

FAULT-TOLERANT, REAL-TIME, SECURE NOTIFICATION SYSTEM

[A. Railean](#)

Technical University of Moldova

Table of Contents

1 Introduction.....	6
2 Target system.....	8
3 Fundamental requirements.....	10
3.1 Interoperability.....	11
3.2 Security.....	17
3.3 Reliability, scalability and performance.....	22
4 Transport layer technologies.....	24
4.1 Twitter.....	24
4.2 SMS.....	26
4.3 Cell broadcast.....	29
4.4 Email.....	30
4.5 XMPP.....	33
4.6 AMQP.....	34
4.7 MQTT.....	34
4.8 Summary.....	35
4.9 Solutions compared.....	37
5 The importance of "instant".....	38
5.1 Hard "real time".....	39
5.2 Real time communications.....	41
6 System design.....	44
6.1 Components.....	44
6.2 Graphical representations.....	48
7 Ethical considerations.....	50
7.1 Security.....	50
7.2 Corruption.....	50
7.3 Vendor lock-in.....	51
7.4 Sharing and cooperative software evolution.....	51
7.5 Public perception.....	53
7.6 Conclusions.....	53
8 Conclusions.....	55
9 References.....	56
10 Appendix.....	59

Index of illustrations

Illustration 1.1: Energy savings and greenhouse gas emission cuts.....	5
Illustration 1.2: Known vulnerabilities in SCADA.....	6
Illustration 2.1: An earthquake notification system.....	7
Illustration 3.1: Google public alerts, displaying notifications received via CAP.....	13
Illustration 3.2: CAP alert on Google Maps.....	14
Illustration 3.3: Alert on Android.....	14
Illustration 4.1: USGS earthquake tweet.....	24
Illustration 4.2: Earthquake details via USGS.....	25
Illustration 4.3: The ubiquity of mobile phones in 2014.....	27
Illustration 5.1: Seismic wave propagation.....	38
Illustration 6.1: Suggested CA hierarchy.....	46
Illustration 6.2: Deployment diagram.....	48
Illustration 6.3: PKI infrastructure deployment diagram.....	49
Illustration 7.1: Evolution of cooperation.....	52
Illustration 10.1: Energy Star compliant buildings in Seattle, 2015.....	57
Illustration 10.2: Internet-facing control system devices.....	58
Illustration 10.3: Detailed view of an alert of Google Public Alerts.....	60
Illustration 10.4: QR code that contains a CSR.....	61

Index of Tables

Table 3.1: CAP features at a glance.....	16
Table 3.2: Basic concepts in security.....	17
Table 4.1: Messaging solutions compared.....	37
Table 5.1: Emergency procedures.....	39
Table 5.2: Real-time solutions compared.....	40
Table 5.3: Communication methods compared.....	43

Index of terms

ACID	Atomicity, consistency, integrity, durability
AMQP	Advanced message queue protocol
API	Application programming interface
APT	Advanced persistent threat
ASN.1	Abstract syntax notation one
BSD	Berkeley software distribution
CA	Certificate authority
CAP	Common alert protocol
CMP	Certificate management protocol
CRMF	Certificate request message format
DKIM	DomainKeys identifier mail
DTLS	Datagram transport layer security
EDXL	Emergency data exchange language
ETSI	European telecommunication standards institute
FLOSS	Free/libre open source software
FTP	File transfer protocol
GPL	GNU public license
HSM	Hardware security module
HTTP	Hypertext transfer protocol
IMAP	Internet message access protocol
ITU	International telecommunications union
LDAP	Lightweight directory access protocol
MQTT	Message queue telemetry transport
NIST	National institute of standards and technology

OASIS	Organization for the advancement of structured information
OCSF	Online certificate status protocol
PKCS	Public key cryptography standard
PKI	Public key infrastructure
POP	Post office protocol
QR	Quick response
RA	Registration authority
RFC	Request for comments
RTOS	Real-time operating system
S/MIME	Secure/multipurpose Internet mail extensions
SASL	Simple authentication and security layer
SCADA	Supervisory control and data acquisition
SMS	Short message system
SMTP	Simple mail transfer protocol
SPF	Sender policy framework
SPKAC	Signed public key and challenge
STOMP	Streaming text-oriented message protocol
TCP	Transmission control protocol
TLS	Transport layer security
URL	Uniform resource locator
VPN	Virtual private network
XML	Extended markup language
XMENC	XML encryption
XMLSIG	XML signature
XMPP	Extensible messaging and presence protocol

1 Introduction

Telemetry plays an important role in modern industry, being applied in a very wide set of contexts, ranging from agriculture to space exploration. It is a key ingredient in increasing quality of life because it enables us to benchmark our attempts to optimize energy efficiency and reduce waste. For telemetric data to be useful, it has to be available at the right time, giving an opportunity to act swiftly, while the information is still relevant, thus there is a powerful driving force to measure more parameters and do it with a greater precision. According to Energy Star, in 2013 program benefits have doubled¹ since 2008, reducing utility bills by \$30 billion, preventing the emission of 277 million tons of greenhouse gases.

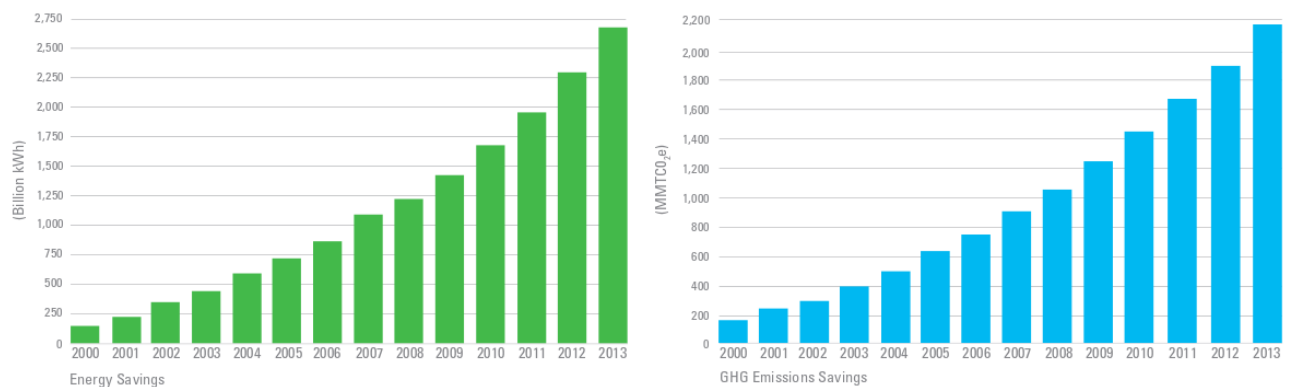


Illustration 1.1: Energy savings and greenhouse gas emission cuts

The ubiquity of the Internet solved the communications problem above, but created a new one in the process. As more telemetric data are available via initiatives² designed to create smart buildings³ and cities, the greater is the damage that can be caused by abuse and manipulation.

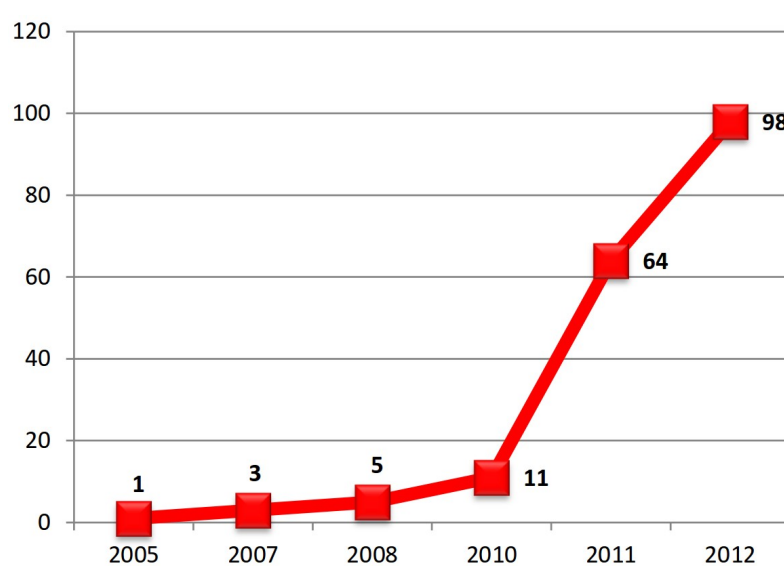


Illustration 1.2: Known vulnerabilities in SCADA

The next challenge is to ensure that security is baked into telemetry systems from the very beginning, instead of being bolted on top at the final stages. This becomes doubly important in the case of systems that provide a remote or automated control facility.

This paper reviews the telemetry communications landscape and focuses on a specific component in the grand scheme of things – the secure transmission of notifications from systems to people, as well as to other systems. The result of this work is an infrastructure design that meets the demands of a modern civilization, addressing present day concerns and anticipating future needs.

2 Target system

The approach developed in this publication is suitable for a broad range of infrastructures, examples shall refer to a specific scenario where a secure, reliable, real-time notification about an imminent earthquake must be received. The system consists of:

- a seismometer
- a telemetric box connected to the seismometer
- a notification broadcast server
- a number of redundant communication channels between the box and the notification broadcast server
- an arbitrary number of subscribers, i.e. systems that will to receive the notification

The seismometer is located in the area where earthquake epicenters of an area are located. When a calamity occurs, the shockwave propagates through the lithosphere and eventually reaches other settlements (see chapter 5 for more details). While the wave is in transit, targets that will be hit soon can anticipate that and prepare accordingly – by shutting down critical infrastructure, slowing down fast-moving transport, turning on a siren, and so on.

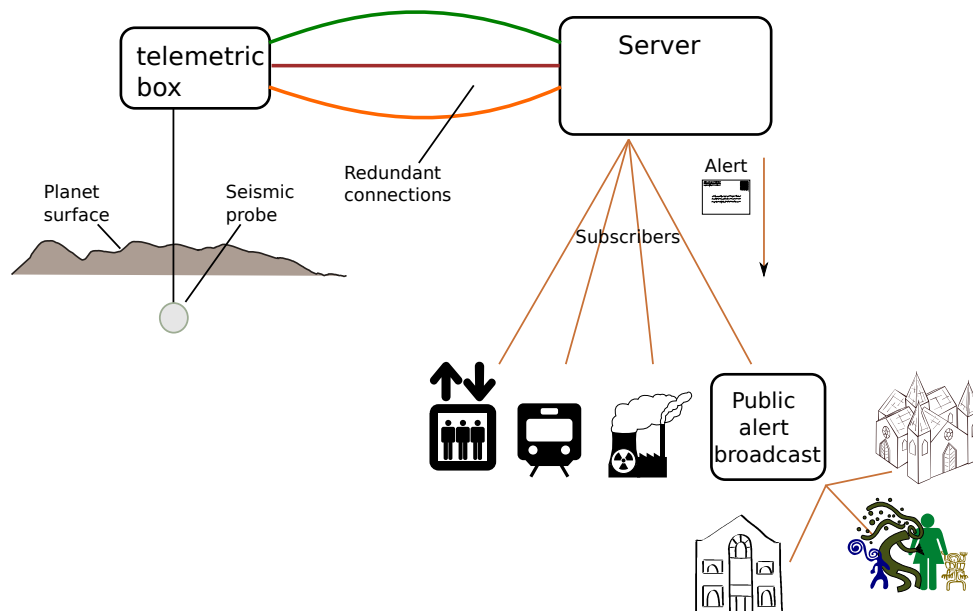


Illustration 2.1: An earthquake notification system

At the first glance, this is a simple mechanism – all it has to do is send one bit when an earthquake occurs. However, we have to take into account a number of ways in which things can go wrong:

- **Fake alerts** will cause unnecessary efforts – certain systems will be shutdown or slowed down, causing temporary disruptions in service. Such unauthentic alerts, if frequent, will desensitize the subscribers as in “the boy who cried ‘wolf!’ story”; this, in turn, will cause them to ignore genuine alerts.
- **Unsent or missed alerts** imply that the system is not reliable, because it failed to perform its primary function when it was necessary.
- **Subsequent shocks** that occur after the initial earthquake can hit people and infrastructure when they think that the worst is already in the past.

Thus the original problem is extended: send a single bit when an earthquake occurs and ensure that the bit arrives *exactly once* and constantly monitor the state of the entire system in order to ensure it is *always ready* to send that single bit.

Reliability must also extend to aftershocks that sometimes happen after the first shockwave occurs. In other words, the system must be ready to send *subsequent notifications*, instead of operating in an “apres le premier bit – le deluge” mode.

3 Fundamental requirements

For a notification system to be successful in terms of utility and practicality, it has to meet some basic needs. They are briefly described in this section, the most important ones will be reviewed in detail in subsequent chapters.

- **Performance**

- **Speed** – notifications have to be delivered swiftly, giving the recipient the power to leverage the information while it is still relevant.
- **Scalability** – a system has to be able to handle massive quantities of recipients spread over a wide geographical area.
- **Reliability and graceful degradation** – be sturdy enough to continue providing the service despite damage taken; for example, in the case of an earthquake the system has to stay alive to notify about subsequent tremors, not just the first one.

- **Security**

- **Authenticity** – guarantees that data comes from a genuine source have to be in place, to thwart spoofing attacks.
- **Integrity** – formal proof of the fact that the message was not changed while in transit.
- **Confidentiality** – transmitted data must not be readable to a third party.
- **Interoperability** – facilitate integration with other systems via use of common protocols or support of various formats.
- **Completeness** – the mechanism must strive towards full coverage of uses cases for a notification system, addressing needs without requiring new protocol elements.
- **Quality of service** – guarantee the delivery of messages within a predetermined time interval or a specific number of times when necessary.
- **Logging and audit** – preserve a trace of transmitted notifications and provide a way to confirm the integrity and authenticity of the log.

3.1 Interoperability

A technology that facilitates interoperability is clearly documented, such that a third party can read the specifications and build a system that smoothly interacts with said technology. Several initiatives to address this problem already exist, therefore it would be wise to leverage them.

3.1.1 CAP – common alerting protocol

CAP is an open standard that has its roots in 2001, maintained by OASIS, currently at version 1.2⁴. It is the product of an international working group comprising over 130 emergency managers and technology experts. It also became an ITU-T recommendation in 2007, known as X.1303.

