compromise is making the device *tamper resistant*—making it difficult or impossible to read memory contents or tap on-device data, such as the secret keys, without destroying the device. Over and above such hardware protection and tamper proofing, data on the device may be encrypted using software. In any case, the effectiveness of encryption depends on devices being able to keep keys secret, and thus on the effectiveness of measures taken to counter key extraction.

7.2 IoT Security Challenges

At first glance, IoT security challenges in the SmartCity domain (as for any IT system) could be considered already solved by applying good security hygiene, such as encrypting data end-to-end by default when communicating over the network, though they get trickier to solve in large-scale distributed systems.

We put emphasis on a selection of problems that we think are inherent to city-wide, large-scale deployments of the IoT for SmartCity applications.

Note, it is not that the underlying concept of the IoT is generally unsuitable for SmartCities, though it provides a vast number of new security challenges with indeterminate consequences. It is the large-scale deployment combined with the special need for protection of the data, which is of sensitive nature, that makes this a true challenge. Those peculiarities put special focus on security problems like key distribution and confidentiality protection on numerous devices. Foremost, it creates a significant larger attack surface due to different interfaces and new possibilities to interact. We have picked on some of them and highlight their importance by giving simplified examples from the SmartCity domain.

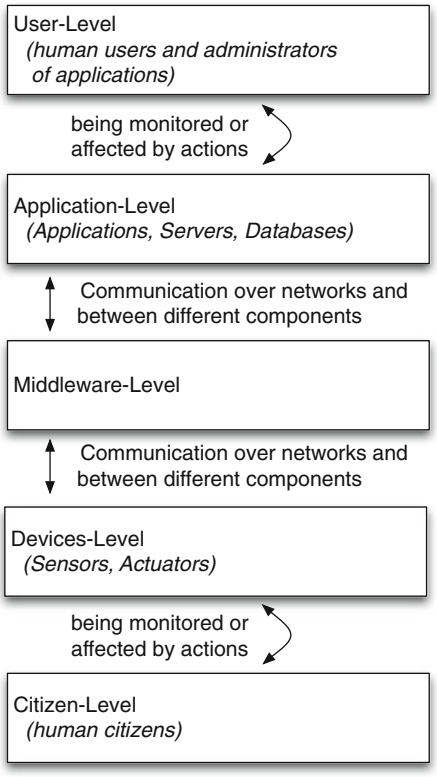

Fig. 7.3 Abstract levels of the IoT

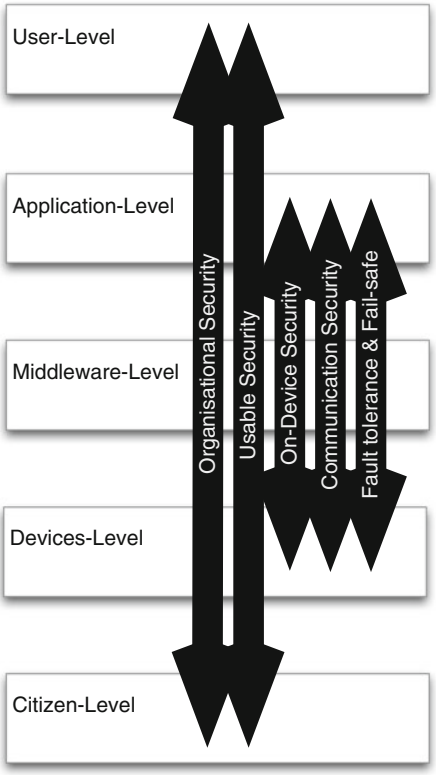

Fig. 7.4 Scope and influence of security choices

Figure 7.3 gives a very simplified layered architecture of the IoT. Via the devices, human users directly or indirectly interact in and with their physical surroundings. In this domain, users are citizens that participate in the SmartCity. At this level of abstraction the middleware is an abstraction layer. It operates, manages and unifies the plethora of IoT devices in order to allow their data and controls to be facilitated by applications. We will revisit this figure when we discuss where to place security controls, shown in Fig. 7.4.

A review of the architectural design for a secure Internet of Things communication with a focus on IP-based solutions is provided by [12]. From the legal perspective new security and privacy challenges are discussed in [13].

7.2.1 Fault-Tolerance and Fail-Safe Behaviour

Numerous SmartCity services are built assuming the ability of devices to communicate. Communication is between the devices themselves, with the middleware and finally with the application as well. Devices automatically connect to wireless

networks, join the infrastructure by registering with the middleware, and become part of services. Nevertheless, IoT devices and middleware are not oriented towards human interaction during operation. Conventional security mechanisms intended to slow down automated attacks, like CAPTCHAs[1] or rate limitation, can no longer be applied. The main reason is that the devices' communication partners are not a human, but a potentially unlimited number of other devices.

Even without dedicated attacks, the connectivity to network infrastructure might not be constant and of changing quality. Transmissions can get interrupted, messages can get reordered or the data transferred can get corrupted at any time. Moreover, the context of devices might change, abruptly or over longer periods of time, whereas these changes are most probably related to device mobility. While the reason for malfunctioning might influence the design of countermeasures, the communication problems or corrupted data usually affects the application regardless of their cause.

Systems need to cope with, survive and ideally actively handle variances and errors in communication, especially given that device-to-device interactions have no human in the loop: the IoT and also the SmartCity applications built on top of it have to cope autonomously. Fault-tolerant systems might overcome problems, like using other wireless frequencies or rerouting messages via alternative networks. Regardless, they need to be able to reach a consistent state if communication fails [14].

One fall-back solution is to provide partial service until the conditions are more favourable. Under all circumstances SmartCity applications must not fail with potentially catastrophic consequences. For example, traffic lights on crossings must still enforce that once one direction has a green-light the other shows red, regardless of what an attacked or malfunctioning system for bus-preference and traffic flow tries to convince them of. If all systems fail, traffic lights shall flash yellow lights in all directions.

Note that some definitions of IT security might consider this a safety feature. We believe that it is too important to not mention it, and argue that this is linked to the problem of not being able to guarantee 100 % availability.

7.2.2 Infrastructure Integration, Monitoring and Updates

IoT is likely to be integrated into the rest of the Internet, and protocols and security mechanisms used in the Internet will be preferred and are expected to be based on IPv6 while other protocols need to bridge to them. End-to-end security needs to survive and adapt to changes in infrastructure and underlying protocols.

Malicious attacks will happen from changing attack vectors, as current trends will continue in the future. Therefore we will see the IoT being exploited by old but also by fresh, novel attack patterns. The automated collaboration between devices and services will create unforeseen threats and damages.

[1] "Completely Automated Public Turing test to tell Computers and Humans Apart".

This creates the need to make sure that the 'health' state of the IoT infrastructure can be monitored consistently. This cannot be a top down monitoring approach as the participating devices and services are not known a priori, but it must be an emergent property. Only if the IoT down to device-level can detect problems, it can react to changes related to malicious activity like intrusions.

A related issue is to maintain software to remove bugs and to patch vulnerabilities. This requires a specification of how patches are released to IoT infrastructure and end devices. Communication of the new software from a trusted source must reach the devices over the network—again requiring fundamental communication security.

7.2.3 Efficient Cryptography and Key Management

Many classical IT security goals are addressable with sufficiently strong and securely implemented cryptographic mechanisms. However, the SmartCities physical space is being sensed or acted upon by numerous, but potentially constrained, devices. To manage the scarce resources on constrained devices we need cryptographic algorithms optimised to save time, memory, energy and as well physical space.

