Principles for

Secure Computing and Communication

Rudolf Bayer, Ph.D.
Professor Emeritus, Informatics, Chair for Database Systems
Technical University Munich
January, 2020
Rdlf.Bayer@in.tum.de

Table of content

Principles for 1
Secure Computing and Communication 1
Table of content 2
Preface 3
The Principles 5
Identity 5
cryptID 5
Authentity 5
Authentication 5
Signatures 6
Keys and Key Pairs 6
Computing Processes 6
Agent 6
Login 6
Transactions 6
Contacts 7
Consequences of the SeCC Principles 7
Agents and Certification 7
Encryption 8
Keys and Key Pairs 8
Login 8
Transactions 8
Contacts 8
Implementation of SeCC 8
Identity 8
Agent 9
Encryption 9
Authentication 9
Transactions 9
Contacts 9
Applications 9
Market Acceptance by Islands of SeCC Systems 9
Acknowledgement 10
References 10

Preface

All computing and communication systems in use today are insecure. They are open to failures and attacks. Therefore, they are a serious threat to the technical infrastructure of modern societies and even to the lives of humans. A striking confirmation of this commonplace observation is the size of the global IT security market of 112 billion USD in 2019 [12], desperately trying to patch the endless sequence of security holes.

The Internet of today suffers from the same security problems. Every computer user experiences this daily, is annoyed and frustrated, but can do almost nothing about it. The causes are many birth defects of the Internet like:

1. Openness and easy availability of email and Internet addresses
2. Anonymity of many users (who is who?)
3. No control over the influx of mails and ads
4. Low quality and unproven content
5. No responsibility of authors for their content
6. Zero cost to the end user

Therefore, users are exposed to spammers, fakers, hate messages, unwanted mails, ads, mobbing and even threats. This causes enormous emotional stress and waste of time. A serious estimate of this damage would probably result in several billion USD per day purely financially.
The emotional and psychological damage to people, especially the young, is dramatic. The fathers of computing and the Internet were idealistic scientists, thrilled by their technical achievements, and did not understand, probably not even consider the impact of what they were doing on mankind.

Desirable communication has exactly the opposite properties:

1. I do not want to post the address of my home, not even my business card, in a dictionary visible to everybody
2. I want to be absolutely sure with whom I am communicating
3. I do not want to be bothered with masses of junk
4. I want to receive and to exchange true statements instead of rumors, fakes or even plain lies
5. I want to trust my communication partners and rely on their responsibility for the content they send to me
6. I am willing to pay a modest price for the quality of communication and my privacy

It is fascinating, that the old culture of handwritten letters had exactly these properties.

1. Addresses were only known to a small, closed circle of friends and acquaintances
2. A serious letter carried the handwritten sender on the back, without it the letter was suspicious. Recognition of this handwriting identified and authenticated the sender without doubt.
3. Finding a personal letter in the mailbox was a joy, junk mail was extremely unusual
4. The content of a letter was serious and generally considered as true. Notorious liars were exposed immediately and ostracized
5. The writer of a letter was responsible for the content and could not deny his statements later
6. Even the moderate postage fee prevented massive misuse of letter writing.

As a computer scientist I am far from proposing to go back to handwritten letters. I am simply asking the question: ***Is it possible to keep the virtues of handwritten letters and nevertheless to enjoy the incredible advantages of modern computing, communication and social media without the sacrifices?***

This paper argues that ***the answer is YES!***

Some security measures to ease the problems stated above are available in principal and theoretically already today, but they are inconvenient to install and to use. And even then, they are desperate stopgaps and do not solve the problems principally. Therefore, they are rarely used in ordinary computing or Internet communication.

This paper proposes some very simple principles to avoid these problems and to reform computing and communication in general including the Internet. In addition, these principles are not theoretical proposals, but they have been implemented as prototypes and demonstrated as feasible with moderate effort and without much cost by my student team at the Technical University Munich.

The Principles

We use the term *agent* as a general term for humans, machines, computer programs, smart contracts, micro controllers, etc. This paper argues that *Secure Computing and Communication (SeCC)* can be achieved with only three fundamental principles for a SeCC system:

1. ***Identity***: every agent has a digital identity
2. ***Authenticity***: every agent must be authenticated
3. ***Signatures***: every action of an agent must be digitally signed

To implement these principles and to achieve SeCC, just a few additional basic technical concepts are needed.

Identity

An *identity* is a unique property of an agent like a person or a machine or a computer program. Examples of identities are the DNA, fingerprint, iris image, MAC address of a computing device, hash code of a program, public part of an asymmetric key pair.

cryptID

A *cryptID* κ is a specific identity of an agent in SeCC. κ is generated via a key pair (π,σ) by asymmetric cryptography and consists of the public key π and the signature of π via σ. The cryptID is denoted as $\kappa = [\pi, \sigma(h(\pi))]$ where $h(\pi)$ is the hash value of π and $\sigma(h(\pi))]$ is the signature of π. For more technical details see the various whitepapers in [8].