The protocol is a member of the EDXL family of standards governed by OASIS, it is an XML-based data structure that facilitates information interchange between different systems. The format accommodates metadata such as:

- categories of notifications (geographical, meteorological, fire, health, etc)
- response types (e.g. shelter, evacuate, prepare or avoid)
- severity ratings: extreme, severe, moderate, minor, unknown
- certainty levels: observed, likely, possible, unlikely, unknown
- alert unique identifier
- alert onset and expiry timestamps
- sender information
- optional human-readable instructions
- optional set of key-value tuples
- optional information about the affected area

Security

Security-wise, it allows the use of XML digital signatures⁵, but it also mandates the processing of non-authenticated alerts:

Processors MUST NOT reject a CAP Alert Message containing such a signature simply because they are not capable of verifying it; they MUST continue processing and SHOULD inform the user of their failure to validate the signature.

While the current version of CAP does not provide any encryption mechanisms of its own, the earlier specification suggested the use of XMLENC:

The alert element of a CAP Alert Message MAY be encrypted, using the mechanisms described by XML Encryption Syntax and Processing [XMLENC]. Other XML encryption mechanisms MUST NOT be used in CAP Alert Messages; however, transport-layer encryption mechanisms may be used independently of this requirement.

The latest version is basically a “relaxed” formulation of the statement above, encouraging the use of XML security mechanisms, but not advocating one specifically:

Because CAP is an XML-based format, existing XML security mechanisms can be used to secure and authenticate its content. While these mechanisms are available to secure CAP Alert Messages, they should not be used indiscriminately.

In conclusion, encryption can be achieved at the transport layer, by tunneling the alert through a connection that is encrypted by the underlying mechanism (e.g. VPN, SSH connection, TLS socket, etc).

Encodings

The XML schema of a CAP message is well-defined, a validator⁶ is also available. XML alerts can be quite lengthy in size and contain redundant data, CAP allows a more compact encoding of messages in ASN.1⁷.

Remarks

Due to the fact that the data structure is either a verbose XML file or a binary ASN1, it cannot be used “as is” with certain transmission methods such as SMS⁸. While it is not a limitation of CAP itself (it was not within the protocol's scope to produce alerts that can be delivered to people directly without any changes, or alerts that fit into an SMS), it implies that for use with some systems it is necessary to write “glue” software that will transform the alert into a format suitable for consumption

by other receivers.

After an analysis of the official protocol reference from OASIS and the recommendation from ITU-T, we conclude that X.1303 was branched from CAP v1.1, because it still contains the reference to XMLENC, which was removed in CAP v1.2.

Adoption

The standard originated in North America and was originally used in the USA, then covering Canada and Australia. Nowadays, there are also deployments⁹ in Colombia, Japan, Taiwan, Indonesia, Mexico and the Philippines. The National Italian Fire Corps has officially adopted CAP as of June 2011¹⁰.

It is embraced by the Google Public Alerts service and is integrated with products such as Maps and Now on Android, here is an actual view of google.org/publicalerts, where the adoption geography becomes clear.

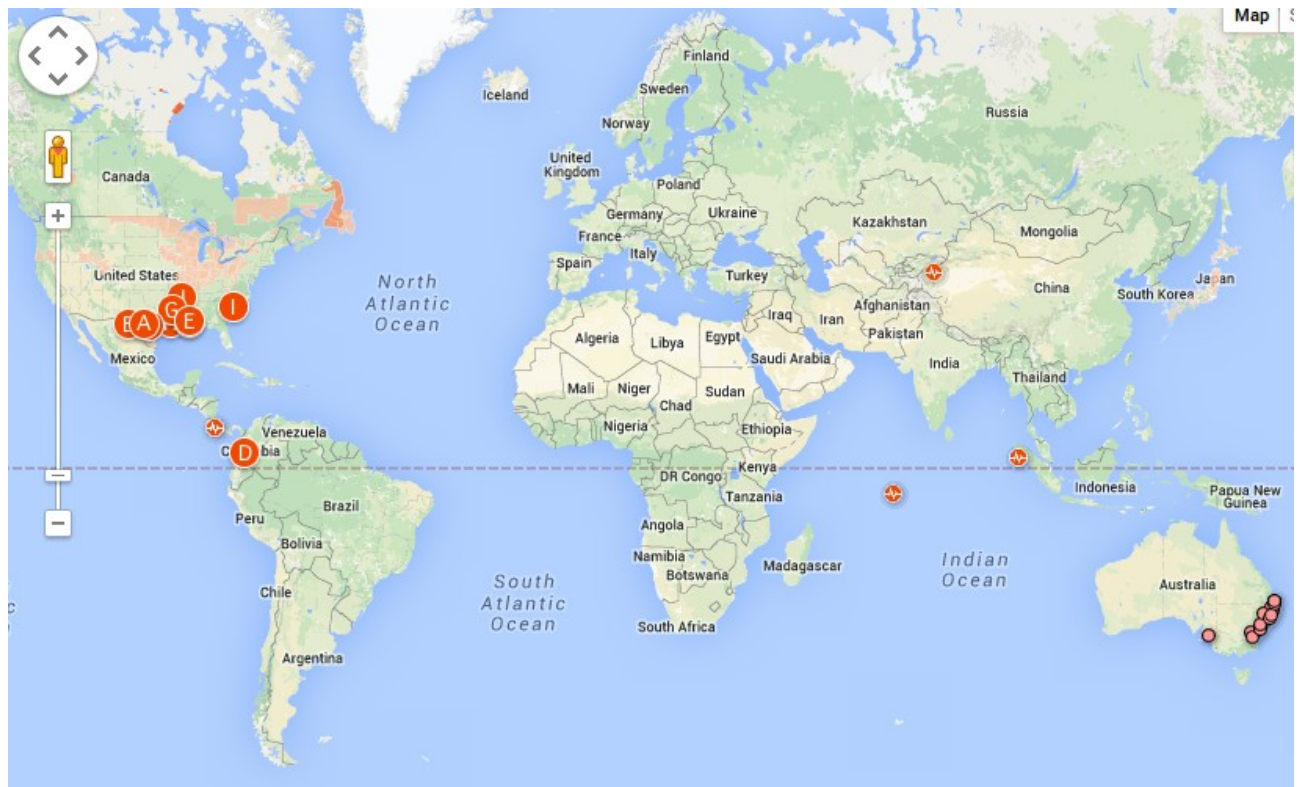


Illustration 3.1: Google public alerts, displaying notifications received via CAP

Notable implementations

Google Public Alerts

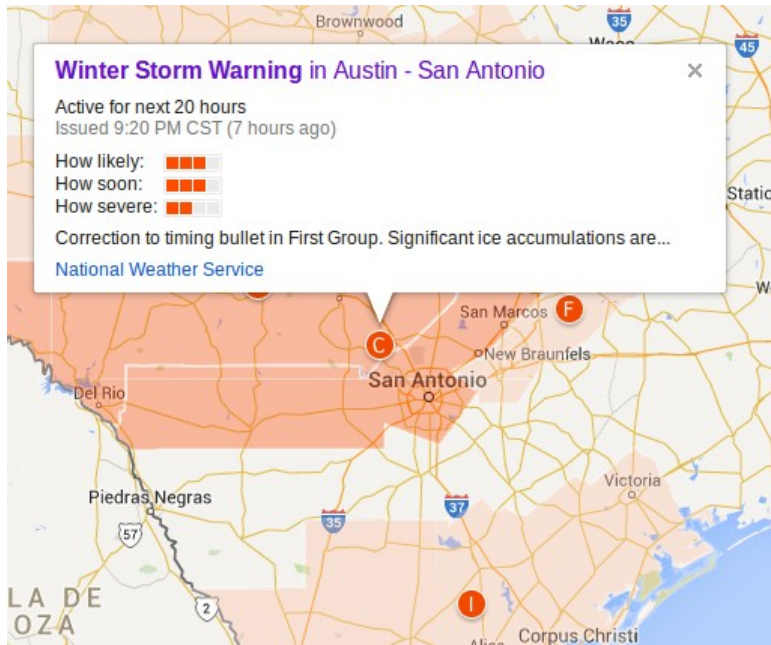


Illustration 3.2: CAP alert on Google Maps

CAP is supported by Google and blends in with the rest of the company's products, this is an excellent shortcut, as it leverages existing technologies and visualizes the data of an alert in a very readable way, taking into account all the metadata that the protocol offers and combining it with search functionality that people are used to.

Alerts are also integrated into the Android platform and will be shown on the screen in the form of a Now card, when an alert is active. In both cases, a

detailed view of the alert is available, see Illustration 10.3.



Illustration 3.3: Alert on Android

Sahana

Another notable example of CAP integration is *Sahana Eden*¹¹, a free/open-source¹² product designed to facilitate disaster and emergency management. It can act as a CAP alert originator¹³, i.e. produce an alert; as well as distribute the alert to end users via SMS¹⁴ and email.

Eden comes with a web-based interface and installation instructions are available for multiple platforms. It covers a wide range of functionality, such as inventory management, a missing person registry, or volunteer coordination; CAP support is just a small component.

It has been deployed in Pakistan, Philippines, Indonesia, Haiti.

Jixel

*Jixel*¹⁵ is a virtual control room software appliance developed in Italy, that facilitates incident management and communication between the agencies involved. CAP support is available, Jixel can produce an alert as well as visualize it on the dashboard. While the software embraces open standards, it is not a free or open-source product.

IPAWS

The Integrated Public Alert and Warning System is an American effort to coordinate different teams involved in emergency situations. It relies on CAP to ensure interoperability is in order.

Summary

It is a widely adapted, mature, comprehensive standard which has been in use for over a decade. Even though it originated in North America, it is an open effort that continues to gain momentum and attract other players.

CAP support is available in several major systems that have already proved themselves useful in field conditions.

Adding CAP compatibility is a mandatory requirement for a notification system used in critical infrastructure.

3.1.2 Section summary

A protocol for alerts should be chosen if the following conditions are met:

- The protocol is standardized and open
- Clear documentation with examples is available
- Bindings or libraries for popular programming languages exist
- The protocol is adopted by several major agencies

Table 3.1: CAP features at a glance

Feature	Implementation
Authentication	XMLSIG, optional
Encryption	XMLENC ¹⁶
Standardized	OASIS, ITU-T
Encoding	XML, ASN.1
Geo-targeting	Coordinates, shapes, 3D
Multilingual	Yes
Validity period	Onset, expiration
Cancellation, Repudiation	Yes
Delivery confirmation	N/A
Feedback	-
Supplemental data	References to audio and images
Adoption	USA, Canada, Australia, international
Layer in network stack	Application

3.2 Security

As the trend discussed in the overview section states, with time security concerns get more serious, which is why it is extremely important that security is present as a key ingredient in a system, rather than be an element added as an afterthought.

A straightforward way to address these issues is to frame the problem as a set of questions and ensure that we apply technologies that answer those questions.

Table 3.2: Basic concepts in security

Question	Concept
Is my conversation peer really who they claim to be?	Authentication
Has the information I just received been altered while in transit?	Integrity
How to ensure that only the recipient can read the data I send?	Confidentiality
How to prevent one from denying that they sent a specific message?	Non-repudiation

These are primordial elements that have to be taken into consideration; an excellent review of the cryptographic primitives that address those questions is available in [2].

The problem will be broken down into sub-components and each of them will be analyzed independently. In each case, the following best practices will be employed:

- **Defense in depth** – multiple, independent layers of security ensure that if one of them fails, the system continues to be secure;
- **Over-engineering** – the system is designed to tolerate loads that exceed the expected ones, thus anticipate new types of attacks that were not invented when the system was created;
- **Need to know** – no component will contain more information than the minimum necessary for it to fulfill its function.
- **Compartmentalization** – in case a component is compromised, the damage is limited to that specific node and does not spread to other ones;

- No component must ever rely on *security through obscurity* as a sole means of defense.

3.2.1 Security infrastructure

The system employs the PKI (public key infrastructure) model, with the following components:

- **CA** – a certification authority that issues digital certificates to entities such as nodes in a network or subordinate CAs. The CA must support the following protocols: CMP¹⁷, OCSP.
- **RA** – a registration authority that receives certification requests and is responsible for verifying the identity of the subject and generating the actual certificate request that will be sent to the CA via CMP. To ensure that end users have a smooth experience, the RA must provide a web interface and support PKCS#10, CRMF¹⁸, SPKAC and requests. This ensures compatibility with Internet Explorer, Firefox, as well as Opera, Safari and Chromium. Support for these request formats also covers certain browsers on smartphone platforms (specifically, Chrome on Android).
- **LDAP** – all issued certificates will be stored for archival purposes using the lightweight directory access protocol. The directory can be searched by different criteria, most commonly one will retrieve information about a certificate by supplying its serial number.

Root CA requirements

At least two layers of CA hierarchy must be implemented: the root CA, which only issues certificates to other CAs, and subordinate CAs that will issue certificates to other subjects. The rationale behind this decision is to reduce the exposure of the root CA to external environments.

The root CA is specifically important, therefore a special policy applies to any interactions that concern it:

- The root CA is a standalone computer that is *not connected* to a network;
- It is *physically isolated* from the rest of the infrastructure, in a room that can only be accessed in the presence of *two staff members* with the necessary privileges;
- Access to the root CA room must be *logged* to a journal;
- *No periphery* is going to ever be connected to the root CA, except the equipment that it was originally set up with: a printer and a web camera. This is to ensure that there are no other

attack vectors that can be exploited (CD-ROM, USB flash disks, etc).

- CSRs will be transmitted to the root CA graphically, via a QR code in which the request data are encoded.
- Outputs from the CA (certificates or delta CRL) will be stored locally and transmitted to the outside world in the form of printed QR codes.
- The secret key is generated by and stored inside a PIN-protected smart-card or token.
- Issued certificates will have a minimal set of “key usage” attributes specifically related to the purpose for which the certificate was issued (e.g. a client certificate does not need the `codeSigning` attribute for digitally signing binaries).

Enforcing these measures will reduce the probability of key compromise, which would in turn compromise the rest of the system.

Security requirements for subordinate certification authorities are not as strict; however, the last items remain relevant in all circumstances – the secret keys are stored in tamper-proof hardware devices such as smart cards, tokens or HSM and CAs shall not issue certificates that bring more power than necessary. This ensures that the secret key can never be copied. If a smart card is stolen, the corresponding certificate is revoked – so the damage is limited; even if there is a window of opportunity while it is still active - the certificate cannot be used to perform actions it was not supposed to be able to handle.

QR codes were chosen because the technology is standardized¹⁹ and can be applied in any context without royalties²⁰. Such codes provide enough capacity to hold a certificate signing request, as well as the certificate itself.

Use of such forms of input does not necessarily eliminate all risks; one possible way to exploit the system is to craft a malicious QR code that would produce a buffer overflow on the target system that interprets the image. However, this can be remedied by screening the requests before passing them to the root CA and by using a decoding QR library that employs bound checking.