Moreover, all cryptographic mechanisms depend on some sort of key material. Here, symmetric cryptography has a drawback, it requires a secret key, pre-shared securely between devices. Once a secret key falls in the attacker's hands the added security by the cryptographic mechanisms depending on it is lost. The key will need to get revoked throughout the infrastructure; enforcing key generation and distribution afresh. Thus, capturing devices and attempting a secret key extraction are valuable attacks. To counter this, the IoT infrastructure should base its security upon public key cryptography.

Also public key cryptography has drawbacks. First, it is computationally significant more expensive than symmetric cryptography. Second, to work on a large scale we need a way to securely distribute and manage a key for each device. Distribution involves initial bootstrapping of trust and keys. Essential key management operations concern key update, revocation and recovery mechanisms.

7.2.4 Ownership and Secure Collaboration

Devices may perform operations for users, users may use devices to authenticate themselves. In this context, authentication is an open issue needing tailored security policies and mechanisms that control what and how data is created, accessed, processed and protected.

At first glance, this challenge cannot be solved without establishing a baseline for communication security: it depends on the ability to provide confidentiality of communications and authenticate the communication partner. For example, the policies that control the access to data need to be transported to the enforcement points over communication links, which themselves need to be protected against attackers.

Users need to build trust relationships with devices, while devices might know each other. Independent thereof, this requires that security policies interoperate on all components and that those are enforced. While these not directly look like communication security problems, the ability to execute these functions depends highly on the ability to provide confidentiality of communications and authenticate the communication partner.

7.2.5 Privacy, Trust and Data Minimisation

Devices are owned by companies, governments or by individuals providing a service. It is in first instance the devices' owners that get access to the information collected. Information that allows inference of actions of their users—information sensitive to leak. Devices should only collect and store information essential for the service and delete data once no longer required, while further processing of collected data (in unintended ways) should be avoided.

So far there is only a privacy assurance—on organisational level—which is not good enough due to a potential conflict of economic interests. There is high risk that user profile data is collected and users are tracked without need. Algorithms to prevent data collection and enforce reduction are still an open issue. And once security and privacy challenges can be technically addressed, their use still needs to be understandable and manageable for human users. Data minimisation is the foundation of the Privacy-by-Design principles presented in [15] and discussed in [16].

7.3 Information Security Goals Related to IoT Security Challenges

Computer systems can become corrupted in many aspects. We need to consider at least three when developing secure systems—the so-called *CIA triad*—also known as the three 'pillars' of information security, namely *confidentiality*, data *integrity* and *availability*. Confidentiality and integrity can be addressed using basic, well-known cryptographic mechanisms. Furthermore, there are numerous additional factors, such as the *triple A*, namely *authentication*, *authorisation* and *accountability*. Excellent all-around introductions into the basic concepts of computer security are provided by [1, 2].

Gaining access to a system is an *active attack*. This might involve stealing it physically or infiltrating it to gain logical control. Let us stress that also active attacks on the communication channel, like inserting messages or downgrading the communication channel, must be considered. By contrast, to eavesdrop or record communication is a *passive attack* and is detectable directly as it involves no alteration of data.

It is of great importance to acknowledge that information security is not limited to the protection of the data that is stored and transmitted as the content of a message. We also need to take into account that other sensitive information is revealed

by traffic data, so-called metadata. Due to the need to first achieve security for the messages' content and then for reasons of efficiency the handling of metadata is less widely implemented. Nevertheless it is a key consideration for the successfully establishment of privately usable services, in particular with regard to the SmartCity. We will cover private communication technologies in more detail in Sect. 7.4.

We will now briefly re-state current definitions and provide pointers to textbooks and standards. It will give you a quick glimpse, but do follow the pointers to find more background information. Additionally, we describe the most applicable security primitives/mechanisms and explain how they could be used to enhance the security and privacy in SmartCity applications. A detailed and entertaining introduction into cryptography is provided by [17]. Find with [18] a recommended and more recent mathematical textbook on cryptography.

7.3.1 Confidentiality

Confidentiality is the property that protects message content from passive attacks. It holds whenever someone not authorised fails to disclose confidential information from stored or transmitted data. The main idea is that an attacker will not learn information he is not authorised to know. On the communication side, the attacker is assumed to be able to eavesdrop on the communication. Therefore on unencrypted data at least the content of the message and the corresponding metadata are known; like the number, the size, the frequency of the messages on the communication link [19]. For data-at-rest the attacker is assumed to have physical access to a device.

7.3.1.1 Encryption to Protect Confidentiality

The cryptographic mechanism to protect confidentiality is encryption. Encryption transforms plaintext into ciphertext using an cryptographic algorithm; the reverse operation is called decryption. A cypher is a complementary pair of algorithms providing both cryptography operations. By introducing a key parameter the encryption function is extended to describe a transformation per individual key value. Classical encryption comes in two easily distinguished forms: symmetric and asymmetric.

Symmetric encryption is named for its use of either the same key for both encryption and decryption, or a key pair that can be easily computed with knowing at least one. For that reason this is specified as a one-key-scheme, since we treat them as effectively the same key. This symmetry yields cryptographic algorithms for encryption and decryption which are relatively simple, easy to implement, and are relatively secure, depending on the key length.

Further, actual symmetric encryption algorithms are often divided into two camps: stream and block cyphers. The *stream-cypher* encrypts a plaintext message bitwise using a keystream. In a perfect cypher the keystream is fully random, of same length like the message, and only used once. This is called a *one-time-pad* [18] and pro-

vides perfect confidentiality. The major drawback is that the keystream needs to be transported to the recipient via a secure channel. Therefore, implementations sacrifice unbreakability by generating the keystream using a pseudo-random number generator generated based off a shorter initial secret sequence (also referred to as seed). The cypher can be broken by predicting the secret sequence.

These days *block-cypher*s are more common, which were developed to counter shortcomings of stream cyphers. A block cypher divides the plaintext into 'blocks' of a specified size (128 Bit for AES) which are then processed resulting in an encrypted block.

Historically, the first openly available (digital) symmetric encryption scheme was the US Data-Encryption Standard (DES), which was progressively strengthened to become triple-DES (3DES). As for standards, 3DES was long superseded by the Advanced Encryption Standard (AES). Note, it is important that systems' cryptographic primitives can be updated to the most recent and thus currently considered secure algorithms. You shall not base the security of a system just on the current standard but have a process to update devices when new standards appear.

Symmetric encryption schemes compared to asymmetric ones are typically cheap to implement, use little memory and are with good performance, and are therefore suited to bulk encryption of large amounts of data. This implementation simplicity means that hardware implementations abound, requiring very little silicon real-estate, making them suitable for smart cards. Most modern CPUs include 'assist instructions' to further accelerate symmetric encryption.

The symmetry also implies that the encryption/decryption key must be kept as a secret between only the sender and the receiver, necessitating some secure key-exchange channel or mechanism. This makes it best suited for systems in which some other key-exchange exists, for example using public key encryption or a channel secured in some other way, such as physical handover; or where the encryption and decryption occur on the same system. The latter use case is particularly interesting given the speed of symmetric encryption for encrypting local files, messages prior to communication, and local memory.