Note: an identity always belongs to a unique agent, whereas an agent may have several identities. A cryptID is generated only once.

Authentity

We use the artificial term *Authentity* as equivalent to *authenticity*, but in a pure technical sense. The authentity is the unique agent to which the identity belongs.

Authentication

For many applications it is important to know who generated and therefore owns a cryptID. *Authentication* is the process of asserting, that a cryptID and its owner belong together.

There are many ways to establish the authentity of a cryptID κ, e.g.

1. By personally meeting the owner of κ and exchanging the QR code of κ.
2. via a signed selfie video of the owner asserting, that he owns κ
3. By sending a PIN letter to the owner who enters the PIN only once.

4. By checking a certificate issued by the hierarchy of Certification Authorities (CA) [9] for a cryptID and its owner.
5. By mutual certification using the Web of Trust of a public key infrastructure [9]

SECC as implemented at the Technical University Munich (TUM) uses the first three techniques for authentication.

Signatures

Keys and Key Pairs

Symmetric keys are denoted by the greek letter α (often with subscripts), asymmetric key pairs are denoted by the greek $\mathbf{(\pi,\sigma)}$ also with subscripts. Here π is the public key and σ is the private key.
The *signature* of a document **d** via κ is an encryption of the hash of d with the private key: $\mathbf{\sigma(h(d))}$. Here h is a hash function.
A signed document **S(d)** is the pair $\mathbf{[d, \sigma(h(d))]}$. Since the public key π is publicly known, it is easy to verify, that d has not been mutilated, that d was signed and therefore, the authorship of d can never be denied.

Computing Processes

Agent

Agents can compute and communicate with each other by creating transactions, signing them and sending messages, etc. Formally speaking, all agents in a SeCC system have an **identity**, must be **authenticated**, and must **sign** everything they do.

Login

The combination of cryptID and digital signature allows a new login technique, in which the cryptID is used as the login name and a signed random number r as the password: $\mathbf{[\kappa, S(r)]}$

Transactions

Transaction is a technical term for interactions between agents. The standard type of interaction between agent U (with κ_U) and agent V (with κ_V) is a message sent from U to V with a so called payload p. The payload is the content of the message. A transaction has the simple form **t = [U, V, p]** or more precisely $\mathbf{[\kappa_U, \kappa_V, S_U(p)\ ]}$. Transactions are correct and processed only, if both cryptIDs are valid and if the payload is signed by U, ***otherwise the transaction is simply discarded by all agents following the SeCC paradigm*** !
In a future ideal communication infrastructure the transport of such mesages should be refused already by the lowest possible level of the IP stack.

Contacts

Contacts are needed to interact with. Contacts are those cryptIDs (enhanced by additional information) whom you accept as communication partners. Since a cryptID is basically anonymous, the authentication is only an aid for you to accept or to decline a cryptID as a contact. Declined cryptIDs are locally or globally stored in blacklists, and transactions sent by them are immediately discarded. You will never hear from a cryptID in your blacklist again. This clear optIn is a reversal from the optOut method of email systems and browsers.

Consequences of the SeCC Principles

The consequences of the Secc paradigm are dramatic:

Agents and Certification

Since agents are the active parts of a computing system they must behave properly. To assure this, the software they use must be certified against an agreement about a task to be accomplished. Such tasks are often called *smart contracts* in the blockchain world.

It is a common place that testing can show the presence of bugs, but never their absence. Therefore, testing is useless for SeCC.

The endless and annoying series of security patches of all computing systems in use today is striking proof of this observation.

SeCC requires the verification and certification of the SW used. In principle, this statement also holds for HW, but HW is not the topic of this paper.

The following methods of certification can be used depending on the security level to be achieved:

1. Automatic formal verification and certification of a program or a protocol or a SW system
2. Verification and certification of the code of a program by several involved parties and their digital signatures
3. open source of the agent software in the hope that there is broad public agreement (e.g. by the open source community) that the code is correct compared to its specifications
4. certification by one or several generally trusted CAs like BSI up to the level of at least L4 (code inspection) of the common criteria [1]

Note: classical testing techniques are marginally useful at best, since intentional backdoors or misbehavior like data leaks cannot possibly be found by test procedures. Secure and trusted computing absolutely requires code inspection and more. This is particularly relevant for the emerging IoT world with billions of communicating and interacting devices in the consumer and automation markets.

Encryption

Encryption as an option guarantees, that communication remains secret if desired. This is particularly needed for the payload of a transaction, but also for senders and receivers if this is desired or needed by an application.