Securing a telemetric box

This section describes the security of a box that contains (or is linked to) sensor equipment that reads physical values and reports them back to the server. An example we will consider is a box

connected to a seismometer (see Target system).

- All data transfers towards the main server are digitally signed with a secret key unique to the box. Alternatively, they can be tunneled through a VPN, connection to which is performed via an X509 certificate.
- The secret key is generated by a smart-card or token and it never leaves the device.
- The box includes a battery that supplies at least one hour of autonomy (the time is a function of how long it takes support staff to reach the box).
- Self-diagnostic telemetry about the state of the box is always transmitted to the server, including the temperature, free space and available RAM, battery charge.
- The box is tamper proof.
- The box has an electronic seal²¹ and any breach of the seal will be transmitted to the main server along with other telemetry data.
- Unscheduled box opening results in the revocation of the certificate issued to the box.
- The certificate will be revoked if the box was stolen.
- The box has multiple communication channels, to preserve transmission capability if one of the channels fails.
- Mutual authentication is applied when transmitting data to the server.

Securing the notifications server

This is the component that receives notifications from all the telemetric boxes, then processes and forwards them to the subscribers.

- Its secret key is generated by a smart-card or a token and it never leaves the device.
- It is not exposed to the Internet directly, it only responds to communications through a VPN, thus avoid the situation in Illustration 10.2: Internet-facing control system devices (see page 59).

3.2.2 Cryptographic algorithms

The system relies on standardized algorithms and in no circumstances shall it employ proprietary

ciphers that were not subjected to cryptanalysis by a standardization body.

Ciphers and key lengths shall be chosen in accordance with NIST recommendations, unless overridden by the legislation of the country where they are used. For example, Russia mandates the use of ГОСТ P 34.10-2012²² - the system must use a plugin architecture that allows the use of different cryptographic primitives.

3.2.3 PKI policies

Special consideration has to be given to a number of edge cases, such as what happens when the X509 certificate of a box or of a CA expires. Issuing a certificate with a very long lifetime is not a solution, because it merely postpones the problem instead of solving it. Moreover, as technology advances, keys and algorithms that were strong in the past are not strong enough in the future – therefore expiring certificates are natural way of phasing out old technology and replacing it with modern means.

Several items are highlighted below, while others can be distilled from the principles discussed on page 18, or from documents such as the X509 certificate management policy of the US Department of Defense²³ or Microsoft's certificates life-cycle²⁴.

- Certificates are renewed before they expire, a new CSR will be signed with the “old” secret key while it is still valid.
- Certificates are re-keyed (i.e. a new key-pair is generated) at regular intervals, to ensure that key-lengths match modern security standards; or if the existing key has been compromised.
- Certificate validity periods are a function of the importance of the agent that uses the certificate: less important certificates have shorter life-spans; critically important ones last longer.
- High-profile agents use bigger keys.
- CAs must not issue certificates with a lifespan that exceeds their own validity.

3.2.4 Logging and audit

Each alert notification transmitted through the system must be digitally signed by the sender. This implies that it can only be sent by the possessor of the secret key, hence the system has the property of non-repudiation. Once a message was transmitted, one cannot “take their words back”. Moreover,

alerts cannot be spoofed – because that would require the secret key.

To ensure that the integrity of the log that keeps track of all the activities has not been compromised, it has to meet the following requirements:

- **Off-site storage** (for example, by logging to another server via the `syslog` protocol).
- **Append-only** attributes must be set for the log files.
- **Paper trails** must be available for highly critical operations (e.g. the root CA issuing a certificate).

3.3 Reliability, scalability and performance

For the system to be trustworthy, we have to ensure that it will not fail to render its services in abnormal circumstances and that any anomalies are detected and handled swiftly. This is achieved by using redundancy, backup mechanisms, as well as protocols and technologies that were designed to withstand surges in load.

3.3.1 Clustering and fail-over

The notification broadcast server is a critical component that must not be a single point of failure. If it collapses, subscribers will not receive notifications when they need them, hence the system's primary objective is not met.

This server represents an attractive target for external attackers, because much more damage can be incurred if it does not operate. Therefore, the system must be designed with the knowledge that it can become the target of terrorism or of an APT.

The following aspects have to be taken into account:

- **At least one fail-over server** must be available, such that it can take over when the primary system fails;
- **Geographical separation** ensures that a single natural disaster, logical or physical attack cannot affect the entire system;
- **Backup power sources** are necessary to ensure that outages will not disturb the system's availability;

- **Redundant communication channels** ensure that there is no single connection that has to fail in order to render the entire system inoperable.

3.3.2 Self-diagnostics

The system relies on getting continuous information from remote nodes (referred to as telemetric boxes in Target system on page 8). It is imperative to know with certainty that all the nodes are on-line, not compromised and in a good shape. This is implemented by sending telemetry related to the telemetric box itself (for details, see Securing a telemetric box).

Keep-alive messages must be transmitted regularly, to test connectivity and thus ensure that the telemetric box can send an actual alert when necessary.

4 Transport layer technologies

A reuse of existing standards was discussed in 3.1 Interoperability; however, the picture is incomplete without a solution for transporting the alert message. Such technologies are referred to as *transport layer*, despite being slightly out of tune with the ISO/OSI definition of “transport layer”.

For example, certain protocols operate “on top of HTTP”, i.e. they use HTTP for transport, even though technically the protocol at the transport layer is TCP. Another example that will be discussed later is the use of Twitter for the dissemination of notifications; Twitter itself operates on top of HTTP, while some other service can run on top of Twitter. In that case we say “It uses Twitter for transport”, remembering that underneath that there is HTTP, and lower still is TCP.

4.1 Twitter

This is a social media service that allows one to post a 120-character long message that will be distributed to their subscribers (known as followers). Messages, known as tweets, can be correlated with each other via the use of a hashtag, such as `#earthquake`. Its subscriber base is not public, but estimated between 232 million²⁵ and 645 million²⁶ users around the world.

One notable example of alerts via Twitter is the USGS – U.S. Geological Survey, a facility called “Tweet Earthquake Dispatch”²⁷. The institution publishes alerts via two official accounts, @USGSSted and @USGSBigQuakes.



Strong earthquake,
ROMANIA,
Nov-22 19:14 UTC,
3 #deprem tweets/min,
on.doi.gov/1tcdIRS

Illustration 4.1: USGS earthquake tweet

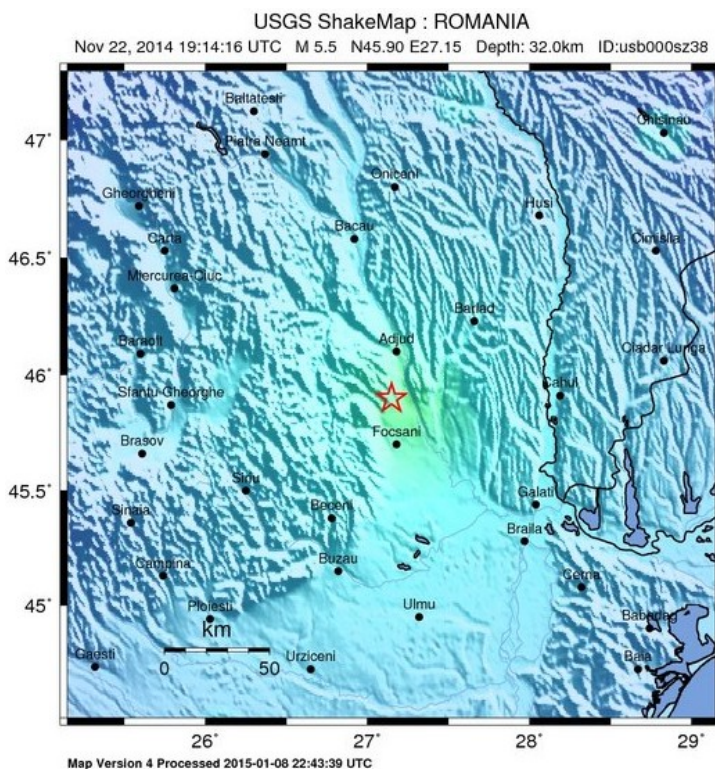
When an earthquake occurs, a tweet is posted by a bot and delivered to the followers of @USGSSted.

Some people use Twitter with their mobile phones and tablets, they will receive a *push* notification about the tweet if the client was configured to do so. In other words, the alert comes to the subscriber informing them that “there is an earthquake”, rather than the subscriber having to manually ask “is there an earthquake?” every now and then, in case there is one.

The format of the messages is not an official standard, but it adheres to a structure that is made public on the USGS web-site:

earthquake tweets contain a magnitude descriptor, location, origin time, and a link to the USGS webpage with the most recent information about the event. In addition to the seismically derived parameters, the alerts also include the frequency of tweets in a region surrounding the event that contain the word “earthquake” or its equivalent in several languages. Our observations show these tweets often originate from people who have experienced the shaking effects of the earthquake. After some significant earthquakes, @USGSed will also tweet supplementary information about the event.

The shown tweet refers to an earthquake in Romania in November 2014, but we can also infer that the tremors were felt in Turkey, because #deprem was tweeted 3 times per minute, which is Turkish for “earthquake”. Strangely enough, the service did not mention the Romanian word “cutremur”, even though the calamity was felt in neighboring Moldova too, so there were millions of people who thought “cutremur” when that happened.



PERCEIVED SHAKING	Not felt	Weak	Light	Moderate	Strong	Very strong	Severe	Violent	Extreme
POTENTIAL DAMAGE	none	none	none	Very light	Light	Moderate	Mod./Heavy	Heavy	Very Heavy
PEAK ACC.(%g)	<0.05	0.3	2.8	6.2	12	22	40	75	>139
PEAK VEL.(cm/s)	<0.02	0.1	1.4	4.7	9.6	20	41	86	>178
INSTRUMENTAL INTENSITY	I	II-III	IV	V	VI	VII	VIII	IX	X+

Scale based upon Worden et al. (2012)

Illustration 4.2: Earthquake details via USGS

Latency

Tweets are not suitable for an instant notification system, as discussed in Real time communications. Given that USGS also indicates how many times earthquake-related keywords were tweeted in the area, the implication is that the bot has to analyze a large number of tweets in order to collect some samples. This adds latency to the communication process, taking away precious seconds that may have been critical in a life or death situation.

Another factor concerns usability – not everyone who has Internet uses Twitter, not everyone who uses Twitter has a smartphone, not everyone with a smartphone and Twitter has a permanent Internet connection, and even if they do – they haven't necessarily configured it for

push notifications.

This rules out Twitter as a transport for notifications that have to be delivered within a second or less, but it is still viable for cases which we are dealing with alarms that concern the future that is at least 10 minutes away. The rationale is that the information can spread via word of mouth and other social media – which is much better than not using Twitter at all.

Reliability

Despite the fact that Twitter has millions of users, it was not designed to act as a guaranteed method of delivering critically important information. The service does have a “Twitter Alerts” feature²⁸, but its description fails to provide any guarantees of quality of service.

The purpose of the product is to deliver critical alerts to users, either via push or via SMS; they will be displayed in a different visual style, attracting attention. The product has been instrumental during events²⁹ such as super-storm Sandy, the earthquake and tsunami in Japan in 2011 or the Boston marathon bombing.

Feedback

One of Twitter's strong sides is the ability to provide feedback by tweeting back to the original tweeter. In other words, it is not just a tool for disseminating information, but also for collecting reactions.

4.2 SMS

Short Message Service is a standardized³⁰ technology that enables us to send and receive texts using a mobile phone or a modem with a SIM card. The technology is well-established, the first specification³¹ goes back to 1994, thus giving it two decades of field use at the time of writing.

Due to its maturity, SMS functionality is supported by all mobile phones on the market today, people are accustomed to this feature and expect it to be available by default.

There are several defining characteristics:

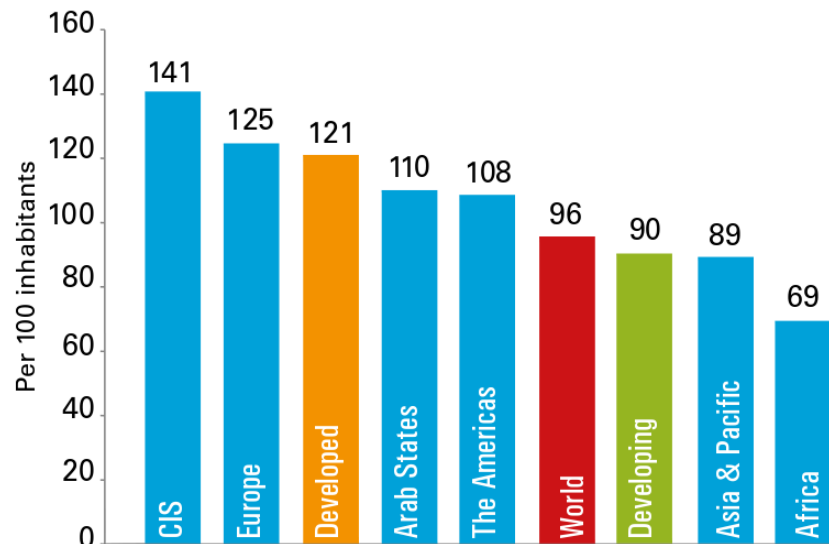
- Message length is capped at 160 characters; it will be less for texts that use an encoding other than GSM 7-bit;
- Concatenated SMS allow longer messages to be sent, they are broken down and assembled

automatically by the sender and the recipient, respectively;

- Delivery receipts are available.

Ubiquity

According to statistical reports³² from ITU, the number of mobile-phone subscribers is about to reach the total population of the planet. At the end of 2014, the worldwide average is 96 phones per



100 inhabitants, the rate goes above 100 in industrialized countries. This implies that SMS can be potentially used by a very large target audience. In fact, an earlier ITU report estimated the number of sent SMS in 2010 at 6.1 trillion³³, three times as much as in 2007.

Illustration 4.3: The ubiquity of mobile phones in 2014

Push

SMS are pushed onto a recipient's phone, i.e. one does not have to continually check if they have new messages; instead – one is notified when they have a new text. This is a major plus from the usability perspective, as there is no need to keep polling a server, thus resources are not wasted.

Usability

Flash SMS are a special type of message that will be shown on a recipient's phone's screen without any form of user interaction. This is important in the context of emergencies, as it minimizes the time required to become aware of potentially life-saving information.

Reliability

Although SMS is a common technology, there are some caveats, reliability being one of them. A message will not be delivered to the recipient if the network is congested.

This becomes especially important when a message targets a large group of people - when the

network is fully saturated, some will simply not receive the text in a timely fashion, if at all. An attentive and introspective reader might correlate this with their experience when sending best wishes to friends, seconds before a year ends.