As the name implies, *asymmetric encryption* requires two different keys: one each for encryption and decryption. The encryption key can be made public this mechanism is also called *public key encryption*. In detail the pair of keys differ depending on the particular encryption algorithm. Often, the key length parameter can be used to adapt to projected advances in computing power to prohibit attackers to compute the secret decryption key from the encryption key (in reasonable time). Of course, publicly distributable keys are very convenient, resolving the issue of how to exchange a secret encryption key in the first place.

The first openly available asymmetric encryption algorithm was Diffie–Hellman (DH) [20], followed soon by the highly popular Rivest, Shamir and Adleman (RSA) [21], and more recently the efficient Elliptic Curve Cryptography (ECC) [22, 23]. All three of these are based on number theoretic properties which are difficult to reverse, the key component in keeping the decryption key secret. For example, RSA uses keys based on the product of two very large prime numbers; obtaining those numbers

from the encryption key is thought to be the only way to obtain the corresponding decryption key, but factoring such a large composite number is extremely costly.

In comparison to symmetric encryption algorithms, asymmetric encryption is relatively expensive to implement. Operations such as exponentiation and multiplication are required in all of the prominent algorithms, needing many more instructions/clock cycles (and therefore power) than a symmetric scheme such as 3DES or AES. An hardware implementation is also costly in terms of silicon real-estate, and is rarely done. The convenience of public keys are not always an appropriate tradeoff for the costs involved, and most real-life systems actually use asymmetric encryption at the beginning of a communication session only to exchange a (secret) symmetric encryption key which is subsequently used for the remainder of the session.

Recent advances in Elliptic Curve Cryptography (ECC) have yielded asymmetric cryptography implementations to get significant more efficient than RSA [18, 24]. They work on a different mathematical assumption: a discrete logarithm problem on elliptic curves. The result are keys which are much shorter, while giving the same cryptographic strength, and allowing for more efficient implementations. Most importantly, strong ECC can be mathematically tailored to run in very constrained devices and very small silicon footprint, even in RFIDs [25–27]. Most public key encryption schemes play a further role in signatures.

7.3.1.2 Confidentiality in SmartCities

Today's citizens may assume that data flowing between devices at their private home or in their immediate vicinity is confidential. That it is inaccessible to any other parties without consent or warrant, and that at least not everyone is able to collect and process this kind of data at will. In a SmartCity environment, this rather clear separation between private and public environment blurs.

To give an example: The city of Amsterdam in the Netherlands supports more then 40 smart city projects ranging from smart parking to the development of home energy storage for integration into the smart grid [28]. One of these projects concerns the installation of smart energy metres with incentives provided to households who plan to actively save energy. Smart metres record energy consumption in households and report these in short intervals (e.g. 2s) to the provider. The benefit for the energy provider is that frequent reports allow demand management. Consumer can then benefit from lower rates during off-peak times, whereas this also depends on people actively changing their energy use. The downside is that these frequent measurements reveal detailed information on household activities, including presence, electrical devices in use, and even what content consumers watch on television.

Critical problem is reliable and secure communication, and as well trust towards the provider processing the information. Depending on location and environment of the smart metre communication might be via powerline, WiFi, cell phone network or the Internet. In contrast to energy monitors, smart metres permit two-way low-latency communication. These networks are more or less accessible and prone to eavesdropping and remote exploitation.

Here, encryption can harden unauthorized access to collected data and helps to protect information considered confidential. For example, symmetric encryption can be hardwired into network infrastructure on a hop-by-hop basis with reasonable effort and resources, providing a basic layer of protection between devices forwarding traffic. Asymmetric encryption can help to establish secure connections between end devices. In this use case between the smart metre and a trusted energy provider. If the energy provider is not trusted, then additional steps need to be taken to ensure that high frequency readings are difficult to associate with a specific consumer or metre [29–34].

Confidentiality protects against disclosure of information to unauthorized parties. This property helps to build secure communication channels to distribute software and updates (see Sect. 7.2.2), and to ensure secure collaboration (see Sect. 7.2.4). It is as well helpful for building long-term trust relationships (see Sect. 7.2.5). It is important to note that confidentiality is an essential security property required to build private storage and communication systems.

7.3.2 Integrity and Veracity

Integrity is violated whenever data or a system is modified in an unauthorized way, without being detectable. Modification can occur due to transmission errors or due to an active attack. However, for the correct and expected behaviour of an IT system the reason for a modification does not matter. Altered data must become reliably distinguishable from unchanged data. The protection mechanisms against malicious modifications however differ from those that detect random transmission errors. Thus, a Cyclic Redundancy Check value (CRC) can be used to protect against random small transmission errors [35], but cannot provide integrity against a malicious attack, as anyone—including the attacker—can compute a new CRC valid for the changed data.

Integrity protection mechanisms can be further distinguished into detective and preventive measures. The former fulfils its duty as it ‘detects the violation of internal consistency’ [36]. The preventive view of integrity protection (see classic definitions in [37]) requires mechanisms that prohibit unauthorized modifications upfront. In a nutshell, data integrity is ‘[t]he property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner’ [38].

As with every security goal, let us dig a bit deeper and understand the limits of integrity. *Veracity* is the property that the data in an IT system truthfully reflects the real-world aspect it makes a statement about. It is important to acknowledge that integrity notions and protection mechanisms are not concerned with the quality of the data, specifically from an application’s perspective. If information is incorrectly captured into the system data integrity protection will prohibit undetected modifications to this false data. Thus, this must be covered by mechanisms to protect the veracity, like increasing the tamper-resistance of the sensing device, making a consistency check by comparing the actual value to a prediction made, or comparing it

with assertions made by several witnesses assuming the adversary being unable to corrupt a sufficient number of assertions [39].

For a better understanding of the limitation of integrity with respect to veracity let us look at a traffic estimation example: Integrity protection allows to check at the receiving application that only verifiably unmodified traffic flow information gathered by known and authenticated sensors of a road segment will be used to do the traffic estimation. However, an obstructed or broken sensor starts with a wrong observation and thus records wrong data. This data does not truthfully reflect the correct information about the road segment's current traffic flow—it does not offer Veracity, as the data is wrong with respect to the physical world—but if the integrity protection is still working this data is still verifiably integrity protected.

With this limitation in mind, let us look at technical mechanisms to protect the integrity.

7.3.2.1 Digital Signatures Schemes (DSS) and Other Protected Checksums to Protect Integrity and Gain Entity Authentication

In order to protect the integrity of a message, a simple checksum like CRC, is not sufficient against a dedicated attack. The mechanism of choice is the protected checksum. It prevents an attacker to adjust the checksum to match changes made to the data [38].

We will briefly introduce the two most common forms: digital signatures and keyed hashes, also known as Message Authentication Codes (MAC). In general, a *cryptographic hash function* is a good mathematical hash H function that additionally offers the avalanche-, the one-way property and one of the two collision-free properties:

- Avalanche property: Given two inputs s' and s'' that differ only slightly the outputs $H(s')$ and $H(s'')$ must differ in each bit with a probability of $1/2$.
- One-way property: Given H and a hash result $h = H(s)$, it is computationally infeasible[2] to find s. Of course, given H and an input s, it must be relatively easy to compute the hash result $H(s)$.
- Weakly collision-free property: Given H and an input s, it is computationally infeasible to find a different input, s', such that $H(s) = H(s')$.
- Strongly collision-free property:
  Given H, it is computationally infeasible to find any pair of inputs s and s' such that $H(s) = H(s')$.

