Fachhochschule Aachen

Campus Jülich

Fachbereich: Medizintechnik und Technomathematik

Studiengang: Technomathematik

Secure Multi-Party Computation for Decentralized Distributed Systems

Masterarbeit von Frederic Klein

Diese Arbeit wurde betreut von:

1. Prüfer: Prof. Dr. rer. nat. Alexander Voß
2. Prüfer: Dr. Stephan Jonas

Aachen, Dezember, 2016

Diese Arbeit ist von mir selbständig angefertigt und verfasst. Es sind keine anderen als die angegebenen Quellen und Hilfsmittel benutzt worden.

Frederic Klein .....
Unterschrift

Abstract

In recent years gamification has become a part in many areas of our daily routine. In regard to our personal life, companies like Amazon or Runtastic can base their gamification approach on publicly sharing personal achievements and statistics to improve user commitment. In contrast, gamification concerning our work life has to satisfy much higher privacy demands. Since comparison is a key component for gamification, privacy protecting computations of system wide statistical values (for example minimum and maximum) are needed. The solution comes in the form of secure multi-party computation (SMPC), a subfield of cryptography. Existing frameworks for SMPC utilize the Internet Protocol, though access to the Internet or even a local area network (LAN) cannot be provided in all environments. Facilities with sensible measuring systems, e.g. medical devices in hospitals, often avoid Wi-Fi to reduce the risk of electromagnetic interference. To be able to utilize SMPC in environments with Wi-Fi restrictions, this thesis studies the characteristics of mobile ad hoc network (MANET) and proposes the design of a SMPC framework for MANET, especially based on Bluetooth technology, and the implementation as a C library.

Since MANETs have a high probability for network partition, a centralized architecture for the computation and data preservation is unfavorable. Therefor a blockchain based distributed database is implemented in the framework. Typical problems of distributed systems are addressed with the implementation of algorithms for clock synchronization and coordinator election as well as protocols for the detection of computation partners and data distribution. Since the framework aims to provide distributed computations of comparable values, protocols for secure addition and secure comparison are implemented, enabling the computation of minimum, maximum and average.

Devices of diverse computational power will be used to verify the applicability for wearables and Internet of Things (IoT) grade devices. Also field-tests with a smart phone ad hoc network (SPAN)(20-50 nodes) will be conducted to evaluated real life use cases. In contrast, the security of the framework and attack scenarios will be discussed. In summary, this thesis proposes a framework for SMPC for decentralized, distributed systems.

Contents

1 Introduction **1**

1.1 Case Study: "The Hygiene Games" 2

2 Background **3**

2.1 Secure Multi-Party Computation 3

2.2 Mobile Ad Hoc Networks 15

3 Design **19**

3.1 Requirements 19

3.2 Decentralized, Distributed Computing 24

3.3 Architecture 35

4 Implementation **37**

4.1 Development Tools 37

4.2 Module Structure 37

4.3 Interfacing the Library 38

5 Evaluation **39**

5.1 Testing Tools 39

5.2 Examination of Computation Time Dependent on Computing Power . . . 39

5.3 Examination of Computation Time Dependent on Number of Participants 39

6 Discussion **40**

7 Conclusion **41**

References **42**

List of Figures

2.1 Simple secure sum protocol for ring 7
2.2 Existing SMPC software grouped by properties 15

3.1 Unified Modeling Language (UML) use-case diagram for the general functional requirements of a node 20
3.2 UML use-case diagram for the functional requirements for the coordinator 21
3.3 UML use case diagram for developer 23
3.4 UML activity diagram for exponential backoff algorithm 25
3.5 Formation of fully meshed computation group 27
3.6 UML sequence diagram for passing of communication token t 28
3.7 Round Trip Time 29
3.8 Example computation of adjustments with Berkeley 29
3.9 Avoidance of false non-termination detection through heartbeat messages 31
3.10 Database synchronization scheme 33
3.11 Securing communication with Rivest, Shamir and Adleman (RSA) and Advanced Encryption Standard (AES) 34
3.12 UML component diagram 36

List of Tables

2.1 Binary representation of secrets s_i 11
2.2 Randomized binary representation of secrets 11

2.3 2^{nd} round 12
2.4 3^{rd} round 12
2.5 Negation of binary representation for minimum determination 13

3.1 Functional requirements 22
3.2 Non-functional requirements 23

List of Acronyms

2PC secure two-party computation.