Speed

Although SMS delivery is perceived as instant, it is only so when the network is not loaded and messages are sent occasionally. When there is a spike in text communications (such as at the end of a year), some are not delivered and re-enqueued by the SMS center for later delivery. This is why some of us continue receiving “Happy new year” messages hours after midnight on January the 1st.

SMS delivery is *not real-time*, in the terms discussed in The importance of “instant”, depending on the network load, the time of delivery can vary.

SMS sending is *sequential*, i.e. if sending a message to a recipient takes N seconds, sending it to 50 people requires 50xN seconds. There is no broadcast facility; thus even if the first recipient gets the message swiftly, it is, by definition, going to take longer to reach the last person in the list of recipients.

Based on these criteria, SMS have to be ruled out as a transport for notifications regarding critical infrastructure or emergencies.

A detailed review of reliability, speed and security issues is provided in “Characterizing the limitations of third-party EAS over cellular text messaging services”³⁴.

Security

There are concerns on this front as well. SMS can be spoofed, there is no possibility to digitally sign or encrypt them – thus a notification cannot be private (technically, the mobile operator can access it) or authentic (no formal way to prove it really came from the sender).

This implies that there can be fake alerts or fake alert cancellations.

Feedback

SMS can be received and replied to, hence there is a possibility of collecting feedback and using it to automate certain procedures.

4.3 Cell broadcast

Although an SMS cannot be sent to multiple recipients simultaneously, broadcasting in mobile networks can be achieved via *cell broadcast*. The service is standardized by ETSI³⁵, it operates on top of a GSM network, thus it can leverage the ubiquity of this technology (see SMS above).

In essence, it can deliver a message to a group of subscribers connected to the same cell. As in the case of SMS, messages are pushed to the recipients. Here are the basic characteristics of cell broadcast messages:

- Length of up to 82 bytes, allowing 93 Latin characters in GSM-7bit encoding;
- Can be displayed automatically on the screen;
- Does not depend on network load;
- Broadcasting reaches all the terminals currently connected to a cell at the same time.

Usability

Notifications can be displayed automatically on the screen and optionally, it can be accompanied by a distinguishing sound.

Specific messages can be sent to each cell, meaning that they can be tailored to the context, e.g. adapted to the local weather conditions in the area, translated to a specific language, contain addresses related to a zone, etc.

Messages will also be delivered to international guests, as long as they are connected to the network; there is no need to have a SIM card of a local operator.

A possible drawback is that depending on the phone brand and model, cell broadcasting may be turned off by default. Since it is a rather obscure feature, non tech-savvy people may not be aware of it or fail to understand what it is for when they stumble upon it. One way to deal with that is to write legislation that mandates operators to enable it by default or provide a facility to turn it on remotely³⁶.

Reliability

Cell broadcasts do not rely on the same network signaling as voice or data, which is why they never produce network congestions and will always be delivered reliably, regardless of how many subscribers are out there. To put this in the terms used in the description of SMS, a “Happy new year” cell broadcast would *immediately* reach *everyone* in a geography.

Speed

Cell broadcasts are delivered in the same amount of time, whether there is just one recipient or ten thousand of them; this means that there is less room for variation in the amount of time it takes for the notification to arrive. As soon as the message was processed by the operator's equipment, it is broadcast into the medium and the laws of physics get to do their work. This makes cell broadcasts a *near real-time* technology, which is a plus for emergency notifications.

Security

Unlike SMS, cell broadcasts can only originate from the equipment of the mobile operator, not from anyone currently signed onto the network. Sending fake alerts would require compromising the security of the operator or coercing an authorized employee into doing so. While this is possible in principle, the bar is higher than in the case of SMS.

There are privacy advantages as well – messages are delivered to everyone in a target area, rather than to a specific phone number; this implies that the recipient remains anonymous.

Feedback

Cell broadcast messages provide no way to reply, thus it is a “send only” feature.

4.4 Email

Electronic mail is widely used and is one of the first services that were a part of the original Internet, specifications that define how to transmit messages were already available in 1982³⁷, SMTP is still used three decades later. The technology is very mature and nowadays it accounts for a large chunk of global correspondence. A notable example of email used for emergencies is the USGS earthquake notification service³⁸.

Poll and push

Post Office Protocol, known as POP, is the method typically used to retrieve new emails, it was defined in RFC937³⁹, in 1984. It is a polling protocol, which means that a client has to connect to the server at regular time intervals and ask “are there any new messages”? This is inefficient, because network resources are wasted on checking for updates even when there are none. Further, if a new message arrives between checks, it will be detected when the next check is performed, thus there is

more latency.

A better alternative is IMAP IDLE⁴⁰, which adds push notifications and enables a client to get notified about new messages immediately. It works by keeping the TCP connection active and notifying the client when a new message arrives; afterwards, the client can retrieve the message from the server.

Some⁴¹ email providers provide push emails that are based on proprietary technologies other than IMAP IDLE; while they offer the same user experience, their use is not recommended (see Vendor lock-in on page 52).

It must also be pointed out that even though the underlying TCP connection can have an indefinite lifetime, an IMAP server will disconnect a client after a timeout of inactivity, unless special measures were taken to reconfigure it accordingly, or the client sends another IDLE command shortly before the timeout, here is the relevant quote from the specification:

The server MAY consider a client inactive if it has an IDLE command running, and if such a server has an inactivity timeout it MAY log the client off implicitly at the end of its timeout period. Because of that, clients using IDLE are advised to terminate the IDLE and re-issue it at least every 29 minutes to avoid being logged off. This still allows a client to receive immediate mailbox updates even though it need only "poll" at half hour intervals.

Speed

Messages are delivered via intermediate nodes on the Internet, each of them is free to apply its own rules for processing filters, thus incurring unpredictable delays. Email is not suitable for contexts where hard real-time is imperative, unless messages travel only within a confined perimeter where the infrastructure is under our full control and special precautions were taken to ensure the data travel predictably.

However, the effort necessary to make email behave deterministically might be greater than the time necessary to deploy a solution that was originally designed for push and for real-time operation.

Security

Although emails are transmitted in plain-text, there are extensions (such as S/MIME⁴² or OpenPGP⁴³) that provide the means to ensure that the messages are authentic, unchanged, and even concealed while in transit over public networks. This is achieved by leveraging cryptographic algorithms: a message can be digitally signed and encrypted, such that the designated recipient can

independently verify the message and assess its trustworthiness. These security mechanisms are discussed in detail in 3.2 Security.

However, there are some aspects that must be taken into account:

- **Spoofing** – in a default configuration, an SMTP server will accept a message and its metadata as is, giving the sender an opportunity to populate the headers with whatever information they please. As a result, the sender's identity can be forged with minimal effort. Measures such as SPF or DKIM can be applied to verify the authenticity of the sender; while the message itself can be encrypted with S/MIME.
- **Privacy** – the payload of an email can be encrypted, while its headers have to remain in plain-text; which implies that an external observer knows the subject of the message or who the sender and the recipient are; sometimes this can be too much.
- **Spam**⁴⁴ - the practice of flooding a recipient's mailbox with junk messages that advertise irrelevant products or services leads to annoyance, but in the worst case it could:
 - fill the recipient's quota and make it impossible for them to receive new email
 - distract the person's attention from an important message

The concerns above can be addressed by keeping the email infrastructure separate from the rest of the Internet; using public email servers for critical infrastructure or government services is not recommended.

Other characteristics at a glance

- Message lengths are practically not limited, though constraints can be (and usually are) set by the mail server administrator;
- Messages can contain attachments with arbitrary payloads, such as an XML containing a CAP alert;
- Various routing schemes can be applied to deliver a message to multiple recipients;
- Feedback can be provided by replying to the email.

4.5 XMPP

Extensible Messaging and Presence Protocol is an open standard⁴⁵, rooted in 1999⁴⁶; it was originally known as Jabber and designed as a decentralized instant messaging solution. There are a number of free/open-source implementations of the protocol, as well as libraries that make it possible to integrate an XMPP client into a program with minimal effort.

The protocol philosophy allows extensions to be added, thus new features can be introduced, giving enough flexibility for future growth. This advantage can be leveraged and XMPP can be applied for purposes other than instant messaging – notifications, inter-process communication on different hosts, or remote monitoring⁴⁷ of sensors.

XMPP's maturity, openness and the ubiquity of client libraries for different languages are pivotal to its popularity; it is used internally by Facebook chat⁴⁸, supported by Skype⁴⁹ and Google Talk⁵⁰.

Security

There protocol addresses concerns of confidentiality, data integrity, peer authenticity and non-repudiation. Unlike in the case of other protocols, XMPP's security features were baked into the design⁵¹ and the basic functionality is a mandatory requirement for compliant implementation, so XMPP has a good track record.

TLS is used to provide confidentiality and integrity of the data; SASL protects against user spoofing. Peer verification can be performed via X509 certificates; the RFC encourages the use of SHA-256, and OCSP. The standard also takes into account edge cases such as the expiry of a peer's certificate while the dialog is in progress – the desired behaviour is to gracefully close the connection.

Highlights

- Push notifications are supported, as XMPP keeps a TCP connection continuously alive.
- Binary data cannot be sent directly, unless it is encoded in base64 or an alternative method of transfer is used (e.g. send a URL for an FTP or HTTP download).
- Reliable delivery is not available, but this issue, like many others, can be implemented by adding another layer of abstraction.
- As the protocol is primarily designed for instant messaging, feedback collection from recipients is obviously possible.

4.6 AMQP

Advanced Message Queueing Protocol is a relatively new player that aims to become the most comprehensive protocol for interoperability between messaging middleware. It became an ISO⁵² standard in 2014 and the number of mature implementations testifies to its quality – RabbitMQ, ActiveMQ and Windows Azure Service Bus are a few notable examples.

Messaging patterns

AMQP facilitates typical communication patterns such as publish-subscribe, fan-in, fan-out and request-response. This is an important feature, because it provides the potential to use it as an underlying mechanism for a larger system. For example, a CAP alert can be received by multiple consumers simultaneously in a fan-out pattern, each of the receivers will process and adapt the message to different media, such as SMS (where it must not exceed 160 character), Twitter (where the limit is 120 characters) or a Cell Broadcast (where the length is at 93 symbols). Other consumers can be responsible for logging the data or visualizing it on a dashboard.

Push

Messages can be pushed to consumers, thus AMQP is suitable for context where having a very low latency is important.

Security

The security section of the protocol v0.9 specification is very brief⁵³, focusing on the messaging itself, namely – possible buffer overflows and how to handle denial of service attacks.

Securing communications is delegated to the underlying technology, such as TLS.

As of version 1.0, the specification provides detailed security guidelines that discuss the use TLS and SASL.

4.7 MQTT

Message Queue Telemetry Transport is a lightweight protocol designed to transmit data from sensors, it is specifically aimed at cases when there is little processing power, memory or network bandwidth available, hence its lightness. As of 2014 it is an open standard⁵⁴ managed by OASIS, currently at version 3.1.1.

Its primary goal is to implement the publish-subscribe paradigm, where data from a sensor are

sent to a broker, which then relays them to subscribed parties.

Reliability

MQTT provides a number of quality of service options, that can ensure that the message arrives exactly once, at least once or at most once.

The protocol also provides a feature called “last will and testament” that enables the broker to publish a message on the behalf of a subscriber that disconnected abnormally. This can be applied to track the state of a sensor's connectivity.

The transferred messages are acknowledged – thus MQTT's reliability is good.

Security

The protocol itself is concerned with delivering the data reliably, so the standard does not mandate the implementation of specific security primitives. However, it encourages the use of TLS as an underlying transport⁵⁵.

Therefore, in the case of MQTT, the security level of the infrastructure depends on the broker that was chosen and its configuration.

4.8 Summary

A number of technologies were reviewed in this chapter, providing insights about which if them is most suitable for a specific context, a comparison table that places them side by side is available on the next page.

In conclusion, we can establish several key points:

- MQTT is the preferred choice for relaying telemetry data from sensors to a checkpoint; for example – a set of seismic sensors can continuously send their readings to an MQTT broker. In this case we leverage the “last will and testament” feature, as well as the fact that the protocol incurs little overhead.
- AMQP is an excellent choice for sending the data to other systems, in this case we leverage its routing capabilities that will efficiently distribute the message to different consumers, based on different criteria.
- Cell broadcast is the best solution when a notification has to be delivered to a large number of

people in a specific geographical area, with minimal delay.

- Social media can be an additional outlet that will continue to disseminate a message to a global audience. Note, however, that it has to be an additional transport, not the sole one.
- XMPP is a suitable candidate, especially when other parts of the infrastructure already use it elsewhere; though it has to be kept in mind that out of the box, the protocol does not offer guaranteed delivery.
- SMS is not a recommended way of disseminating alerts, it is not reliable when the network load is high and it cannot guarantee fast delivery of data.
- When the protocol itself does not provide security, this can be addressed by tunneling communications through a VPN or TLS connection.

4.9 Solutions compared

Table 4.1: Messaging solutions compared

Technology	Twitter	Google Public Alerts	SMS	Email	Cell broadcast	TCP	UDP	XMPP	MQTT	AMQP
Standard	no	no	ETSI TS 100 901	RFC 2822	3GPP TS 44.012 ⁵⁶	RFC 793	RFC 768	RFC 6120	OASIS	ISO/IEC 19464
Security			GSM	S/MIME, PGP	Yes, authentic message	TLS, payload encryption	DTLS, payload encryption	Yes	TLS, payload encryption	TLS, payload encryption
Feedback	yes		yes	yes	No	yes	yes	yes	Yes	Yes
Delivery	Broadcast, point to point		Point to point	Point to point, multicast groups	Broadcast	Point to point	Point to point, broadcast	Point to point, multicast via extensions	Publish-subscribe	Point to point, broadcast, multicast, routing
Message size	120 symbols		160 bytes	Not limited ⁵⁷	93 bytes	Not limited	~64KB	-	256 MB ⁵⁸	2 ⁶⁴ bytes
Delivery confirmation	Yes (HTTP)		Yes	Yes	No	yes	No, has to be checked above	No, has to be checked above	Yes, 3 levels of QoS	yes
Latency	Low	Low	Low	Average	Very	Extremely low	Extremely low	Very low	Very low	Very low
Physical medium	Any	Multiple	Radio	Any	Radio	Any	Any	Any	Any	Any

5 The importance of "instant"

Notifications have to be delivered to the subscriber in a time-frame that gives them sufficient room for maneuver. A reminder about an unpaid bill can be delivered any time within an hour, while notifications about a departing bus require minute precision. In contrast, in the context of high performance trading or the control of a nuclear reactor, a much higher resolution is required and a difference of milliseconds can be crucial.