Using a hash function over a message gives something comparable to a fingerprint of the message. It has the main advantage that it maps an arbitrarily large message into a fixed length. Thus, if applied on the input first, all subsequent algorithms work on a fixed sized data object.

[2]Computationally infeasible means that it is possible, but that doing so would require a very long time and very powerful resources.

Digital signatures are a cryptographic tool to gain integrity protection for a message. A digital signature scheme (DSS) is based on asymmetric cryptography. Like in asymmetric encryption, two related keys are involved: the secret signature creation key and a related public verification key. The key algorithms of DSS implement the two functionalities: Sign and Verify. The sign algorithm takes the secret signing key and the message and generates a signature value. The verify algorithm takes the signature value, the message and the public verification key to obtain the result. If the result is positive this means that the signature is valid. In turn this yields two things: First, the document has not been altered according to an integrity policy and second, the signature value has been created involving the secret key corresponding to public key used for verification.

If a trusted link between an entity and the public verification key exists, a valid signature on a message implies that the message originated at the entity. This gives origin authentication and can be used to build entity authentication protocols. Those protocols need to be designed very carefully [40].

The most widely known algorithms are ECDSA (based on elliptic curves) and DSA (based on RSA). Both embody the use of aforementioned cryptographic hash functions, like SHA256 or SHA512. The message is first hashed with a secure, especially collision-free, hash algorithm, and then the resulting hash is used as input for the signature generation and validation algorithms. Due to the avalanche property the hash value of two messages will be already different if a single bit is changed. Hence, the integrity policy of those schemes (by choice of the hash algorithms) is to detect any subsequent change, even a single bit, as a violation of the signed data's integrity.

Additionally, the collision freeness properties do not allow an attacker, which by definition does not have access to the secret signing key, to take an existing signature on a message and then compute (or find) a colliding message that has the same hash. If it would be feasibly to find such a collision an attacker would be able to break the unforgeability property [41] as the attacker has then a valid signature on a message that was never signed by the signer, a valid forgery.

Keyed hashes allow detecting, based on the cryptographic hash function, if the input data object is changed. However, they forbid an attacker to simply re-calculate the hash value, as the hash value can be only correctly computed with the knowledge of the secret key. It can be built by several ways, one of which is to use any normal cryptographic hash (which is keyless) and concatenate the message with a shared secret key, not forgetting padding. An example is the HMAC algorithm [42].

In terms of the integrity protection this gives a symmetric pendant to a digital signature. However, due to the shared secret it does not offer the same level of authentication of origin, nor does it allow to prove the integrity to third parties. The reason for this missing authenticity of origin is that in a symmetric key-based integrity check the verifier needs to know the secret that was used to generate the integrity check value. Hence, in theory the verifier, as well as the 'signer', can generate a valid keyed hash for any message.

Comparing keyed hashes and digital signatures the main advantage of symmetric methods are speed and simplicity in the implementation, however the missing

authenticity protection the key management issues related to symmetric keys are a severe drawback. Digital signatures allow for a simpler key management as the public keys are not needed to be kept confidential. Thus, they allow presenting the public key alongside the signature and the message to any other entity which is able to verify the signature based only on public information. Only the entities (the importance here is that it can be as little as *one* single entity) that are in the possession of the secret signing key can produce a valid signature.

7.3.2.2 Integrity (and Related) in SmartCities

In SmartCities sensed data is gathered and used by algorithms to enable smarter decisions. Thus, the decisions are based on the data gathered, and bad quality data can lead to bad decisions. Imagine a SmartCity without integrity protected messages were every sensor value could have been tampered with. An faulty air pollution sensor in one city area might cause that area to be declared into a Zero Emission Zone (ZEZ). This leads to neighbourhood cars not being allowed to access the area, giving the citizens in the area far less noise, but congested roads in the neighbourhood due to through traffic. Would that not be an incentive to falsify pollution messages?

Likewise all control messages could be manipulated, so why not change the 'Access denied' message from the barrier control system into an 'Open barrier' message? The solution to that attack is that you detect any subsequent tampering with sensor data by applying integrity protection. Additionally, the origin of those integrity protected messages must be known to be a trusted source. Of course, a defect or even manipulated sensor could sent false readings. Then they would still be integrity protected, hence SmartCities need to deploy additional processing logic to detect such wrong readings and send repair personnel to investigate the actual sensor.

Integrity protection is a basic security functionality. It is needed for the authentication it aids with secure collaboration and with ownership (see Sect. 7.2.4). Finally, they can be used to authenticate software and support securing the distribution channel of software maintenance (see Sect. 7.2.2) against injected malicious code. Integrity protection detects erroneously or maliciously modified information and helps using them as input for further processing, and thus can be used as early detectors for a fail-safe behaviour (see Sect. 7.2.1). This is even more true if they are used in combination with veracity protection mechanisms.

7.3.3 Availability

Availability ensures timely and reliable access to devices and services. It goes with the assumption that a particular resource is not accessed in a non-legitimate manner, whether authorised or unauthorized. The availability property is violated if an attack succeeds by degrading a computer resource or rendering it unavailable. Typically,

this is done with so-called Denial-of-Service (DoS) attacks using up all the available resources and therefore increasing the response time. If the service is unavailable when accessed by an authorised entity, then the result is as bad as the service or device does not exist at all.

To ensure availability under all circumstances is a hard to solve problem.

7.3.3.1 Availability in SmartCities

When monitoring critical environment, sensor values and alert messages need to arrive timely, otherwise detection fails and no alarm is triggered. Critical values might be related to industrial contamination with potentially hazardous consequences for example in terms of air pollution, high radiation levels or decrease of water quality.

Take for example the city of Tarragona in Spain, which is next to chemical factories. Here the constant and reliable monitoring of air pollution for critical substances is vital to protect citizens. It is essential that potential air contamination can be detected before it reaches the closest households. In order to achieve this, the detection sensor devices deployed need to send their data to a monitoring server. The city council can then detect, or even preview critical events, with automated alarms and potentially react in a timely manner. If parts of the infrastructure, like the sensor, the network, or the monitoring server, gets unavailable, this detection will fail and the population can therefore not be warned in time.

To improve availability helps to address issues related to the challenge of fault-tolerance and fail-save behaviour (see Sect. 7.2.1).

7.3.4 Authentication, Authorisation, Accountability

The three goals are often grouped together and then referred to as 'AAA' or 'triple A'. Successful authentication and authorisation allows achieving accountability for a certain action by a certain entity.

Accountability enables the detection of actions (e.g. violations) or attempted actions to be traced to the potentially responsible entity [38]. In order to achieve this the entity needs to be first authenticated and then the request for access is subject to authorisation. Depending on the level with which entities can be differentiated by the authentication mechanism, they can be held accountable. Related to accountability is the notion of *non-repudiation*, which 'is a service to generate, collect, maintain, make available and validate evidence concerning a claimed event or action in order to resolve disputes about the occurrence or non occurrence of the event or action' [43].

Initially *authentication* is required. It is the 'process of verifying a claim that a system entity or system resource has a certain attribute value' [38]. This definition was carefully chosen because it highlights that it is concerned with checking a certain

attribute value. Authentication in general is not concerned with checking the identity of an entity. While the identity can be an attribute, this certain attribute should be seen as the feature of interest.