AES Advanced Encryption Standard.

API application programming interface.

DDoS Distributed Denial of Service.

GSM Global System for Mobile Communications.

HTTPS HTTP over Transport Layer Security (TLS).

IoT Internet of Things.

L2CAP Logical Link Control and Adaptation Protocol.

LAN local area network.

LSB least significant bit.

MAC media access control.

MANET mobile ad hoc network.

MSB most significant bit.

NDK Native Development Kit.

OS operating system.

RFCOMM radio frequency communication.

RSA Rivest, Shamir and Adleman.

RTT Round Trip Time.

SDK software development kit.

SMPC secure multi-party computation.

SPAN smart phone ad hoc network.

TLS Transport Layer Security.

UML Unified Modeling Language.

UTC Coordinated Universal Time.

Chapter 1

Introduction

In the last couple of years gamification has found it's way into many areas of our daily life. In regard to our personal life, companies like Amazon or Runtastic can base their gamification approach on publicly sharing personal achievements and statistics to improve user commitment. In contrast, gamification concerning our work life can have much higher privacy demands. Since comparison is a key component for the gamification approach, privacy protecting computations of system wide statistical values (for example minimum and maximum) are needed. The solution comes in the form of SMPC, a subfield of cryptography.

Existing frameworks for SMPC utilize the Internet protocol, though access to the Internet or even a LAN cannot be provided in all environments. Especially many hospitals tend to avoid Wi-Fi to reduce the risk of electromagnetic interference with medical devices.

To be able to utilize SMPC in environments with Wi-Fi restrictions, this thesis studies the characteristics of mesh-networks and proposes describes the design of a SMPC framework for mesh-networks.

Context

Restatement of the problem

Restatement of the response

Roadmap

1.1 Case Study: "The Hygiene Games"

Gamification

Wireless Networks in Hospitals

Chapter 2

Background

In this chapter a general understanding of SMPC and the key features of MANETs is established.

First the idea for SMPC is introduced in 2.1 Secure Multi-Party Computation. Since secret sharing is used for the development of SMPC protocols, Shamir's secret sharing scheme is presented in 2.1.1 Secret Sharing. Protocols for secure addition and secure comparison with passive security are introduced in 2.1.2 and 2.1.3 and existing frameworks for SMPC are briefly discussed in 2.1.4.

To be able to define requirements for the new framework (see 3.1), the key features of MANETs are identified in 2.2 Mobile Ad Hoc Networks, with a focus on the wireless technology standards Bluetooth and Wi-Fi and the differences to similar network types like mesh networks.

2.1 Secure Multi-Party Computation

SMPC is a subfield of cryptography. The target of SMPC is to run computations over inputs from multiple parties while keeping these inputs secret. In 1982 Yao described the problem of two millionaires trying to find out, which one is wealthier, without giving each other information about their actual capital (Yao 1982). Yaos solution for this secure two-party computation (2PC) is considered to be the basis for general SMPC protocols. Cramer, Damgård, and Nielsen (2015) describe for example benchmark analysis as a use-cases for SMPC: companies want to know how well they are doing in their business area compared to other companies, while they do not want to share their current busi-

ness numbers with competitors. Using a protocol for secure comparison (as described in 2.1.3 Secure Comparison Protocol) the companies can calculate the best performer without leaking business information. Clifton et al. (2002) describe privacy preserving data mining as another use-case: data mining on patient data can for example be used to indicate disease outbreaks but there is of course a privacy concern. Using SMPC algorithms, statistics can be computed while keeping the personal patient data private.

For SMPC two types of adversaries have to be considered: semi-honest and malicious adversaries. Semi-honest adversaries "follow the protocol specification, yet may attempt to learn additional information by analyzing the transcript of messages received during the execution" (Aumann and Lindell 2007). Malicious adversaries "are not bound in any way to following the instructions of the specified protocol" (Aumann and Lindell 2007). SMPC protocols that can tolerate semi-honest parties (up to a specific threshold) provide semi-honest or passive security. SMPC protocols that are secure against malicious adversaries achieve malicious or active security. Cramer, Damgård, and Nielsen (2015, p. 82) also differentiate between unconditional or perfect security and computational security: if security can be proven for an adversary with unlimited computation power a protocol has unconditional security. In contrast, computational security can only be proven for a polytime adversary.