Keys and Key Pairs

They are the technical basis for the SeCC approach. A secondary consideration is which encryption technique should be used, presently RSA and ECC are prime candidates.

Login

The new login is extremely convenient for the user compared to present methods: The login is generated completely automatically and the user does not have to do or to remember anything, also no passwords. In principle, a single cryptID could replace all present login names and passwords of an agent.

Transactions

They are the fundamental building blocks of all interaction between agents. There is a clear definition of the correctness of a transaction **t = [U, V, p]** and this correctness can be checked automatically and efficiently.

Contacts

The form of contacts proposed makes it very easy to get rid of spam and fake news. Mails and news MUST be signed by the author. Thus spammers and reporters or sources of fake news can be branded on a private blacklist or on public blacklists.
The combination of required signatures, authentication and blacklisting can also prevent hate messages in the Internet, child pornography, etc. Furthermore, authentication eases the discovery and the pursuit of criminals.

Implementation of SeCC

The following comments are very brief. For a reference implementation of the basic principles see the **C-chain** variant of Blockchains [2], [3] and the C-chain applications [4], [5]. You can download them from [8].

Identity

Every agent generates his identity and cryptID the first time it is started, registers it in a public database and stores it in a save place like a key chain or in the trusted platform module (TPM) of a PC or on the hardware security module (HSM) of an IoT device [6].
Properties and data to authenticate this cryptID are also stored in a public database as part of the registration process. These data are signed by the

owner of the cryptID and therefore cannot be faked and can be edited only by him.

Agent

Agents are typically implemented as Apps [4], [5], [7] or as Web services [5], [11] or as smart contracts on micro controllers [6]. Also human agents act in reality indirectly via the Apps on their devices.

Encryption

Any of the classical cryptographic techniques available in public libraries can be used like RSA or ECC. The proper choice depends largely on the requirements of an application [6], [10].
In the IoT world an agent is typically a micro controller or an edge computer. Here ECC is a favorite because of its small footprint and high performance.
HSMs are useful as cryptographic coprocessors with two important advantages:

1. they are available on the market at very low prices because of mass production e.g. for the automotive industry
2. most of them come with a high level of certification e.g. by BSI [6]

Authentication

C-chain uses the following Authentication methods:

1. By personally meeting the owner of κ and exchanging the QR code of κ.
2. via a signed selfie viedo of the owner asserting, that he owns κ
3. By sending a PIN letter to the owner who interst he PIN only once.

Transactions

Transactions by a sender U for a recipient V and with a payload p are simple constructs of the fixed form $\mathbf{[\kappa_U, \kappa_V, S(p)]}$. Structure, size, content and encryption are the responsibility of an application.

Contacts

They are straightforward to implement similar to contact lists of phones and messenger services.

Applications

Applications built on SeCC like [4], [5], [11] are easy to develop [7]. SeCC is integrated into C-chain Apps, who manage identities, encryption and digital signatures completely automatically, the user barely notices it and many tasks like logins and passwords are drastically simplified.

Market Acceptance by Islands of SeCC Systems

Secc systems can be implemented and used in small domains first, e.g. between project teams in a company. Then such islands can be connected to growing networks of domains for secure and trusted cross communication.

Acknowledgement

The SeCC system with several applications (the blockchain variant **C-chain** plus a variety of Apps) has been implemented and tested by a team of Informatics students at the Technical University Munich (TUM).

My thanks go to my students Laurenz Baumgart, Valentin Bootz, Kilian Eisert, Johannes Ismair, Moritz Kellermann, Stefan Löhnert, Lisa Lörinci, Stefan Madzharov, Karl Mattes, Jonas Mayer, Piero Schlandt, Tim Schmidt, Christian Thieme, Benedikt Thoma, Paul Wieland, Nouman Zeb.

References

Some of the following references can be found on
https://db.in.tum.de/research/projects/C-chain/?lang=en Project C-chain

1. Wikipedia: Common Criteria, https://en.wikipedia.org/wiki/Common_Criteria
2. https://db.in.tum.de/research/projects/C-chain/C-chain-Scient.pdf?lang=en
3. https://db.in.tum.de/research/projects/C-chain/C-chain-Highlights.pdf?lang=en
4. App C-chain-v25.apk
5. App buergerakte_2_6.apk
6. K. Eisert: Bachelor Thesis
7. J. Ismair: Master Thesis
8. Dropbox: https://www.dropbox.com/h
9. https://de.wikipedia.org/wiki/Public-Key-Infrastruktur
10. M. Kellermann: Master Thesis
11. P. Wieland: Bachelor Thesis
12. https://www.fortunebusinessinsights.com/industry-reports/cyber-security-market-101165