In order to restrict access you need to check if an entity is authorised to carry out a certain action: *Authorisation* is the 'process for granting approval to a system entity to access a system resource' [38].

7.3.4.1 SmartCities Authentication and Authorisation Problems

There are authentication problems on different layers of a system as complex as the SmartCity. As an example, assume the SmartCity detected a high risk of pollution and wants to restrict access to the inner city area. Assume that today only electric cars are allowed because the inner city is declared a Zero-Emission Zone (ZEZ). So the first question is: 'Who is the entity of the system that you want to authenticate?', which can be hard to answer in technical detail (see [40]). Do we want to authenticate a single car, the car's on-board-device, or its passengers? Here it becomes obvious that peer-entity authentication can happen on various layers.

For example, the city of Milan in Italy was reported in 2004 by the World Health Organisation as being one of Europe's most air polluted urban centres caused by very high downtown traffic volumes [44]. In 2008, the city introduced electronic road pricing to address traffic congestion, to promote sustainable mobility and public transport, and to decrease levels of smog. The restricted inner city so-called 'Area C' [45] and toll income is reinvested into sustainable energy projects. The area is accessible via gates monitored by video cameras equipped with automatic number plate recognition technology.

In this use case, the attribute value of the car is its license plate number. If cars can proof to the road toll system that they have a certain license plate, then this information can individually account individual cars for its road usage. Nevertheless, we acknowledge that the system design is very bad for privacy, since this data stored and correlated contributes to surveillance of citizens. There is no need for an application to have authentication based a unique identifier.

For example, to restrict inner city access to electric cars, it is sufficient if these are distinguishable from non-electronic ones. This means that the relevant attribute value is 'I am an electric car'. Of course this claim has to be proven, such that the system controlling access can be sure that it grants access for an electric car. The showing of attributes for access control should be done in a privacy preserving fashion using for example anonymous credentials [46].

Apart from having all these protocols, especially the ones that allow for privacy preserving authorisation, in all the IoT devices and in all the systems, each entity needs to have the credentials to prove its own claims to the system that asks for authorisation. This means that all participating systems need some form of keys and some form of trust that is associated with it. For example, would the city trust the car manufacturer's to vouch for the claim 'I am an electric car', or would the car need to be regularly inspected and be issued with a token issued by a state trusted institution.

However you do it, you are in need of sophisticated key management and key distribution. If not, some attacker might disguise his hybrid-car on demand as an electric car to cruise in the inner city.

The 'triple A' in general are important building blocks to tackle the challenges of secure collaboration (see Sect. 7.2.4) and to build trust relationships (see Sect. 7.2.5). They are also needed to authorise interconnectivity towards an existing infrastructure (see Sect. 7.2.2).

7.3.5 End-To-End versus Hop-To-Hop Protection

Before we move on, we want to highlight that all the above mentioned goals in general can be achieved between two parties. This means authentication could happen directly between the application in a smart phone talking with some individual barrier to access the inner city via some network infrastructure. But it could also happen between the smart phone application and an application server, and then again between the application server and the barrier control system, and then again between the control system and the individual barrier. The former is called end-to-end, while the latter is called hop-to-hop.

From a networking perspective each communication link between two individual systems is called a hop. Each system at the end of such a hop might require to know with whom they exchange data with, meaning they might need to authenticate, and be able to communicate in a secure manner. This means they might apply mechanisms to gain confidentiality, integrity and availability.

Depending on what system and what communication link you need the protection, the data transmitted in the communication system can be protected by different means. One option is to protect the transport link, the other is to protect every message separately. Both options have serious disadvantages suggesting a layered approach. See Fig. 7.1 for an example. To make the differences obvious, let us consider confidentiality protection by encryption. Hop-to-hop, or so-called link-level protection, encrypts data between neighbouring network nodes. The advantage is that encryption keys can be 'hardwired', which avoids issues with key-exchange. Major disadvantages are that all forwarding nodes have access to the unencrypted data and once the key leaked security is compromised.

In end-to-end security the confidentiality and the integrity protection is between the endpoints of the communication. As such, authentication (and finally the authorisation) can be performed between the endpoints as well. Note that for authorisation decisions it is important to understand what you want to authorise, in other words, what are the endpoints of your communication. For example, to logically authenticate a specific car you need to be sure that what you technically authenticate is affixed to that specific car, so it cannot be easily removed and placed into another car. Achieving end-to-end protection means that the need to trust the intermediate systems is removed. While this is preferable, it cannot always be achieved due to layered approaches and independent subsystems. Then a good system design

shall allow identifying these hop-to-hop or system-to-system protection and highlight those trusted systems for the risk assessment.

Implementing end-to-end security is essential for secure collaboration (see Sect. 7.2.4), building trust and to implement private communication (see Sect. 7.2.5).

7.4 Technical Communication Mechanisms Towards Privacy

The key concepts of information security are confidentiality, integrity and availability. Confidentiality and integrity, we learned that these can be addressed with end-to-end encryption and signatures using public key cryptography. We also learned that the use of encryption should be enabled as the default. But irrespective from the successful use of modern cryptographic techniques, attackers may eavesdrop communication and analyse network traffic.

The SmartCities wide area networks are impossible to physically secure against unauthorized access. In the IoT domain the local network access is predominantly wireless and is therefore prone to eavesdropping. To protect citizens from any kind of hidden loss of personal information when accessing public resources the communication also needs to be protected against traffic analysis. For example SmartCity applications do heavily depend on users location information to provide location-based services. Nevertheless there is no public interest to leave citizens traceable and facilitate continuous surveillance. The traditional information security goals are unable to protect location information, since these leak from metadata and can be extracted by traffic analysis. Here we need new means of protection.

Traffic analysis [19, 47] focuses solely on communication patterns and the extraction of information out of metadata, irrespectively from the use of content data encryption. In traffic analysis, network traffic is captured with the aim to gather information about the network, its devices, and its users. This discloses not only the communication partners and their frequency of communication, but also reveals information about the area of network alignment. Information extracted can be further processed and analysed by combining techniques from data mining and machine learning [48]. Further extracted information can be individual usage patterns, but as well identifiers used for future tracking.

Traffic analysis works without any knowledge of messages' contents, and is therefore applicable to encrypted messages. Large numbers of captured messages make traffic analysis more effective. It might reveal even more information if the traffic flow or the messages themselves can be modified or tampered with by other means.

Find an excellent introduction into privacy and data protection by design in [49], with more details covered in [50]. The Privacy-by-Design principles are presented in [15] and discussed in [16].

7.4.1 Network Properties for Private Communication

Even if the protection of user data is addressed by means of end-to-end encryption, we still need to look into information loss caused by leaking protocol metadata. This leakage can go up to the point, which may render end-to-end encryption obsolete. Traffic analysis attacks were successfully run on SSH [51] and Skype [52]. To limit success of these kind of attacks, at least the following properties [47] shall be taken into consideration by the network of devices:

- Coding—All messages with the same encoding can be traced.
- Size—Messages with the same size can be correlated.
- Timing—By observing the duration of a communication and considering average round-trip times between the communication partners patterns of network participation can be extracted.
- Counting—The number of messages exchanged between the communicating parties can be observed.
- Volume—Volume combines information gained from message size and count. The volume of data transmitted can be observed.
- Pattern—By observing communication activity, patterns of sending and receiving can be observed.