Since the target group for the protocols used in this thesis are gamification systems potential adversaries are likely of the semi-honest type. Gamification system are usually based on intrinsic motivation. Especially in the context of workplace related gamification without public recognition, there is nothing to be gained from trying to corrupt the system, only the significance of the computation results is reduced.

Honest, but curious parties are more likely, but providing the majority of semi-honest parties (which is the requirement for gaining additional information from combined shares, see 2.1.1), requires considerable efforts. Even if single scores are revealed, their isolated information content is almost valueless for the adversaries and targeting specific nodes over a longer amount of time adds additional complexity because of the spatial degree of freedom of the nodes (compare 2.2). Therefor, in context of gamification systems, this thesis focuses on practical SMPC protocols for passive security based on secret sharing.

2.1.1 Secret Sharing

Cramer, Damgård, and Nielsen (2015, p. 32) describe secret sharing schemes as the main tool to build a SMPC protocol with passive security. In 1979 Adi Shamir described a (k, n) threshold scheme for sharing secret data D: "Our goal is to divide D into n pieces D_i, ..., D_n in such a way that: (1) knowledge of any k or more D_i pieces makes D easily computable; (2) knowledge of any $k-1$ or fewer D_i pieces leaves D completely undetermined (in the sense that all its possible values are equally likely)" (Shamir 1979). Shamir's secret sharing scheme is based on polynomials of degree $k-1$ with $a_0 = D$ (compare 2.1).

$$q(x) = \underbrace{D}_{a_0} + a_1 \cdot x + ... + a_{k-1} \cdot x^{k-1} \tag{2.1}$$

To divide D into n pieces the polynomial is evaluated: $D_i = q(i),\ i = 1, ..., n$. For cryptographic protocols it is not practical to work with real arithmetic, instead a finite field is used: Shamir (1979) specifies that modular instead of real arithmetic is used. A prime p with $p > D$, $p > n$ is selected and used to define the set $[0, p)$. "The coefficients $a_1, ..., a_{k-1}$ in $q(x)$ are randomly chosen from a uniform distribution over the integers in $[0, p)$, and the values $D_1, ..., D_n$ are computed modulo p" (Shamir 1979, p. 613) (compare 2.2).

$$q(x) = D + a_1 \cdot x + ... + a_{k-1} \cdot x^{k-1} \mod p \qquad D,\ a_i \in [0, p)\ , \quad p \in \mathbb{P} \tag{2.2}$$

Cramer, Damgård, and Nielsen (2015, p. 7) declare the set restricted by p as $\mathbb{Z}_p = \{0, 1, ..., p-1\}$. They also use the notion *secret* S for the data to be shared and *shares* s_i for the computed pieces of the secret.

The reconstruction of a secret S can be done using Lagrange interpolation (compare 2.3).

$$S = \sum_i s_i \prod_{i \neq j} \frac{-x_j}{x_i - x_j} \mod p \tag{2.3}$$

k shares s_i are needed to reconstruct S, so only the associated values for i are used in the Lagrange interpolation.

Example Computation

Consider the following task: a secret $S = 8$ is supposed to be shared among $n = 4$ parties $P_i, \quad i = 1, ..., 4$. The threshold for the number of needed shares for the reconstruction of the secret shall be $k = 3$ (public).

First a prime p has to be chosen, which has to be larger than the secret ($p > S$) and the number of parties ($p > n$): $p = 17$ (public information)

Since $k = 3$, the polynomial has a degree of $k - 1 = 2$ (compare 2.4).

$$f(x) = S + a_1 \cdot x + a_2 \cdot x^2 \mod p \tag{2.4}$$

The coefficients are selected randomly uniformly out of $\mathbb{Z}_p = \{0, 1, ..., p-1\} = \{0, 1, ..., 16\}$: $a_1 = 13$ and $a_2 = 4$ and the shares s_i are computed (compare 2.5).

$$\begin{aligned} f(x) &= 8 + 13 \cdot x + 4 \cdot x^2 \mod 17 \qquad (2.5)\\ &\Downarrow \\ f(x_1) &= f(1) = 25 \mod 17 = 8 = s_1 \\ f(x_2) &= f(2) = 50 \mod 17 = 16 = s_2 \\ f(x_3) &= f(3) = 83 \mod 17 = 15 = s_3 \\ f(x_4) &= f(4) = 124 \mod 17 = 5 = s_4 \end{aligned}$$