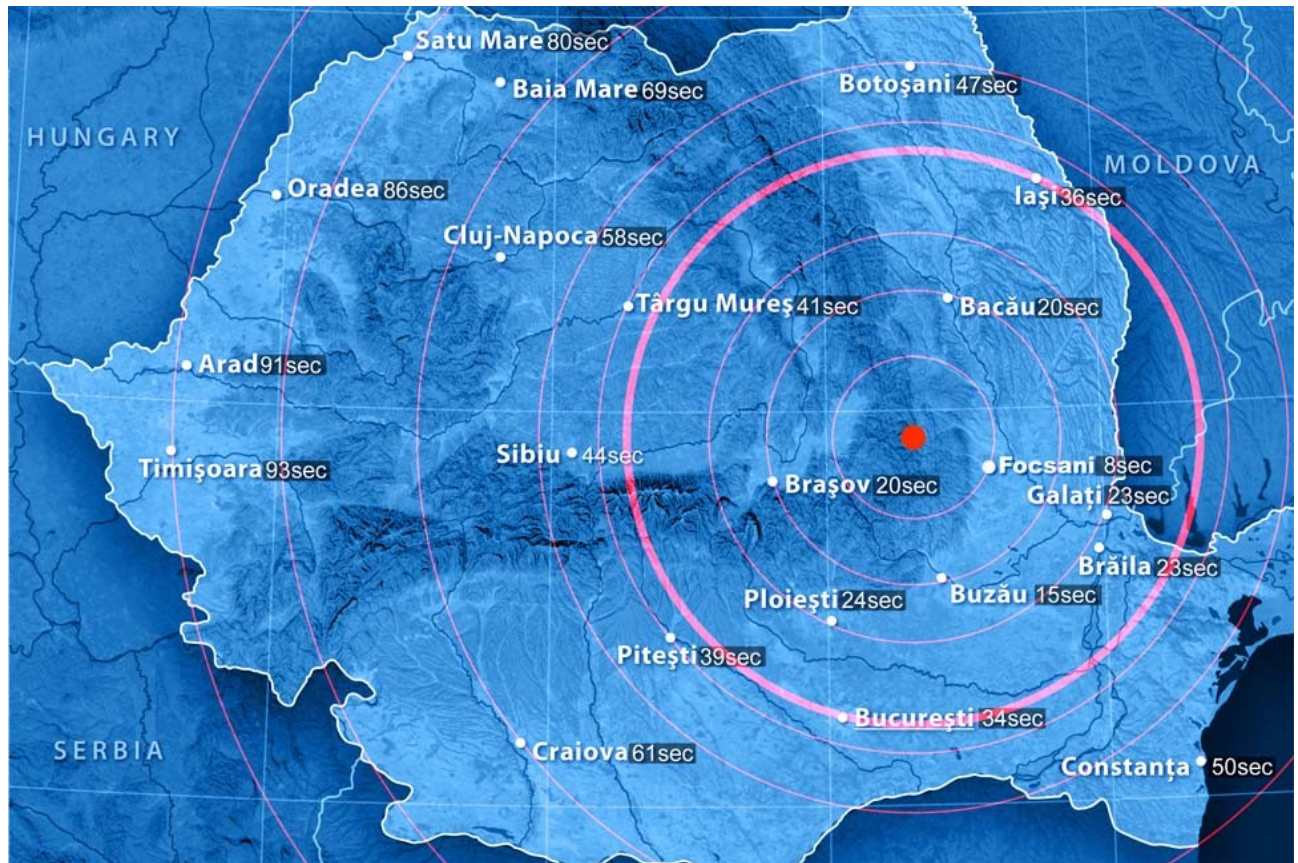


Illustration 5.1: Seismic wave propagation

The map⁵⁹ visualizes the time it takes a seism centered in Vrancea (Romania) to reach a specific region. While someone in Oradea has over a minute and a half to react, residents of Focșani will be affected in as soon as 8 seconds. If the notification arrives in ± 4 seconds, some recipients will get it on time, while others will receive it during or after the earthquake itself. Such late notifications will be useless, or even harmful, because they distract the recipient in a potentially life-threatening situation. Therefore, it is imperative that the notifications arrive within a predefined interval, or not arrive at all.

Such precision is not relevant for a human being, because the sum of operations necessary to

receive and read an SMS is longer than what it takes for the shockwave to reach that location. The process of hearing a ring-tone and becoming aware of it, getting the phone out of a pocket, unlocking it and viewing the SMS adds up to a significant amount of seconds.

Automated control systems, on the other hand, do not possess the *qualities* that make humans human, thus they will benefit greatly from notifications that arrive within a predefined time interval. A shutdown procedure can be initiated, the infrastructure can be switched into a safe mode of operation, etc. The system will not “forget” to do it, nor will it “accidentally skip a step” or “press the wrong button”. Here are some examples of reasonable reactions to natural emergencies:

Table 5.1: Emergency procedures

Calamity	Infrastructure	Reaction	Time frame
Earthquake	High-speed train	Slow down gradually and come to a full stop	≤ 30 s
Earthquake	Elevator	Stop at the nearest floor and leave doors open	≤ 5 s
Earthquake	Data center	Park heads of hard disk drives	≤ 2 s
Earthquake	Hospital	Suspend work in progress to avoid mistakes ⁶⁰	< 1 min
Volcanic eruption	Air traffic control	Redirect traffic around projected cloud path	> 1 min

5.1 Hard “real time”

Modern operating systems can run multiple programs simultaneously, acting as if they are running in parallel. This is only true for systems that have multiple processors; in all other cases programs are executed sequentially: the CPU switches its focus from one to another so often that a person perceives the entire experience as a continuous, parallel one. This technique is called *multitasking* and is employed by a typical Windows machine. A person can be lead to believe that the system operates in real time, because it reacts swiftly to their input: you make a click – an image is shown instantly; you press a button – you hear a beep right away, and so on. This is an illusion.

The order in which the processes will be executed is defined by the operating system's scheduler, it relies on its own criteria to determine what process gets to be active at a specific moment in time. This is suitable for a personal desktop or even for a server, but in the context of critical infrastructure,

precious milliseconds can be lost simply because the scheduler decided that CPU time had to be dedicated to another task.

A real-time operating system, in contrast, is designed to provide formal guarantees that a specific task will be executed in a predefined amount of time. The emphasis is on the fact that this amount of time is deterministic, i.e. known in advance, rather than on the fact that this amount of time will be short.

Strong guarantees are needed for deployments where the infrastructure must react to an emergency in a finite amount of time. Such guarantees can be offered by a real-time operating system. Therefore we conclude that in contexts where the stakes are high (in terms of human lives or material cost) and determinism is necessary, an RTOS must be used.

A number of commercial and FLOSS operating systems that meet this requirement are available: QNX⁶¹, VxWorks⁶² or FreeRTOS⁶³.

An alternative approach is to rely on special versions of the Linux kernel that provide hard real-time features: RTAI⁶⁴, Xenomai⁶⁵ or RTLinux⁶⁶. These mechanisms run along with a traditional Linux kernel; thus allowing the same platform to be used with both: conventional and real-time software.

Finally, an operating system can be skipped entirely and instead the solution can be implemented in a dedicated hardware device, or custom firmware can be written for a micro-controller. Since there is no operating system, the device will perform only a specific function that was baked into its circuit or its firmware.

Table 5.2: Real-time solutions compared

	Real-time OS	Real-time kernel	Dedicated hardware
Complexity	Special APIs necessary	Below average	high
Development tools	Specific	Typical	Platform specific
Integration	Separated hardware	Platform can be shared	Separated hardware
Flexibility	Average	High	Low

5.2 Real time communications

A notification has to be transmitted to a remote agent, employing some form of communication. It is cost-effective to achieve this by leveraging an existing infrastructure - the Internet. This network is fast, fault-tolerant, well-supported by a very wide range of hardware and is easy to extend; the key benefit is that it already covers every part of the planet where human civilization is present. Its ubiquity lead to the fact that a lot of other services that have previously used their own channels are now routed over IP, the prime examples are telephony and television, which transitioned to VoIP and IPTV respectively.

However, there are several important factors that must be taken into account:

- *Packet-switching* - the Internet protocol design does not guarantee delivery of a packet, some packets could be duplicated; moreover – each packet might arrive to the destination via a completely different path. The obvious implication is that there is no determinism when it comes to predicting exactly how long it will take for a packet to arrive.
- *Decentralization* – the network is not under the full control of any single entity and it is of reason to expect that packets will cross geographical and political borders. At times this can be a problem: a country's government can selectively block⁶⁷ traffic or turn it off⁶⁸ altogether; an accident⁶⁹ can cut off a country or a region of the world.

Based on the above, we can conclude that the Internet is not suitable for the transmission of notifications in real-time, unless specific measures are applied.

5.2.1 Real-time network stack

A network stack such as RTNet⁷⁰ provides a subset of the Berkeley sockets API to a GNU/Linux operating system with a custom kernel, offering deterministic UDP and TCP over IP to real-time processes.

With this element in place, it can be guaranteed that a real-time process can deterministically push something into a network. Due to the fact that RTNet offers a standardized API, no changes will be necessary⁷¹ in the software that sends data.

5.2.2 Dedicated network resources

Once the data are in the transmission channel, we can expect it to arrive at the destination in an

amount of time known in advance, if one or several of these elements are considered:

- *A dedicated segment* exists between the source and the destination. If it is not used for other agents' traffic, then there is no possibility that a node will get congested or queue our packets while handling other data.
- *Dedicated networking hardware* is used to process the packets, thus we ensure that the amount of processing time is constant, because everything is carried out through physical logical circuits that behave predictably.
- *A real-time OS on the intermediate nodes* is an alternative that enables us to achieve deterministic processing of packets on a system running on general purpose hardware.
- *Static IP routes* ensure that effectively there is no packet switching, all the packets will flow through the same path. This implies that a path is chosen beforehand and all intermediate nodes in it are configured accordingly.
- *Quality of service* agreements can guarantee timely delivery even when packets travel through a public network that is shared with other agents. In this case, network configurations are used to ensure that our packets will be prioritized and handled before other traffic; thus there is no need to deploy our own network infrastructure, as long as static routes are used to ensure packets flow through the same path.
- *Direct layer 2⁷²* communication can simplify the architecture – since there are no packets involved, because we are operating at a lower layer of abstraction.

5.2.3 Sub-perfect is good enough for practical purposes

While it is clear that a civilization that put a human on the moon and successfully landed a probe on a comet⁷³ can build a network that will deliver a message in real-time, we have to consider that in many cases the costs will be highly prohibitive. Therefore we have to consider the possibility of settling for less than perfect, but good enough for practical purposes. A context where this works very well is an experiment in neuro-engineering, in which a monkey in the USA controls a robot in Japan, via electrodes attached to its brain; the round-trip between the monkey and the robot is 20 ms less⁷⁴ than between the monkey and its own muscles.

For some scenarios, using the Internet 'as is' might be good enough for all the possible use cases.

The only aspect that requires additional care is to have several redundant links, if the primary one fails.

5.2.4 Conclusion

A wide range of solutions are available, thus the problem is technically resolved.

Table 5.3: Communication methods compared

	Internet	Real-time Net	Dedicated channel
Cost	Inexpensive	Medium	Expensive
Complexity	Low	Medium	
Compatibility	Excellent	Good	Average
Existing coverage	Excellent	Build your own infrastructure	Build your own infrastructure or lease
Performance	Good enough for practical purposes	Real-time	Real-time

6 System design

The functionality discussed in the previous chapters is built on top of a software stack. This chapter describes the role of each component and its relationship with the others.

6.1 Components

6.1.1 Messaging broker

This component is responsible for receiving, processing and distributing notifications. It is a free and open source messaging broker called RabbitMQ⁷⁵. Several reasons are at the foundation of this choice:

- **polyglot broker** – RabbitMQ can receive messages using one protocol and reply via another. It supports AMQP and MQTT, both of which will be applied in the system. It also supports STOMP and STOMP-WS, thus making it easier to connect to with a web-application running in a browser;
- **clustering and failover support** – this quality makes it suitable for high-availability systems, such as ours; multiple methods⁷⁶ are available: shovel, federation, clustering;
- **free and open source** – the code of RabbitMQ is available , so is commercial support;
- **documentation** is very detailed, it includes installation, configuration and management tutorials, as well as source code of clients written in different programming languages;
- **cross-platform** compatibility covers Linux, FreeBSD, Windows and Solaris, among others; it also runs on VxWorks, which is in tune with the analysis given in Hard “real time” (see page 40 above);
- **security** works out of the box, with support for authentication via LDAP or X509 certificates; TLS v1.2 is supported as well;
- **powerful routing** capabilities provide the flexibility necessary to implement diverse communication patterns necessary in an infrastructure;
- **plugin architecture** – RabbitMQ's functionality can be extended, in case the standard components are insufficient.

The broker is configured to run multiple exchanges, each of them will handle notifications from different types of sensors. Routing schemes are used to deliver the message to other subscribers or processing tools.

6.1.2 Database

PostgreSQL⁷⁷ is a cross platform, free, open-source, relational database.

- **Unlimited database size**⁷⁸ means that very large amounts of data can be stored, the bottleneck is not going to be the database itself;
- **SQL 2011 compliant**, with ACID (atomicity, consistency, isolation, durability) support, transactions;
- **Bindings** are available for a broad range of programming languages.

The database is where all the alerts are stored for archival purposes. It must reside on a separate system.

6.1.3 VPN server

OpenVPN⁷⁹ is an open source, free, cross platform VPN server. Its role is to unite all the agents of the system into one network that is not accessible from the rest of the Internet.

It supports mutual X509-certificate authentication and there are clients for multiple platforms, including Android, Linux, Windows.

Another important advantage is that the VPN can be configured such that the clients cannot “see” each other, this option is leveraged in order to compartmentalize the network and limit potential damage.

6.1.4 PKI components

These are the elements responsible for maintaining the security framework of the system: the root CA and a number of subordinate CAs that issue certificates to telemetric boxes or other nodes. Below is a suggested trust hierarchy that reflects a possible scenario for the Republic of Moldova – a CA dedicated to issuing certificates to earthquake-related entities, and another one for biological hazards.