Finally, the observer can also perform a long-term intersection/disclosure analysis of the network by observing devices and the network for long time and reducing the set of possible communication paths and recipients by analysing online and offline periods. Characteristic usage patterns, such as an IoT device connecting every minute, may appear and can be used to further reduce the number of possible paths.

The following Table summarises the message properties and how they can be addressed (Table 7.1).

To counter traffic analysis we need to minimise any kind of information leakage. Therefore the network property we ideally aim for is unobservability, or at least anonymity. The *unobservability* property ensures that messages and random noise are indistinguishable from each another. In terms of network nodes it insures that their activity goes unnoticeable and that messages cannot be correlated. It is a very powerful property combining anonymity and *dummy traffic*. Anonymity breaks down into the unlinkability and unidentifiability property. The *unlinkability* property ensures that neither messages nor network nodes system can be correlated. Whereas *unidentifiability* ensures that these are indistinguishable, building a so-called anonymity set. Given that the anonymity set is always greater one, the system provides the *anonymity* property.

The word anonymity is derived from the Greek word ‘anonymia’, meaning nameless, and is interpreted to mean *the right not to be identified*. These aforementioned terms are defined in detail in [53]. In sum they are related as follows:

Unobservability = Anonymity + Dummy Traffic with
Anonymity = Unidentifiability + Unlinkability

Table 7.1 Protection against passive attacks

Attacks based on	Proposed solutions
Message coding	Change coding during transmission e.g. with k-nested encryption
Message timing	(1) batched forwarding of messages
	(2) random delay of messages (delaymin $\geq$ latencymax)
Message size	Use a predefined message size and padding small messages
Message counting	Receive and forward a standard number of messages and use dummy traffic
Communication volume	Protect message size and communication volume
Communication pattern	Continuous network participation
Message frequency	Use a standardized message exchange pattern
Brute force	No clear protection, dummy traffic helps
Long-term intersection	No clear protection, continuous connectivity and dummy traffic help

7.4.2 Proxies, VPNs and Dummy Traffic

A certain degree of anonymity can be achieved by using a proxy or a VPN, whereas both solutions route traffic via a relay. Proxies were initially developed for surfing the web, but SOCKs proxies can also forward TCP streams. Unfortunately, an observer with access to the traffic entering and leaving the proxy over extended periods of time, can reveal the communication relation. VPN networks on the other hand are slightly more robust, since they provide a layer of encryption to the incoming traffic. Therefore, incoming and outgoing traffic cannot be mapped easily.

Message frequencies and flow can still be analysed. The message flow between parties includes both the traffic volume and communication pattern. Communication partners have a unique distinguished behaviour that can be fingerprinted. An observer can perform a brute force analysis of the network by observing all possible paths of communication and generating a list of all possible recipients. Dummy traffic inserts additional messages with meaningless content into the network during times of less communication. The sender inserts them into the network in a form not distinguishable from real messages. This is done to pretend a constant high traffic that makes traffic analysis to require significantly more resources. Most efficient is the concept of dummy traffic used in conjunction with mix or ring networks with an implicit addressing scheme to prevent mapping.

7.4.3 Anonymity: Mix Networks and Onion Routing

Leakage of metadata can be further reduced by providing increased protection against network traffic analysis. This includes hiding network endpoints, timing

and location information. Traffic analysis is resisted by ensuring networks support the anonymity property as implemented by anonymising networks, most commonly using proxy chains. Anonymising proxy networks have started with the implementation of Chaums Mix in 1981 [54]. The system tunnels encrypted traffic through a number of low-latency proxies.

Initially, interest in this field was primarily theoretical, but in the last 30 years a lot of research in this field has looked at developing practical and usable systems for preserving anonymity [19, 55].

Onion Routing [56] was primarily developed to allow anonymous web browsing in close to real-time, but the concept is applicable to prevent traffic analysis in any network. Here the traffic is forwarded by multiply relays in ways that it is hard to tell which actually carry the traffic. This is achieved by using dummy traffic and nested point-to-point encryption. Once traffic leaves the onion routing network it can be observed, and therefore end-to-end encryption is needed which remains the responsibility of the end nodes. A well-known implementation of Onion Routing is 'The Onion Router' (TOR) [57].

Mix networks provide a unidirectional communication channel only, but provide stronger protection in comparison to onion routing. They additionally enforce a uniform message format and introduce extended delays. Mixing possesses the following attributes. They can

- hide the relationship between sender and receiver of a message,
- guarantee anonymity of the sender, the receiver, and both to all third parties and
- protect against revelation of signalisation relations, location updates, time of communication, and kind of service.

Mix networks and onion routing offer a high degree of end node controlled protection. The overhead of techniques based on Mixing is moderate. With Onion Routing there is a sufficiently mature solution available for SmartCity environments.

7.4.3.1 Anonymity in SmartCities

Smart metres provide automatic metre readings in high frequency for intervals defined by the electricity provider. In Sect. 7.3.1.2, we discussed the need that readings need to be confidential, since the frequent measurement reveal detailed information on household activities. Reference [29] discusses the need to further anonymise metre readings stored at the provider, so that it is hard to map the measurements with a particular metre or customer.

Nevertheless without protection, metre traffic can be mapped to communication partners, traced to a specific metre, and remains prone to traffic analysis and interception. Traffic can be modified, packets can be injected, and replayed. Triggered or natural changes in the communication pattern and volume can reveal as well sensitive household information, to the point of encryption getting obsolete. Classic traffic anonymisation techniques, like proxy chains and mixing, can support to address this thread with moderate costs.

7.4.4 Unobservability: Broadcast and DC-Networks

Broadcast and multicast based concepts offer full receiver anonymity. They protect the receiver of a message by sending it to all, or a set of, recipients of a network. An implicit destination address is utilised for enabling only the recipient to recognise the message. This can be done by public key encryption, which also provides authenticity, integrity and confidentiality. Here every recipient attempts to decrypt the message, whereas only the intended will succeed by using the correct private key.

With DC-Net a different mechanism was introduced [58]. The DC-Net is a round-based broadcast protocol where members can unobservable publish a one bit message per round. This is called 'superposed sending' and is very secure but prone to Denial-of-Service attacks. To address these protections against disrupting nodes were proposed [59, 60]. By 'superposed receiving' [60] DC-Net was extended to support anonymous receiving of messages. See [47] for a detailed description of the basic DC-Net-algorithm. Only few implementations exist [61, 62], probably due to DC-Nets sensitivity to disruption.

It is possible to categorise concepts into re-routeing-based and non re-routeing-based concepts [63]. Broadcast or multicast and DC-Net are the only non-re-routeing-based systems. These come expensive as the network grows since all members of the network need to at least send or receive one message. The situations changes dramatically whenever a shared medium becomes available [64], like access networks based on WirelessLAN and IPv6LowPAN.

We note that the overhead of broadcast-based solutions in todays switched networks comes with significant costs and, even worse, do not scale. Currently there are no mature solutions to obtain the unobservability property at reasonable costs. It is an open issue, which requires additional research.