If for example parties P_2, P_3 and P_4 pool their shares, they can reconstruct the secret S using Lagrange interpolation (using also the public information: $p = 17$):

$$S = \sum_i s_i \prod_{i \neq j} \frac{-x_j}{x_i - x_j} \mod 17 \qquad \textit{with } i, j \in \{2, 3, 4\} \tag{2.6}$$

$$\begin{aligned} &= s_2 \cdot \frac{-x_3}{x_2 - x_3} \cdot \frac{-x_4}{x_2 - x_4} + s_3 \cdot \frac{-x_2}{x_3 - x_2} \cdot \frac{-x_4}{x_3 - x_4} + s_4 \cdot \frac{-x_2}{x_4 - x_2} \cdot \frac{-x_3}{x_4 - x_3} \mod 17 \\ &= 16 \cdot \frac{-3}{2-3} \cdot \frac{-4}{2-4} + 15 \cdot \frac{-2}{3-2} \cdot \frac{-4}{3-4} + 5 \cdot \frac{-2}{4-2} \cdot \frac{-3}{4-3} \mod 17 \\ &= 96 - 120 + 15 \mod 17 \\ &= -9 \mod 17 \qquad (2.7) \\ &= 8 \end{aligned}$$

Note: in cryptography $a \mod n$ for $a < 0$ (negative dividend) is calculated by adding a

multiple of n ($mn \mod n = 0$), so that $m \cdot n + a > 0$: e.g. $-9 \mod 17 = (\underbrace{1 \cdot 17 - 9}_{>0}) \mod 17$ (compare 2.7), which resolves to: $a \mod n = n - (|a| \mod n), a < 0$.

2.1.2 Secure Addition Protocol

For an environment with honest parties there are simple SMPC protocols to compute the sum over shares. Clifton et al. (2002) describe a ring based method, where the initializing party adds a random number R to the secret input s_1 before passing it to the next node. Each node then adds its secret until the first party receives the result. By removing R the party can than reconstruct the sum over all secret inputs (see figure 2.1).

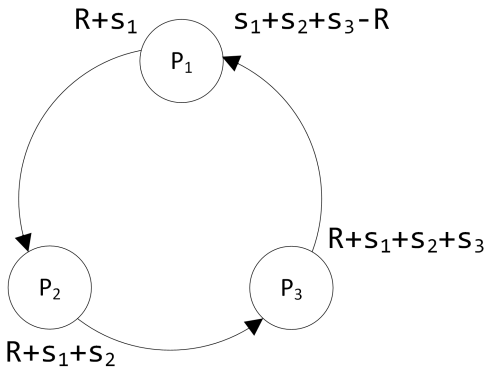


Figure 2.1: Simple secure sum protocol for ring

This method is efficient ($2n$ messages for computation and announcing the sum in a n-node ring) but if parties collude, party P_i only needs the output of P_{i+1} as received by party P_{i+2} to reconstruct the secret input of P_{i+1}. Clifton et al. (2002) propose using shares in combination with permutation of the ring order, so neighbors change in each iteration and the number of parties in need to pool their data increases. This approach was extended in the "k-Secure Sum Protocol" (Sheikh, Kumar, and Mishra 2009). Especially with a focus on security ($k \to n$) the permutation of the ring approaches share-exchanges between each node. To reduce the complexity through the ring permutation and motivated by the restrictions of the network (see 2.2.2), for which the protocol is intended, this thesis uses a Shamir based protocol for a fully connected mesh network.

In 2.1.1 it was demonstrated how a secret can be reconstructed from the shares using Lagrange interpolation. It is also possible to reconstruct the sum of secrets by using the sums of shares for a Lagrange interpolation.

Proof:

n shares for m secrets s_l:

$$s_{l,i} = f_l(x_i) = s_l + \sum_{i=1}^{k-1} \alpha_{l,i} x_i^i \mod p \tag{2.8}$$

$$\Leftrightarrow \begin{cases} s_{1,i} = f_1(x_i) = & s_1 + \alpha_{1,1} x_i + \alpha_{1,2} x_i^2 + ... + \alpha_{1,k-1} x_i^{k-1} \mod p \\ & \vdots \\ s_{m,i} = f_n(x_i) = & s_n + \beta_{n,1} x_i + \beta_{n,2} x_i^2 + ... + \beta_{n,k-1} x_i^{k-1} \mod p \end{cases}$$

$$\text{with } \{l \in \mathbb{N} \mid 1 \leq l \leq m\},\ \{i \in \mathbb{N} \mid 1 \leq i \leq n\},\ \{p \in \mathbb{P} \mid p > \sum_l s_l\},$$

$$\{\alpha \in \mathbb{N} \mid 0 \leq \alpha \leq p\},\ \{k \in \mathbb{N} \mid 2 < k \leq n\}$$