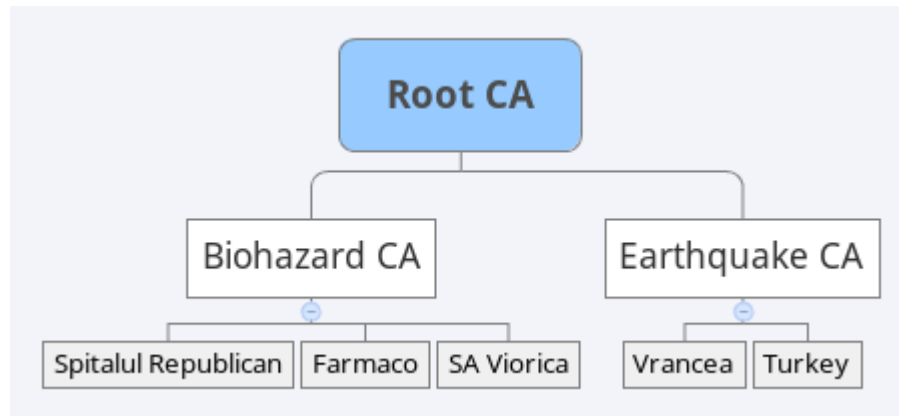


Illustration 6.1: Suggested CA hierarchy

In the context of CA software, the choice depends on the answer to the question “whom do we trust?”. In one case an authority might prefer a free, open-source solution; in another – it could be a commercial solution such as Dekart CA⁸⁰. As long as the CA is standards-compliant, there will be no interoperability issues.

6.1.5 LDAP

A directory server is responsible for storing certificates issued by CAs and other CA-related data, such as the CRL. This allows nodes to swiftly verify the identify of their peers when establishing a connection, by retrieving their certificate from the LDAP server.

The recommended choice is OpenLDAP⁸¹, a cross-platform, open source directory. It is a mature implementation suitable for our scenario.

The directory hierarchy will mirror that of the CA chain of trust, issued certificates will be leaf nodes in the directory tree.

6.1.6 Telemetric boxes

This is the equipment placed on-site. Most choices are problem-specific, so it is not of reason to impose constraints at this point. However, it must be emphasized that regardless of the operating

system chosen to run this equipment, the monitoring software must use the MQTT protocol for telemetry transmissions.

This choice is mandated by the fact that MQTT provides a “last will and testament” feature, that will notify others about the disappearance of the unit. Without MQTT, this feature would have to be re-implemented independently.

6.1.7 Processing modules

These are the individual applications that subscribe to exchanges on RabbitMQ and are notified when a new message arrives. Routing rules will be applied to deliver specific messages to specific workers. These software modules run on a system separate from the broker itself.

There are several basic workers:

- **Logger** – write all of the transmitted alerts to a database, for logging purposes;
- **CAP Interop** – a worker that receives generic notifications, transforms them into a CAP alert, according to the specifications, and passes the alert on to other peers.

Optionally, the set can be extended to ensure that notifications reach a broader audience:

- **Twitter bot** – take incoming notifications and publish them on Twitter in a format suitable for that platform;
- **Cell broadcast module** – this component is responsible for producing the content for a cell broadcast and passing it to mobile operators for dispatch to mobile subscribers;
- **SMS sender** – produce and send a text to a mobile subscriber. Note that such a method of communication is not suitable for mass messaging (see Reliability on page 28); however, it can be acceptable when the notification is supposed to be delivered to a small group of people, such as a support team;
- XMPP, Email and other protocols can be supported by writing specific workers and subscribing them to a specific exchange.

Note that not all of these systems must provide the same level or reliability and perform equally fast. For example, a Twitter bot does not necessarily need to run on a hard real-time platform, instead it can be located on a conventional system. The same principle applies to everything else – the system

becomes easier to manage, as it is distributed cross a number of physical machines that are independent from each other, instead of being lumped together in one place.

6.2 Graphical representations

UML diagrams visualize the components of the system and the links between them. Below is a high-level overview of everything except the security infrastructure.

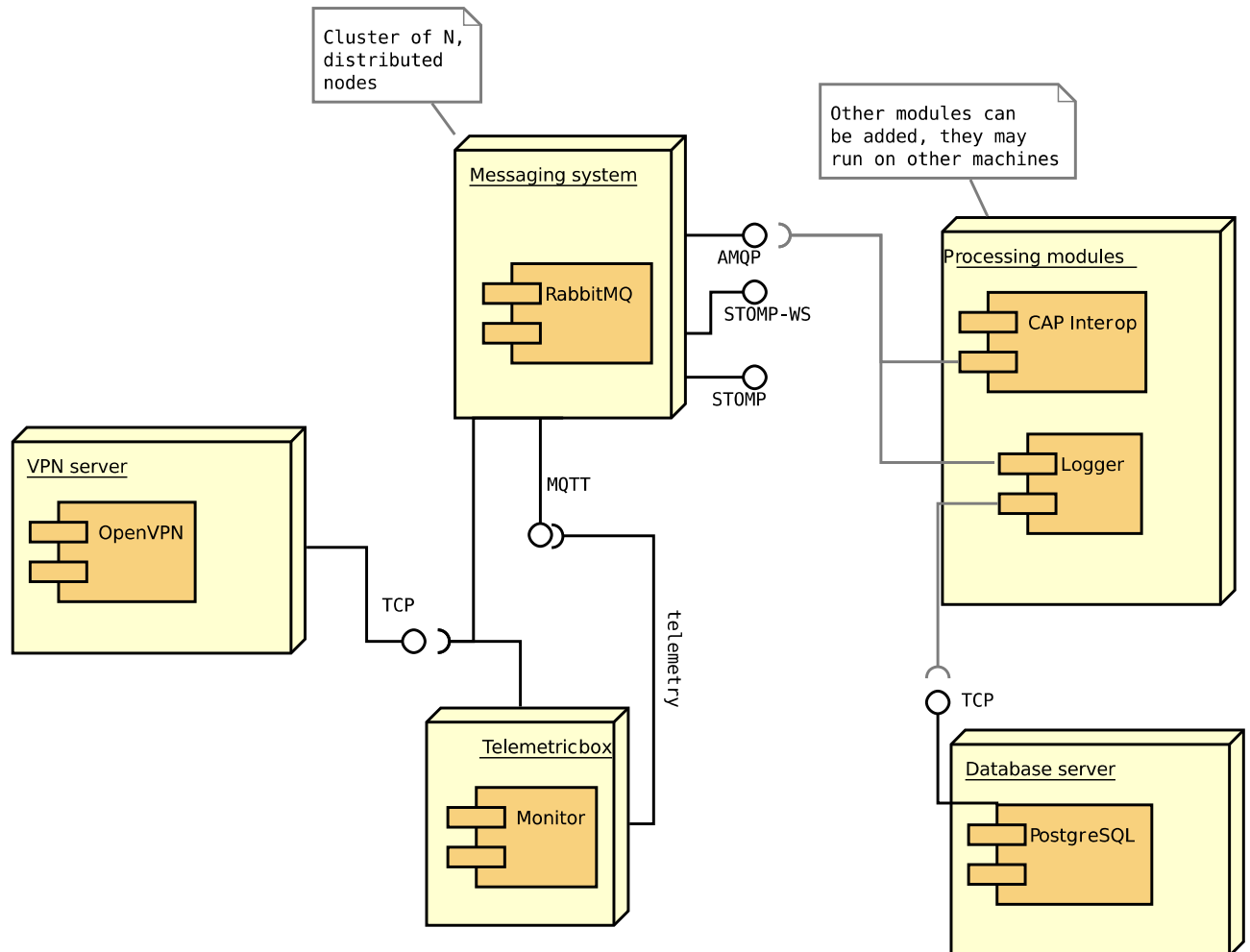


Illustration 6.2: Deployment diagram

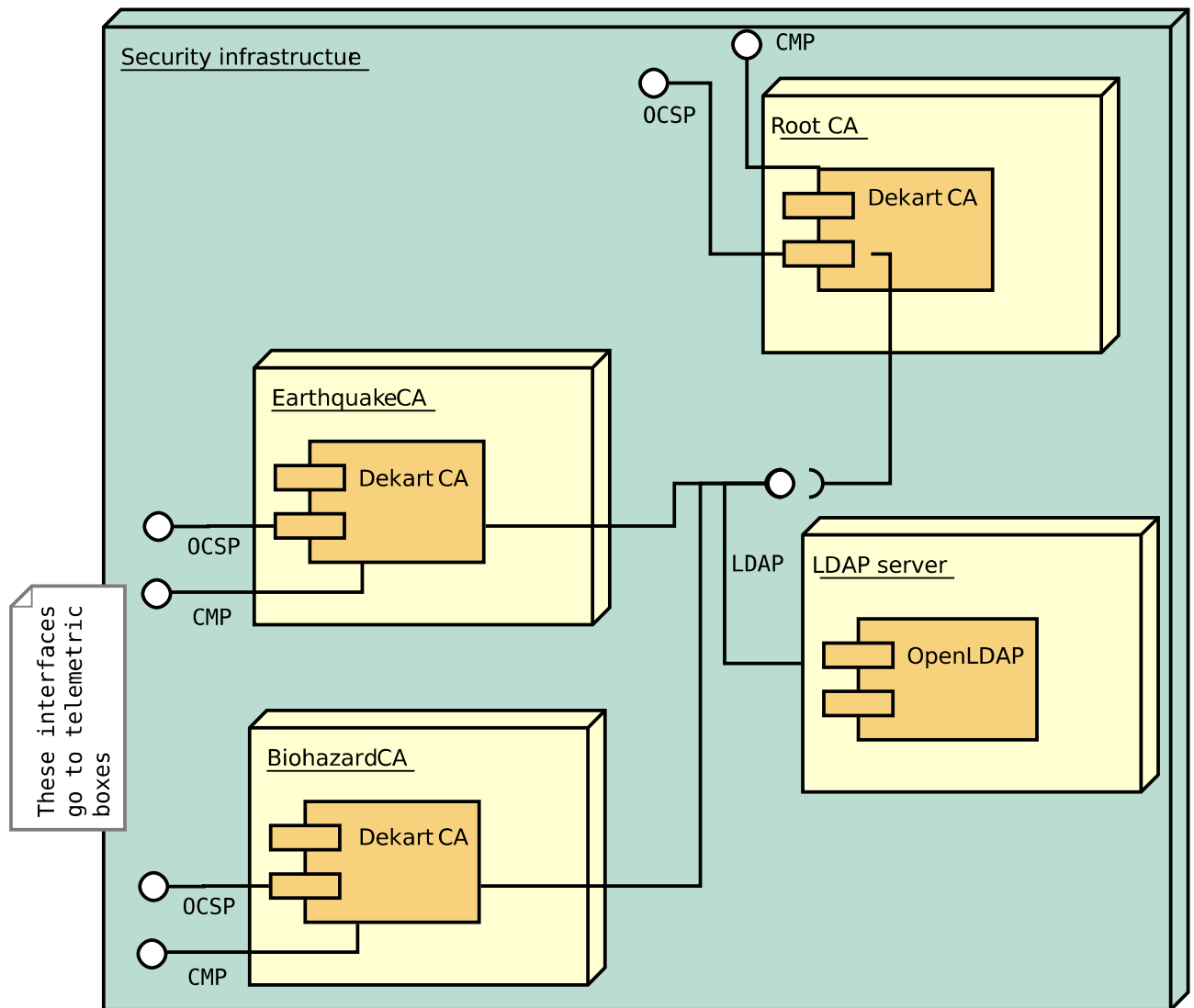


Illustration 6.3: PKI infrastructure deployment diagram

7 Ethical considerations

7.1 Security

The system designed in the context of this paper is suitable for use in industrial settings, and there is a possibility that one day it will be utilized as a component of critical infrastructure. This turns it into an attractive target for sophisticated attackers. Examples of such attacks are Stuxnet⁸² in 2010, or an incident at a German steel factory in late 2014⁸³. Consequences of such attacks vary, in the worst case they can lead to loss of life, while a milder version of that would be material damage of expensive equipment and service disruptions.

Without a doubt, the right question to ask is not “will it be targeted?”, but rather “when will it be targeted?” or “what is the best way to react when the system is under attack?”.

One of the prerequisites for trusting a system is *transparency* – i.e. the possibility to examine its internals and understand how it works. This is doubly important in the context of security, where it is imperative for third-party security experts to be able to analyze a system and look for weak spots or intentionally placed backdoors. This point is further emphasized by Kerckhoffs' law⁸⁴, which states that a secure system continues to be so even everything about it is known to an enemy, except the secret element (a key or a password).

Thus, it is mandatory to *fully disclose the security architecture* and principles of the system to those who will use it.

7.2 Corruption

Another aspect that has to be considered is money laundering opportunities that arise when complex software systems are acquired in developing countries such as Moldova. There are a handful of examples that illustrate that – a health-care system⁸⁵ that cost 1 million Euro and turned out to be just a set of tweaks of an earlier product; or a multiple vote fraud prevention system that cost approximately 1.7 million Euro and failed in the first minutes when used in an actual election (to add more damage, it was later touted as an absolute success⁸⁶ by the media).

A solution that takes away the possibility of money laundering, is to make the software and its accompanying documentation *free of charge*. It is imperative that for information to be widely

disseminated, to make the public aware of the fact that it cost nothing, thus any attempt to pay for it ought to be treated with suspicion. Although this does not completely eliminate any dishonest transactions, it makes it more challenging for corrupt officials to ensure their schemes pass unnoticed.

7.3 Vendor lock-in

This is a practice employed by some businesses to indirectly force a consumer acquire more products or services from the same company. Such results are achieved via proprietary software or hardware interfaces, file formats, network protocols or expendables (e.g. cartridges, special memory cards, chargers, etc). The most obvious consequence is that a consumer might end up paying more to continue receiving updates; but there is a much more serious effect too. If the vendor goes bankrupt or simply decides to discontinue the product because it is not financially attractive to them anymore – the consumer hits a technological dead end.

This can become a grave issue for critical infrastructure, because such a strategically important component will eventually become vulnerable to new threats that were not around when the system was conceived.

Vendor lock-in can be avoided by ensuring that the consumer has *access to the source code* of the software and is free to change it as they see fit, without being subjected to any restrictions.

7.4 Sharing and cooperative software evolution

If time shows that the software successfully handles notifications for critical infrastructures, it would become attractive to other players who are facing the same challenges. It would be noble to share it with them, potentially exerting a positive influence on the quality of life in different regions of the planet, even in the distant future. This point is made by Joseph Weizenbaum in [3]:

“...the scientist and engineer, has responsibilities that transcend his immediate situation, that in fact extend directly to future generations. These responsibilities are especially grave, since future generations cannot advocate their own cause now. We are their trustees.”

This can be further enhanced by agreeing beforehand that any improvements others make to the software are *contributed back* – thus ensuring that the evolution of the system is powered by everyone who uses it, not just the original creator.