7.4.4.1 Unobservability in SmartCities

One of the most sensitive data of citizens more recently collected and stored by governments in a SmartCity context is medical data. For example, the city of Johannesburg in South Africa aims to go paperless by 2016. This includes deploying a digital media health record to improve record keeping and as well patient care.[3] There is high risk that sensitive medical information will leak at some point and be processed in unintended ways, most probably not for the benefit of citizens. We conclude that the protection of stored data, location information and usage patterns might not be sufficient for medical data. Here, we may consider to protect as well the very existence of stored information and the access to it. Communication and storage systems that provide the unobservability property can further help to protect overly sensitive information, that cannot be stored by other means (e.g. paper).

[3] http://ehealthnews.co.za/joburg-invests-in-ehealth-to-benefit-patients.

7.5 Summary and Conclusion

There are many security challenges for Information and Communication Technologies (ICT) in the SmartCity domain. Some of them are well-known challenges in computer systems, like fault-tolerance, monitoring and software maintenance. Some are rooted from the networking domain, like traffic analysis, availability especially of wireless communication, and the plethora of problems when it comes to establishing secure communication links. These are magnified due to limited resources of devices in the Internet of Things, like cryptography must be especially efficient (see Sect. 7.2.3). Luckily, for a lot of these, technical and cryptographic solutions exist and are pushed towards general use and are currently pushed towards standardization.

This means that any active new development must reconsider if it has enabled all the latest security functionality, all people involved must stay vigilant and up-to-date with security enhancing tools. In general, no system as big as a SmartCity can be build securely without considering security from the design phase and always enable it as default in each and every subsystem. Still, it needs security professionals to gain the oversight to constantly judge the overall's system. This needs organisational strategies that also allow to update and react on technological changes. If some algorithm or system is known to be insecure it needs to be phased out quickly. Unfortunately, note that for secure system's engineering, the equation *secure subsystem* + *secure subsystem* = *secure system* does *not* hold. Adding new or combining existing systems is very likely introducing new security problems.

A good start to address these are the traditional information security goals. There are various models. We covered the 'three pillars' of information security, namely confidentiality, integrity and availability. We as well covered the 'triple A': authentication, authorisation and accountability. With the provided corresponding examples for the SmartCity domain you get a good starting point to discuss these topics with security experts. Table 7.2 provides an overview which challenges discussed in Sect. 7.2 can be addressed by which security and privacy functionality.

Finally, there are very hard technical challenges like identifying and managing the ownership over data. This is also known under the term of data provenance. Please consider that under EU privacy laws it is not the one who gathered or processed the data who legally has a say in what can be done with it; it is the data subject, the citizen, who the data is about or connected to. This will require a usable set of interfaces allowing interactions of ordinary citizens with data collecting and processing backends not yet technically foreseen.

Finally, technically supporting privacy of communications still remains to be adequately addressed. Privacy will come with a performance hit. The potential impact especially for achieving communication privacy is far more drastic than switching on encryption. And privacy is not achieved with just switching on encryption. We showed that the issues of information leakage can render even the best encryption by-passable given sufficient metadata and computing resources. Here, anonymous and unobservable communication helps to minimise leaking information. We outlined Proxies, Onion Routing and DC-Net as potential solutions, whereas the latter

Table 7.2 Mapping which challenges can be addressed by which security and privacy functionality

	Confidentiality	Integrity	Availability	'Triple A'	'End-to end'	Anonymity and unobservability
	Sect. 7.3.1	Sect. 7.3.2	Sect. 7.3.3	Sect. 7.3.4	Sect. 7.3.5	Sect. 7.4
Sect. 7.2.1 Fault-tolerance and fail-safe behaviour		✓	✓			
Sect. 7.2.2 Infrastructure integration, monitoring, updates	✓	✓		✓		
Sect. 7.2.4 Ownership and secure collaboration	✓	✓		✓	✓	
Sect. 7.2.5 Privacy, trust and data minimisation	✓			✓	✓	✓

lacks maturity and requires significant more research. Privacy is also expensive when just used for storing the data in a secure and privacy-preserving fashion, e.g. cryptographic well established methods like Shamir's secret splitting would require at least three replications [65].

In SmartCity environments, we need to deal with all the well-known security challenges. Most of them can be addressed with cryptography. The leakage of metadata can be addressed with anonymity systems.

Be prepared that technically adequate security always comes at a cost. However the hidden costs and dangers to the society as a whole that come from **any** insecurity or privacy-breach of an ICT-supported nerve system of a SmartCity are far greater. Once that the technical systems enabling a SmartCity become (or are just perceived as) an enemy, the general public will start fighting them; this will stall adoption and kill active participation of citizens in their SmartCity. In the end, any system has to offer incentives for collaboration; so you better build secure and privacy-preserving SmartCities that can gain and maintain their citizen's trust.

Acknowledgements H.C. Pöhls and R.C. Staudemeyer were supported by the European Unions 7th Framework Programme (FP7) under grant agreement n° 609094 (RERUM). H. C. Pöhls was also partly supported by the European Unions Horizon 2020 Programme under grant agreement n° 644962 (PRISMACLOUD).

References

1. Gollmann D (2011) Computer security, 3rd edn. John Wiley & Sons
2. Stallings W, Brown L (2014) Computer security: principles and practice, 3rd edn. Pearson Education

3. ISO/IEC (2014) ISO/IEC 27001: Information technology—Security techniques—Information security management systems—Overview and vocabulary. Technical report
4. Mitnick KD, Simon WL (2003) The art of deception: controlling the human element of security. John Wiley & Sons
5. Slay J, Koronios A (2005) Information technology, security and risk management. John Wiley & Sons, Australia Ltd
6. Paul M (2012) The 7 qualities of highly secure software. CRC Press
7. McGraw G (2006) Software security: building security, vol 1. Addison-Wesley
8. Viega J, McGraw G (2001) Building secure software: how to avoid security problems the right way. Addison Wesley
9. Tragos EZ, Pöhls HC, Staudemeyer RC, Slamanig D, Kapovits A, Suppan S, Fragkiadakis A, Baldini G, Neisse R, Langendörfer P, Dyka Z, Wittke C (2015) Securing the internet of things—security and privacy in a hyperconnected world. In: Vermesan O, Friess P (eds) Building the hyperconnected society- internet of things research and innovation value chains, ecosystems and markets. River Publishers Series of Communications. pp 189–219
10. Issarny V, Georgantas N, Hachem S, Zarras A, Vassiliadist P, Autili M, Gerosa MA, Hamida AB (2011) Service-oriented middleware for the future internet: state of the art and research directions. J Internet Serv Appl 2(1):23–45
11. Tragos EZ, Bernabe JB, Staudemeyer RC, Luis J, Ramos H, Fragkiadakis A, Skarmeta A, Nati M, Gluhak A (2016) Trusted IoT in the complex landscape of governance, security, privacy, availability and savety. In: Digitising the industry - internet of things connecting the physical, digital and virtual worlds. River Publishers Series of Communications. pp 210–239
12. Heer T, Garcia-Morchon O, Hummen R, Keoh SL, Kumar SS, Wehrle K (2011) Security challenges in the IP-based internet of things. Wireless Pers Commun 61(3):527–542
13. Weber RH (2010) Internet of things new security and privacy challenges. Comput Law Secur Rev 26(1):23–30
14. Lamport L, Shostak R, Pease M (1982) The Byzantine generals problem. ACM Trans Program Lang Syst 4(3):382–401
15. Cavoukian A (2009) Privacy by design ... take the challenge
16. Gürses S, Troncoso C, Diaz C (2011) Engineering privacy by design. Comput Priv Data Prot 14:25
17. Schneier B (1996) Applied cryptography: protocols, algorithms, and source code in C, 2nd edn. John Wiley & Sons, New York
18. Katz J, Lindell Y (2014) Introduction to modern cryptography, 2nd edn. Chapman & Hall/CRC
19. Danezis G, Clayton R (2007) Introducing traffic analysis. In: Digital privacy: theory, technologies, and practices, pp 1–24
20. Diffie W, Hellman ME, Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inf Theory 22(6):644–654
21. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126
22. Koblitz N (1987) Elliptic curve cryptosystems. Math Comput 48(177):203–203
23. Miller V (1986) Use of elliptic curves in cryptography. In: Proceedings of advances in cryptology (CRYPTO85). Springer, pp 417–426
24. Hankerson D, Menezes AJ, Vanstone S (2006) Guide to elliptic curve cryptography. Springer Science & Business Media
25. Bock H, Braun M, Dichtl M, Hess E, Heyszl J, Kargl W, Koroschetz H, Meyer B, Seuschek H (2008) A milestone towards RFID products offering asymmetric authentication based on elliptic curve cryptography. Invited talk at RFIDsec
26. Braun M, Hess E, Meyer B (2008) Using elliptic curves on RFID tags. Int J Comput Sci Netw Secur 2:1–9
27. Hein D, Wolkerstorfer J, Felber N (2009) ECC is ready for RFID a proof in silicon. In: Avanzi RM, Keliher L, Sica F (eds) Selected areas in cryptography. Lecture notes in computer science, vol 5381, pp 401–413
28. Municipality of Amsterdam. Amsterdam—SmartCity