Lagrange-interpolation for secret s_l:

$$s_l = \sum_{i=1}^{n} s_{l,i} \prod_{i \neq j} \frac{-x_j}{x_i - x_j} \mod p \tag{2.9}$$

Sum s over secrets s_l:

$$s = \sum_{l=1}^{m} s_l \overset{\text{with 2.9}}{=} \sum_{l=1}^{m} \sum_{i=1}^{n} s_{l,i} \prod_{i \neq j} \frac{-x_j}{x_i - x_j} \mod p \tag{2.10}$$

$$\text{with} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} = \sum_{j=1}^{m} \sum_{i=1}^{n} a_{ij} \quad \text{follows for 2.10}$$

$$s = \underbrace{\sum_{i=1}^{n} \underbrace{\sum_{l=1}^{m} s_{l,i}}_{\text{sum over shares}} \prod_{i \neq j} \frac{-x_j}{x_i - x_j}}_{\text{Lagrange-interpolation for sum over shares}} \mod p \tag{2.11}$$

Example Computation

Public information: $n = 4$, $p = 67$, $k = 4$

Secrets: $s_1 = 13$, $s_2 = 27$, $s_3 = 17$, $s_4 = 1$

Target computation: sum s over secrets $s = \sum_{i=1}^{4} s_i = 58$ without revealing ones secret

to another party.

$$s_{1,i} = f_1(x_i) = 13 + 35x + 22x^2 + 7x^3 \mod 67 \tag{2.12}$$

$$s_{2,i} = f_2(x_i) = 27 + 3x + 19x^2 \mod 67 \tag{2.13}$$

$$s_{3,i} = f_3(x_i) = 17 + 9x^2 + 27x^3 \mod 67 \tag{2.14}$$

$$s_{4,i} = f_4(x_i) = 1 + 13x + 31x^2 + 40x^3 \mod 67 \tag{2.15}$$

with $x_1 = 1$, $x_2 = 2$, $x_3 = 3$, $x_4 = 4$ follows

$$\begin{aligned}
&\overset{2.12}{\Rightarrow} s_{1,1} = 10 && s_{1,2} = 26 && s_{1,3} = 36 && s_{1,4} = 15 \\
&\overset{2.13}{\Rightarrow} s_{2,1} = 49 && s_{2,2} = 42 && s_{2,3} = 6 && s_{2,4} = 8 \\
&\overset{2.14}{\Rightarrow} s_{3,1} = 53 && s_{3,2} = 1 && s_{3,3} = 23 && s_{3,4} = 13 \\
&\overset{2.15}{\Rightarrow} s_{4,1} = 18 && s_{4,2} = 2 && s_{4,3} = 59 && s_{4,4} = 27 \\
&\Rightarrow \sum_l s_{l,1} = 130 && \sum_l s_{l,2} = 71 && \sum_l s_{l,3} = 124 && \sum_l s_{l,4} = 63
\end{aligned}$$

Lagrange-interpolation:

$$\begin{aligned}
s =& \sum_{i=1}^{4} \sum_{l=1}^{4} s_{l,i} \prod_{i \neq j} \frac{-x_j}{x_i - x_j} \mod 67 \\
=& 130 \frac{-2}{1-2}\frac{-3}{1-3}\frac{-4}{1-4} + 71 \frac{-1}{2-1}\frac{-3}{2-3}\frac{-4}{2-4} \\
&+ 124 \frac{-1}{3-1}\frac{-2}{3-2}\frac{-4}{3-4} + 63 \frac{-1}{4-1}\frac{-2}{4-2}\frac{-3}{4-3} \mod 67 \\
=& 527 \mod 67 = 58 = \sum_{i=1}^{4} s_i
\end{aligned} \tag{2.16}$$

As expected, the result of the Lagrange-interpolation for the sum over shares is equal to the sum over the initial secrets (compare 2.16).