This point is made in the chapter titled “The social structure of cooperation” of Robert Axelrod's

“Evolution of cooperation”[4]. Data produced by computer modeling of societies where players adapt different strategies is available in the “Territoriality” section. The model suggests that a group of entities employing cooperative strategies can eventually overtake an entire population, provided there are enough cooperating individuals. Such models are perceived easier when they are visualized^{87 88} as a function of time.

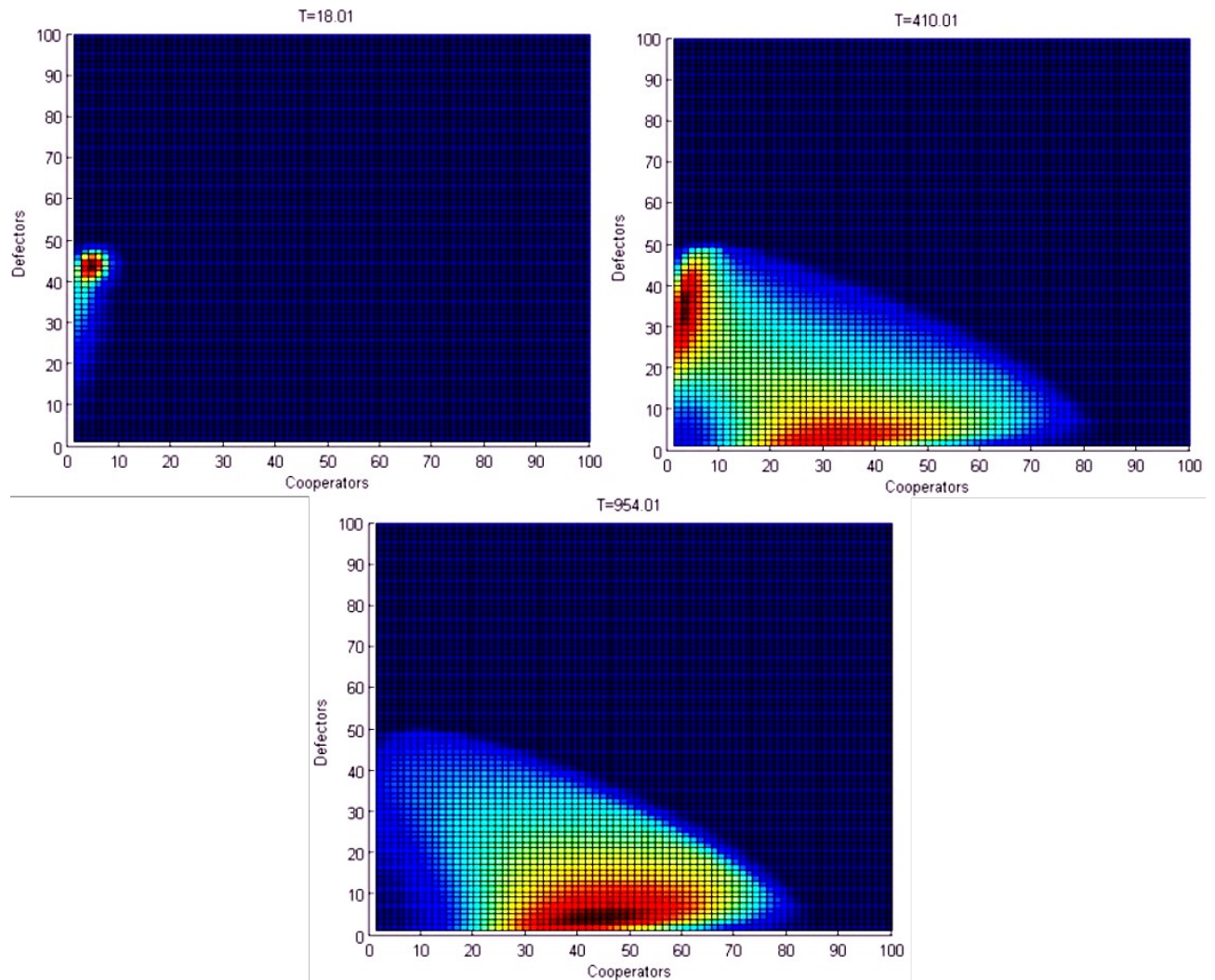


Illustration 7.1: Evolution of cooperation

The images illustrate the dynamics inside a population with a majority of players that mostly *defect* and a minority of “nice” strategies. After a number of iterations the population tends to include agents that mostly *cooperate*. A prime example of a cooperative strategy is “Tit for tat”, devised by Anatol Rapoport [5], which ended up as the winner in two tournaments of the iterated prisoners' dilemma, organized in 1980.

Thus it is reasonable to assume that cooperation via code sharing will eventually pay off, the benefits being represented by disclosure of found defects and software updates; or other, non-technological types of remuneration. Such forms of reciprocal altruism have been observed in flora, the animal kingdom as well as in sports and even warfare, being documented in “Nice guys finish first”⁸⁹, by Richard Dawkins [6]. Cooperative strategies are stable, i.e. they perpetuate in time and can withstand intrusions of rogue players.

The product of this paper itself testifies to the fact that cooperation is an effective way to get things done. It leverages a number of free/libre software components and it would not have been possible to achieve these results if these modules were not readily available.

7.5 Public perception

Aforementioned failures (see Corruption above) were harshly criticized⁹⁰ by the press and by the software engineering community throughout social media, thus reinforcing the “public institutions are incompetent” adage.

A project is more likely to gain traction and develop a positive image (even when not entirely successful) if the entire development cycle is transparent. A person has a more respectful attitude towards an assignment when they understand the rationale behind it and when they know they can get involved and make a meaningful contribution [7]. This brings several benefits:

- people are less likely to criticize a system after its release, because they know they could have contributed constructively at an earlier stage;
- all players involved treat the situation as a learning experience, rather than an opportunity to shift blame or point fingers at each other.

7.6 Conclusions

Strategically, the top priorities are security, access to the source code and avoiding vendor lock-in. While cooperation and sharing are great benefits, they are not mandatory for a successful use of a system.

These requirements can be met by releasing the product under a free/libre⁹¹ license, the primary candidates being BSD (Berkeley Software Distribution) and GPL (GNU Public License). The former

is more permissive, because it leaves one with the opportunity to modify software and keep it to themselves, while the GPL elicits reciprocity and mandates the sharing of modified versions.

Neither license precludes commercialization, so profits can still be made off the sale or maintenance of such a system.

A successful example of FLOSS used in a public institution is the LiMux (Linux in Munich) project⁹², which so far resulted in savings of about 11.7⁹³ million Euro.

8 Conclusions

The aim of this paper was to research and determine the needs of a fault tolerant, secure, real-time notification system for critical infrastructure. The solution is to leverage existing standards and free/open-source software.

Standardization played an important role in the design, as it enabled us to draw from the experience of other countries and international organizations, and produce a complete solution that seamlessly integrates into other environments. Extra attention was paid to European standards, as they are specifically relevant for Moldova's geo-political context.

Free open-source software covered every niche in the resulting software stack, saving a tremendous amount of time and effort. The help of open source contributors is greatly appreciated.

Multiple levels of reliability and security were proposed. The weight of extremely critical systems must be supported by redundancy and strict security; less important systems can have a relaxed security policy that is less expensive to implement and enforce.

The product of this work is a model that meets the reliability, scalability and security needs of a culture that strives towards greater energy efficiency and less waste. Everyone is encouraged to apply my findings or adjust them to their infrastructure.

9 References

1. RFC 3552, *Guidelines for writing RFC text on security considerations*.
2. SCHNEIER BRUCE, *Secrets and Lies – digital security in a networked world*, Wiley 2000.
3. WEIZENBAUM JOSEPH, *Computer Power and Human Reason*, Penguin Books 1976.
4. AXELROD ROBERT, *The evolution of cooperation*, Penguin Books 1984, p.158 – 168.
5. RAPOPORT ANATOL, *Certainties and doubts*, Black Rose Books 2000, p. 150 – 152.
6. DAWKINS RICHARD, *Selfish gene*, Oxford University Press 2006, p. 202 – 233.
7. KOHN ALFIE, *Punished by rewards*, Houghton Mifflin 1999.

10 Appendix

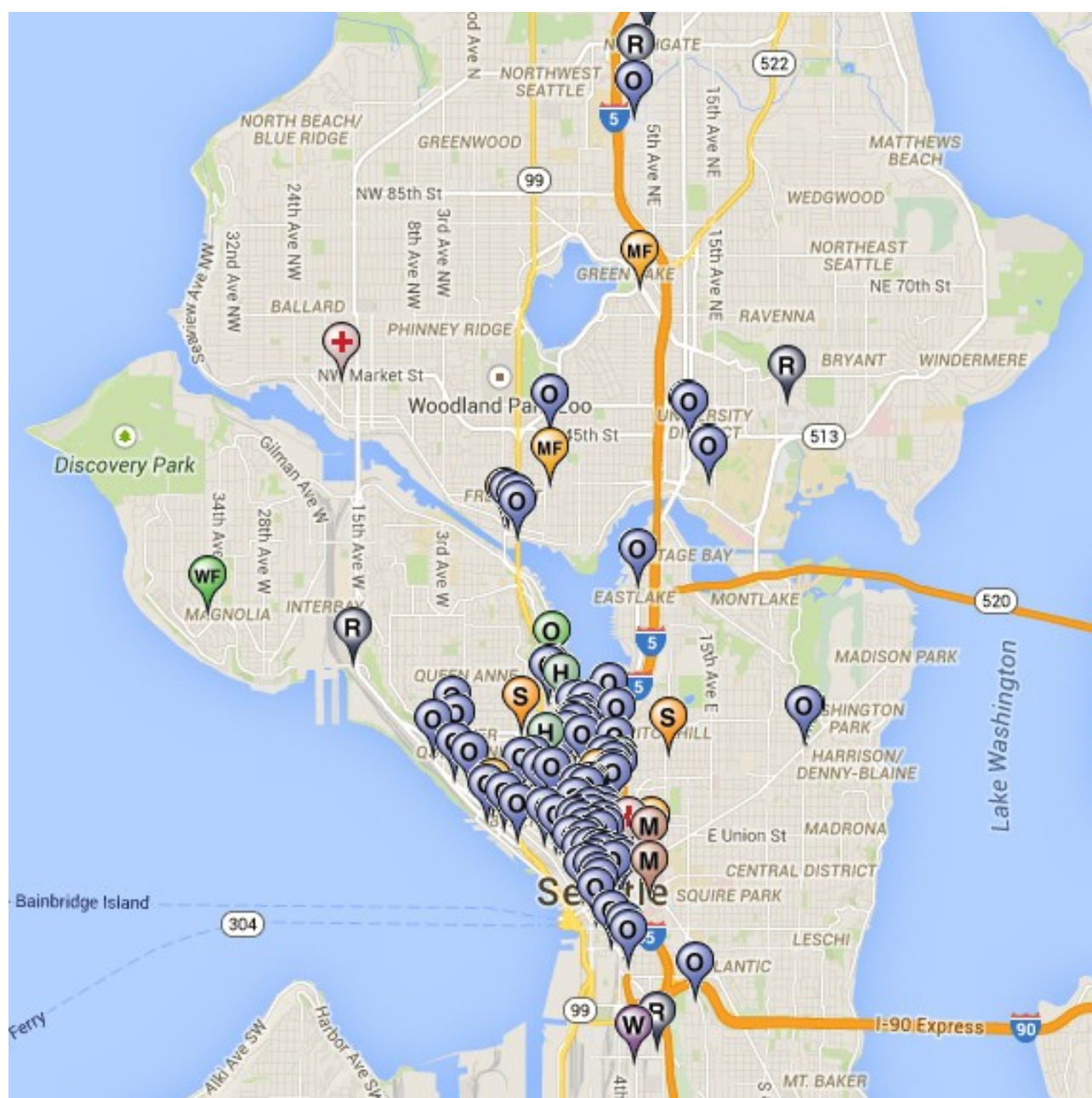


Illustration 10.1: Energy Star compliant buildings in Seattle, 2015.



Illustration 10.2: Internet-facing control system devices

Over 7200 control devices were exposed on the Internet in 2012, according to ICS-CERT⁹⁴.

```

<?xml version = "1.0" encoding = "UTF-8"?>
<alert xmlns="urn:oasis:names:tc:emergency:cap:1.2">
  <identifier>KST01055887203</identifier>
  <sender>KST0@NWS.NOAA.GOV</sender>
  <sent>2003-06-17T14:57:00-07:00</sent>
  <status>Actual</status>
  <msgType>Alert</msgType>
  <scope>Public</scope>
  <info>
    <category>Met</category>
    <event>SEVERE THUNDERSTORM</event>
    <responseType>Shelter</responseType>
    <urgency>Immediate</urgency>
    <severity>Severe</severity>
    <certainty>Observed</certainty>
    <eventCode>
      <valueName>SAME</valueName>
      <value>SVR</value>
    </eventCode>
    <expires>2003-06-17T16:00:00-07:00</expires>
    <senderName>NATIONAL WEATHER SERVICE SACRAMENTO CA</senderName>
    <headline>SEVERE THUNDERSTORM WARNING</headline>
    <description> AT 254 PM PDT...NATIONAL WEATHER SERVICE DOPPLER RADAR INDICATED A SEVERE
THUNDERSTORM OVER SOUTH CENTRAL ALPINE COUNTY...OR ABOUT 18 MILES SOUTHEAST OF KIRKWOOD...MOVING
SOUTHWEST AT 5 MPH. HAIL...INTENSE RAIN AND STRONG DAMAGING WINDS ARE LIKELY WITH THIS
STORM.</description>
    <instruction>TAKE COVER IN A SUBSTANTIAL SHELTER UNTIL THE STORM PASSES.</instruction>
    <contact>BARUFFALDI/JUSKIE</contact>
    <area>
      <areaDesc>EXTREME NORTH CENTRAL TUOLUMNE COUNTY IN CALIFORNIA, EXTREME NORTHEASTERN
CALAVERAS COUNTY IN CALIFORNIA, SOUTHWESTERN ALPINE COUNTY IN CALIFORNIA</areaDesc>
      <polygon>38.47, -120.14 38.34, -119.95 38.52, -119.74 38.62, -119.89 38.47, -
120.14</polygon>
      <geocode>
        <valueName>SAME</valueName>
        <value>006109</value>
      </geocode>
      <geocode>
        <valueName>SAME</valueName>
        <value>006009</value>
      </geocode>
      <geocode>
        <valueName>SAME</valueName>
        <value>006003</value>
      </geocode>
    </area>
  </info>
</alert>




```


Winter Storm Warning in Austin - San Antonio

Active for next 20 hours

Locations: Bexar; Comal; Hays; Kinney; Medina; ... [Show more](#)

Posted 7 hours ago – [National Weather Service](#)

How likely:  How soon:  How severe: 

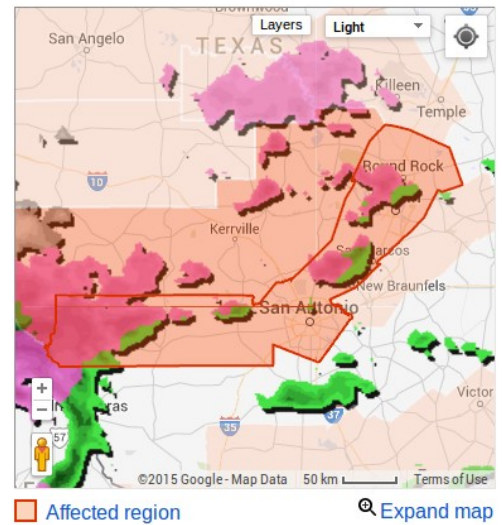
The [original text](#) for this alert has been automatically reformatted to correct capitalization.