29. Efthymiou C, Kalogridis G (2010) Smart grid privacy via anonymization of smart metering data. In: 1st IEEE international conference on smart grid communications, Oct 2010, pp 238–243
30. Jawurek M (2013) Privacy in smart grids. Ph.D. thesis, Friedrich-Alexander-University Erlangen-Nuernberg
31. Lahoti G, Mashima D, Chen W-P (2013) Customer-centric energy usage data management and sharing in smart grid systems. In: Proceedings of the first ACM workshop on smart energy grid security, SEGS '13. ACM, New York, NY, USA, pp 53–64
32. Danezis G, Jawurek M, Kerschbaum F (2011) Sok: privacy technologies for smart grids—a survey of options
33. Mashima D, Roy A (2014) Privacy preserving disclosure of authenticated energy usage data. In: 2014 IEEE international conference on smart grid communications (SmartGridComm), Nov 2014, pp 866–871
34. Pöhls, HC, Karwe M (2014) Redactable signatures to control the maximum noise for differential privacy in the smart grid. In: Cuellar J (ed) Proceedings of the 2nd workshop on smart grid security (SmartGridSec 2014). Lecture notes in computer science (LNCS), vol 8448. Springer International Publishing
35. Peterson W, Brown D (1961) Cyclic codes for error detection. Proc IRE 49(1):228–235
36. Michiels EF (1996) ISO/IEC 10181–6: 1996 Information technology—Open systems interconnection—Security frameworks for open systems: integrity framework. ISO Geneve, Switzerland
37. Clark DD, Wilson DR (1987) A comparison of commercial and military computer security policies. In: 1987 IEEE symposium on security and privacy. Los Alamitos, CA, USA, Apr 1987, pp 184–184
38. Shirey R (2007) RFC 4949–Internet Security Glossary
39. Gollmann D (2012) Veracity, plausibility, and reputation. In: Information security theory and practice. Security, privacy and trust in computing systems and ambient intelligent ecosystems, pp 20–28
40. Gollmann D (1996) What do we mean by entity authentication? In: Proceedings of 1996 IEEE symposium on security and privacy, pp 46–54
41. Goldwasser S, Micali S, Rivest RL (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308
42. Turner S, Chen L (2007) RFC 6151–updated security considerations for the MD5 message-digest and the HMAC-MD5 algorithms
43. ISO/IEC (1997) ISO/IEC 13888-1: Information technology—security techniques—non-repudiation, Part 1: General. ISO Geneve, Switzerland
44. World Health Organisation Europe (WHO/E) (2013) Health impact assessment of air pollution in the eight major italian cities, p 65
45. Municipality of Milan. Milan—Area C
46. Camenisch J, Dubovitskaya M, Haralambiev K, Kohlweiss M (2015) Composable and modular anonymous credentials: definitions and practical constructions. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 9453. Springer Verlag, pp 262–288
47. Raymond J-F (2001) Traffic analysis: protocols, attacks, design issues, and open problems. In: Designing privacy enhancing technologies, pp 10–29
48. Fawcett T, Provost F (1996) Combining data mining and machine learning for effective user profiling. Sci Technol 42:8–13
49. Danezis G, Domingo-Ferrer J, Hansen M, Hoepman J-H, Métayer DL, Tirtea R, Schiffner S, Agency (2014) Privacy and data protection by design—from policy to engineering. Technical report, European Union Agency for Network and Information Security, Dec 2014
50. Danezis G, Diaz C (2008) A survey of anonymous communication channels 1–61
51. Song DX, Wagner D, Tian X (2001) Timing analysis of keystrokes and timing attacks on SSH. In: 10th USENIX security symposium 28913:25

52. Dupasquier B, Burschka S, McLaughlin K, Sezer S (2010) Analysis of information leakage from encrypted Skype conversations. Int J Inf Secur 9(5):313–325 Jul
53. Pfitzmann A, Hansen M (2010) A terminology for talking about privacy by data minimization: anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. Technical report
54. Chaum DL (1981) Untraceable electronic mail, return addresses, and digital pseudonyms, Feb 1981
55. Ruiz-Martínez A (2012) A survey on solutions and main free tools for privacy enhancing web communications. J Netw Comput Appl 35(5):1473–1492
56. Goldschlag D, Reed M, Syverson P (1999) Onion routing. Commun ACM 42(2):39–41
57. Dingledine R, Mathewson N, Syverson P (2004) Tor: the second-generation onion router. In: Proceedings of the 13th USENIX security symposium, vol 13. USENIX Association, pp 303–320
58. Chaum D (1988) The dining cryptographers problem: unconditional sender and recipient untraceability. J Cryptology 1(1):65–75
59. Golle P, Juels A (2004) Dining cryptographers revisited. In: Proceedings of advances in cryptology (EUROCRYPT 2004), pp 456–473
60. Waidner M, Pfitzmann B (1990) The dining cryptographers in the disco: unconditional sender and recipient untraceability with computationally secure serviceability. In: Proceedings of the workshop on the theory and application of cryptographic techniques on advances in cryptology (EUROCRYPT '89) 89:690
61. Corrigan-Gibbs H, Ford B (2010) Dissent: accountable anonymous group messaging, p 12
62. Goel S, Robson M, Polte M, Sirer E (2003) Herbivore: a scalable and efficient protocol for anonymous communication. Technical report, Cornell University
63. Guan Y, Fu X, Bettati R, Zhao W (2002) An optimal strategy for anonymous communication protocols. In: Proceedings of the 22nd international conference on distributed computing systems 2002, pp 257–266
64. Stajano F, Anderson R (2000) The cocaine auction protocol: on the power of anonymous broadcast. Inf Hiding 1768:434–447
65. Shamir A (1979) How to share a secret. Commun ACM 22(11):612–613