Protocol Description

Assumptions:

- number of parties $n > 2$
- secure communication channel

- no malicious adversaries
- upper bound of sum $s \leq b$ can be estimated, so a prime $p > b$ can be chosen

The secure addition protocol, as used in this thesis, consists of six phases:

1. The coordinator announces the number of parties for the computation and the indexation of each party.
2. Each party j sends shares $s_{j,i}$ of the secret input s_j to the other parties.
3. Each party i computes the sum over the received shares $s_{j,i}$.
4. Each party sends the computed sum to the coordinator.
5. The coordinator reconstructs the sum over the inputs using Lagrange-interpolation.
6. The coordinator broadcasts the reconstructed sum.

In total $(n+3)\cdot(n-1) = n^2+2n-3$ messages are exchanged, so the traffic increases with the number of parties squared. Selecting a lower threshold for the secret reconstruction $\frac{n}{2} \leq k < n$ lowers the total messages by $\Delta_{\text{messages}} = n^2 - n(k-1)$.

For a secure channel this protocol is information-theoretically secure: independent from computation power an adversary with $m_{\text{leaked}} < k$ shares will gain no information regarding the inputs.

2.1.3 Secure Comparison Protocol

The secure comparison protocol compares the secret inputs and provides the minimum or maximum in a set without revealing the inputs or the parties holding the minimum or the maximum.

The protocol is based on the privacy preserving protocol for maximum computation as described in Hasan et al. (2013). The general idea is to use bit-decomposition and utilize the secure addition protocol bit-wise. In iterations the secure-sum for the bits $(0 \vee 1)$ of the secrets multiplied with a random value are computed, starting from the most significant bit (MSB), limited by a predefined upper bound, to the least significant bit (LSB). The announced sum gives each party the information that at least one party has this bit set, if the sum is unequal zero. If a party has this bit not set itself it has

a lower value and commits only zeros in the following iterations. Storing the result of each iteration, the parties can reconstruct the maximum. For finding the minimum the protocol from Hasan et al. (2013) needs an extension as described in 2.1.3.2: inputs are negated (using the binary operation NOT), making the minimum in the set the largest value. Afterwards the maximum is determined as described above. Finally the found maximum is negated again to reconstruct the minimum in the set.

2.1.3.1 Example Computation

Public information: $n = 3$, $p = 67$, $\mathbb{Z}_p = \{1, ..., p-1\}$, $k = 3$, $s_i < b = 64$ (upper bound for secret value range)

Secrets: $s_1 = 13$, $s_2 = 27$, $s_3 = 17$

Target computation: $\min(s_i) = 13$, $\max(s_i) = 27$

Since $64_{10} = 1000000_2$ is defined as upper bound for the secret values the MSB is the sixth bit (second column in table 2.1).

Table 2.1: Binary representation of secrets s_i

Decimal $s_{i,10}$	Binary $s_{i,2}$					
13	0	0	1	1	0	1
27	0	1	1	0	1	1
17	0	1	0	0	0	1

Each party multiplies each bit with a random within $\mathbb{Z}_l$:

Table 2.2: Randomized binary representation of secrets

Decimal $s_{i,10}$	Binary $s_{i,2}$						Randomized					
13	0	0	1	1	0	1	0	0	45	61	0	57
27	0	1	1	0	1	1	0	12	31	0	5	15
17	0	1	0	0	0	1	0	24	0	0	0	9

There are six bits, therefore six rounds of secure addition ($\sum_{secure}$) are computed:

$$1^{st} \text{ round:} \quad \sum_{secure} = 0 \quad \Rightarrow \quad 6^{th} \text{ bit of the maximum is } \quad 0$$

$$2^{nd} \text{ round:} \quad \sum_{secure} = 36 > 0 \quad \Rightarrow \quad 5^{th} \text{ bit of the maximum is } \quad 1$$

Party p_1 disqualifies itself as the maximum (see table 2.3)

$$3^{rd} \text{ round:} \quad \sum_{secure} = 31 > 0 \quad \Rightarrow \quad 4^{th} \text{ bit of the maximum is } \quad 1$$

Party p_3 disqualifies itself as the maximum (see table 2.4)

$$4^{th} \text{ round:} \quad \sum_{secure} = 0 \quad \Rightarrow \quad 3^{rd} \text{ bit of the maximum is } \quad 0$$

$$5^{th} \text{ round:} \quad \sum_{secure} = 5 > 0 \quad \Rightarrow \quad 2^{nd} \text{ bit of the maximum is } \quad 1$$

$$6^{th} \text{ round:} \quad \sum_{secure} = 15 > 0 \quad \Rightarrow \quad 1^{st} \text{ bit of the maximum is } \quad 1$$