Correction to timing bullet in First Group.

Significant ice accumulations are likely over the hill country and Southern Edwards Plateau tonight into Saturday.

Shallow Arctic air over South Central Texas will continue to cool temperatures, with most areas along and north of the Balcones Escarpment expected to reach to near or below freezing by midnight. Areas of patchy light rain and drizzle will increase into rain showers mixed with sleet showers late tonight. Light ice accumulations are already occurring over the hill country, and the early onset of freezing conditions will enable several hours of potential ice accumulations. Most ice accumulations should fall into a range from 1/10 to 1/4 inch. However, Bands of sleet showers could support isolated higher accumulations up to 1/2 inch.

Across the central and northern Hill Country, temperatures may not reach above freezing or long enough to melt the accumulated ice. Thus the threat for icy roads could extend into Saturday evening. Farther south and east, areas from Del Rio to San Antonio to Austin may experience a shorter freezing period, with



Other alerts in this area

[Winter Storm Warning in Austin - San Antonio](#)
National Weather Service - updated 6 hours ago

Illustration 10.3: Detailed view of an alert of Google Public Alerts



This QR code contains a base64-encoded certificate signing request. The use of such graphical codes allows the system to perform its functions without the need to connect it to a network or attach storage devices to it. Having these interfaces available would expose the system to unnecessary risks.

Illustration 10.4: QR code that contains a CSR

This is the information encoded in the image above, it contains an ASN.1 data structure.

```
-----BEGIN CERTIFICATE REQUEST-----
MIIB4TCCAUAoCAQAwgaAxiZAhBgkqhkiG9w0BCQEWFEucmFpbGVhbkbkZwthcnQuY29tMRowGAYD
VQQDExFSYWlsZWFuIEFsZXhhbmRydTEuMC8GA1UECx4oAEGAA5ABSAHAAZABIAHMAawAgBDQENQQ/
BDAEQARCBDAEPAQ1BD0EQjEZMBcGA1UEBx4QAEMAaABpAhkAaQBuAQMAAdTEPMA0GA1UEChMGRGVr
YXJ0MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC8KkjVgNaQz/hRRhMqH07UPf1SS01Vk0IX
SkvWxv2fDXcr+6kZmFhPSbpxvcDLvdT0WeTUm4R9FY/7F7oV4DL3JB68ajrbIayWScb6kLkxSZpP
mg08162xME5bIvZxtUcJxhTofW7AFSB0ote5cgYUFU71G0+hjIMsE5YKpF+DGwIDAQABoAAwDQYJ
KoZIHvcNAQEFBQADgYEA0RqtJIXQ16ih90k0xN78sEqDvfgVZmDGsALfNr4aVMBQiiXs/HMtV1V3
Gtw0Ua8JduESS8FwoEbWNjfbk33NpAXBCbrdjfywNt6nYOQnGfnb9jyCmm2gcjJoDmNz47V9iY
3xaBHgwaJp06WPTFuoX7j3ts3DWysZ0uf+lvzeM=
-----END CERTIFICATE REQUEST-----
```

This is the ASN.1 decoded structure, obtained with OpenSSL by running `openssl req -in`

`req.csr -noout -text`

Certificate Request:

Data:

Version: 0 (0x0)
Subject: emailAddress=a.railean@dekart.com, CN=Railean Alexandru,
OU=\x00H\x00\xE4\x001\x00p\x00d\x00e\x00s\x00k\x00
\x044\x045\x04?\x040\x04@\x04B\x040\x04<\x045\x04=\x04B,
L=\x00C\x00h\x00i\x02\x19\x00i\x00n\x01\x03\x00u, O=Dekart

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

Public-Key: (1024 bit)

Modulus:

00:bc:2a:48:d5:80:d6:90:cf:f8:51:46:13:2a:1f:
4e:d4:3d:fd:52:4b:49:55:90:e2:17:4a:4b:d6:5e:
fd:9f:0d:77:2b:fb:a9:19:98:58:4f:49:ba:71:bd:
c0:cb:bd:d4:f4:59:e4:d4:9b:84:7d:15:8f:fb:17:
ba:15:e0:32:f7:24:1e:bc:6a:3a:db:21:ac:96:49:
c6:fa:90:b2:b1:49:9a:4f:9a:03:bc:97:ad:b1:30:
4e:5b:22:f6:71:b5:47:09:c6:14:e8:7d:6e:c0:15:
20:74:a2:d7:b9:72:06:14:15:4e:f5:1b:4f:a1:8c:
83:2c:13:96:0a:a4:5f:83:1b

Exponent: 65537 (0x10001)

Attributes:

a0:00

Signature Algorithm: sha1WithRSAEncryption

39:1a:ad:24:85:d0:d7:a8:a1:f7:49:0e:c4:de:fc:b0:4a:83:
bd:f8:15:66:60:c6:b0:02:df:36:be:1a:54:c0:50:8a:25:ec:
fc:73:2d:57:55:77:1a:dc:34:51:af:09:76:e1:12:49:2f:05:
5a:81:1b:58:d8:df:6e:4d:f7:36:90:17:04:26:eb:76:37:f2:
c0:db:7a:9d:83:90:aa:71:9f:9d:bf:63:c8:23:26:da:07:23:
26:80:e6:37:3e:3b:57:d8:98:df:16:81:1e:0c:1a:26:93:ba:
58:f4:c5:ba:85:fb:8f:7b:6c:dc:35:b2:49:93:ae:7f:e9:6f:
cd:e3

- [1] http://www.energystar.gov/sites/default/uploads/about/old/files/EnergyStar_POY_4page_040414_PrintReady_508compliant.pdf
- [2] <http://smarcitizen.me>, <http://citysdk.eu>, <http://citiesinmotion.iese.edu/>
- [3] There are 25447 “Energy star” compliant buildings in the USA (January 2015), http://www.energystar.gov/index.cfm?fuseaction=labeled_buildings locator
- [4] <http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.pdf>
- [5] Only in XMLSIG format, use of other formats not endorsed by the specification
- [6] <http://cap-validator.appspot.com/>
- [7] Abstract Syntax Notation One, standardized in X.208
- [8] Due to the fact that an SMS is limited to 160 characters
- [9] <https://support.google.com/publicalerts/?hl=en&gl=US#2769138>
- [10] <http://www.vigilfuoco.it/aspx/Page.aspx?IdPage=4554>
- [11] <http://sahanafoundation.org/products/eden/>
- [12] <https://github.com/flavour/eden/>
- [13] http://lirneasia.net/wp-content/uploads/2008/07/rtpb_cap_user_guide_v1_3.pdf
- [14] <https://github.com/flavour/eden/blob/4ae1744b8a9b3e27c86fd2b36e2e06d895dabfe7/modules/pygsm/autogsmmod em.py>
- [15] <http://jixel.eu>
- [16] Explicit reference removed in v1.2, but present in v1.1
- [17] Protocol used for issuing, suspending, revoking and reviving X509 certificates.
- [18] RFC 2511
- [19] JIS X 0510.
- [20] <http://www.qrcode.com/en/faq.html>
- [21] For example: <http://www.tydenbrooks.com/Products/Electronic-Seals/Hyperion-Zigbee-E-Seal.aspx>
- [22] <http://protect.gost.ru/document.aspx?control=7&id=180151>
- [23] http://jitc.fhu.disa.mil/pki/documents/dod_x509_certificate_policy_v9_0_9_february_2005.pdf
- [24] <http://technet.microsoft.com/en-us/library/cc962064.aspx>
- [25] <http://www.businessinsider.com/twitter-total-registered-users-v-monthly-active-users-2013-11>
- [26] <http://www.statisticbrain.com/twitter-statistics/>
- [27] <http://earthquake.usgs.gov/earthquakes/ted/>
- [28] <https://about.twitter.com/products/alerts>
- [29] <https://blog.twitter.com/2013/introducing-twitter-alerts>
- [30] Defined formally in ETSI TS 100 901, originally GSM 03.40
- [31] <http://www.3gpp.org/DynaReport/0340.htm>
- [32] <http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2014-e.pdf>
- [33] <http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2010.pdf>
- [34] Patrick Traynor, PhD, Georgia Institute of Technology, September 2008
- [35] ETSI TS 100 902, formerly 3GPP 03.41, <http://www.3gpp.org/DynaReport/0341.htm>
- [36] ETSI TS 102 900, European Public Warning System (EU-ALERT)
- [37] <http://www.ietf.org/rfc/rfc821.txt>
- [38] <https://sslearthquake.usgs.gov/ens/>
- [39] <https://tools.ietf.org/html/rfc918>
- [40] <https://tools.ietf.org/html/rfc2177>
- [41] Gmail, Apple, Yahoo
- [42] <http://www.rfc-editor.org/rfc/rfc3369.txt>
- [43] <https://tools.ietf.org/html/rfc4880>
- [44] <http://tools.ietf.org/html/rfc3552#section-6.1.1.8>
- [45] RFC 6120, <https://tools.ietf.org/html/rfc6120>
- [46] <http://slashdot.org/story/99/01/04/1621211/open-real-time-messaging-system>
- [47] <http://xmpp.org/extensions/xep-0323.html>
- [48] <https://www.facebook.com/notes/facebook/facebook-chat-now-available-everywhere/297991732130>
- [49] <http://xmpp.org/2011/06/skype-adds-xmpp-support/>
- [50] Until being phased out by Google Hangouts in 2013, <https://www.eff.org/deeplinks/2013/05/google-abandons-open-standards-instant-messaging>
- [51] <http://www.xmpp.org/rfcs/rfc6120.html#security>
- [52] ISO/IEC 19464, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=64955
- [53] <http://www.rabbitmq.com/resources/specs/amqp0-9-1.pdf>, section 4.10
- [54] <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>
- [55] http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718111
- [56] <http://www.3gpp.org/DynaReport/44012.htm>
- [57] Limits are usually set by specific email providers, though the standard mentions none
- [58] <http://jpmens.net/2013/02/25/lots-of-messages-mqtt-pub-sub-and-the-mosquitto-broker/>
- [59] <http://www.alertacutremur.ro/cum-functioneaza/>
- [60] <http://www.jma.go.jp/jma/en/Activities/eew2.html>

- [61] <http://www.qnx.com/>
- [62] <http://www.windriver.com/products/vxworks/>
- [63] <http://www.freertos.org/>
- [64] <https://www.rtai.org/>
- [65] <http://xenomai.org/>
- [66] <https://en.wikipedia.org/wiki/RTLinux>
- [67] “Moldova's internet revolution: Analyzing the role of technologies in various phases of the confrontation ”, Volodymyr V. Lysenko, Kevin C. Desouza
- [68] <http://gizmodo.com/5964624/how-syria-turned-off-the-internet>
- [69] http://www.nbcnews.com/id/23068571/ns/technology_and_science-internet/t/ships-anchor-caused-cut-internet-cable/
- [70] <http://rtnet.org/>
- [71] Unless exotic functionality, not implemented in RTNet is used
- [72] The data link layer is the second layer in the ISO/OSI network stack
- [73] <http://rosetta.esa.int/>
- [74] http://youtu.be/CR_LBcZg_84?t=11m22s
- [75] <https://www.rabbitmq.com/>
- [76] <http://www.rabbitmq.com/distributed.html>
- [77] <http://www.postgresql.org/>
- [78] <http://www.postgresql.org/about/>
- [79] <https://openvpn.net/>
- [80] <https://digitalid.dekart.com/>
- [81] <http://www.openldap.org/>
- [82] <http://www.symantec.com/connect/blogs/exploring-stuxnet-s-plc-infection-process>
- [83] <http://www.itworld.com/article/2861675/cyberattack-on-german-steel-factory-causes-massive-damage.html>
- [84] http://en.wikipedia.org/wiki/Kerckhoffs%27s_principle
- [85] <http://e-sanatate.md/News/3819/ancheta-softul-ministerului-sanatatii-reparatie-de-1-milion-de-euro-a-unui-soft-deja-creat-din-fonduri-europene-conjuncturi-neclare-si-decizii-dubioase>
- [86] [http://www.timpul.md/articol/\(analiza\)-radiografia-unui-succes-remarcabil-in-r-moldova-un-exemplu-despre-cum-poate-fi-folosita-tehnologia-pentru-impiedicarea-votului-multiplu-67090.html?action=print](http://www.timpul.md/articol/(analiza)-radiografia-unui-succes-remarcabil-in-r-moldova-un-exemplu-despre-cum-poate-fi-folosita-tehnologia-pentru-impiedicarea-votului-multiplu-67090.html?action=print)
- [87] <http://youtu.be/JwJY4RHjeHk>
- [88] Numerical solutions and animations of group selection dynamics; Burton Simon, Aaron Nielsen
- [89] Selfish gene, chapter 12; <https://www.youtube.com/watch?v=I71mjZefg8g>
- [90] <http://on.fb.me/13Sq3ie>, <http://on.fb.me/1xJzWNj>, <http://on.fb.me/17cAhfF>
- [91] <http://railean.net/index.php/2011/11/22/simple-comparison-of-open-source-software-licenses>
- [92] <https://joinup.ec.europa.eu/elibrary/case/linux-it-evolution-open-source-success-story-never>
- [93] <http://www.giraffedog.com/blog/ubuntu-linux-hints-tips/city-munich-successfully-ditches-microsoft-favour-linux-open-source/>
- [94] https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_Oct-Dec2012.pdf