Table 2.3: 2^{nd} round

Decimal $s_{i,10}$	Randomized					
13	0	$\underline{0}$	~~45~~ ^{0}	~~61~~ ^{0}	0	~~57~~ ^{0}
27	0	12	31	0	5	15
17	0	24	0	0	0	9

Table 2.4: 3^{rd} round

Decimal $s_{i,10}$	Randomized					
13	0	0	0	0	0	0
27	0	12	31	0	5	15
17	0	24	$\underline{0}$	0	0	~~9~~ ^{0}

In total, each party has the bits $0|1|1|0|1|1$ stored and can reconstruct the correct maximum $\max(s_i) = 27$.

2.1.3.2 Protocol Extension for Minimum Determination

Using the negation of the binary representation, the order of the corresponding values in decimal numeral system is inverted (compare table 2.5). The computation is then the same as for the maximum search. The reconstructed maximum is finally negated to result in $\min(s_i)$.

In the second round booth P_2 and P_3 disqualify themselves as maximum. After six rounds each party holds: $1|1|0|0|1|0$ as the maximum. Negated this gives the minimum as $0|0|1|1|0|1_2 = 13_{10}$

Table 2.5: Negation of binary representation for minimum determination

Decimal $s_{i,10}$	Binary $s_{i,2}$						Negated $\bar{s}_{i,2}$					
13	0	0	1	1	0	1	1	1	0	0	1	0
27	0	1	1	0	1	1	1	0	0	1	0	0
17	0	1	0	0	0	1	1	0	1	1	1	0

2.1.3.3 Protocol Description

Assumptions:

- number of parties $n > 2$
- secure communication channel
- no malicious adversaries
- upper bound of sum $s \leq b$ can be estimated, so a prime $p > b$ can be chosen

The secure comparison protocol, as used in this thesis, consists of the phases for secure addition within iterations for the bitwise length of a predefined upper bound for the inputs:

1. The coordinator announces the number of parties for the computation and the indexation of each party.
2. For minimum-search: each party negates the secret input.
3. For each bit in the secret input starting from MSB to LSB each party runs through iterations:
   (a) If input is flagged as lower than maximum, then use $s_j = 0$ as the input. Otherwise multiply actual bit b with a random value R: $s_j = b \cdot R$.
   (b) Each party j sends shares $s_{j,i}$ of the input s_j to the other parties.
   (c) Each party i computes the sum over the received shares $s_{j,i}$.
   (d) Each party sends the computed sum to the coordinator.
   (e) The coordinator reconstructs the sum over the inputs using Lagrange-interpolation.
   (f) The coordinator broadcasts the reconstructed sum.

(g) Each party stores if the sum for the bit was equal 0 (set bit 0) or unequal 0 (set bit 1).

(h) Each party compares if bit from the computed sum is greater than own bit. If so input is flagged as lower than maximum.

4. For minimum-search: each party negates the stored sum-result.

Note: the assumption $n > 2$ for the secure addition and secure comparison protocols is not strict enough, if sum, min and max are computed for the same parties, since for $n = 3$ the secret between minimum and maximum can be restored (for a honest majority the mapping of values to parties is still secure though).

2.1.4 Existing Frameworks

In this section a short overview over existing SMPC solutions is given. While SMPC is an intensely researched field, practical work is less common.

The following solutions were considered

- MpcLib (see Zamani (2016))
- SEPIA (see Burkhart et al. (2012))
- SPDZ (see Keller et al. (2016))
- Sharemind (see sharemind.cyber.ee (2011))
- Enigma (see Zyskind, Nathan, and Pentland (2016))

Some key-features of the solutions are illustrated in figure 2.2. All projects emerged from university research. With the exception of Sharemind and Enigma, the frameworks seem to target primarily other researchers, reflecting in the lack of documentation and thereby reduced usability. The open-source library MpcLib is C# based, SPDZ uses C++ and Python and SEPIA is a Java library. Sharemind and Enigma are also booth based on university research (Enigma at MIT and Sharemind at University of Tartu) but evolved into market-ready business solutions. While Sharemind uses dedicated application-server, Enigma uses a distributed system of nodes based on Blockchain technology for SMPC,

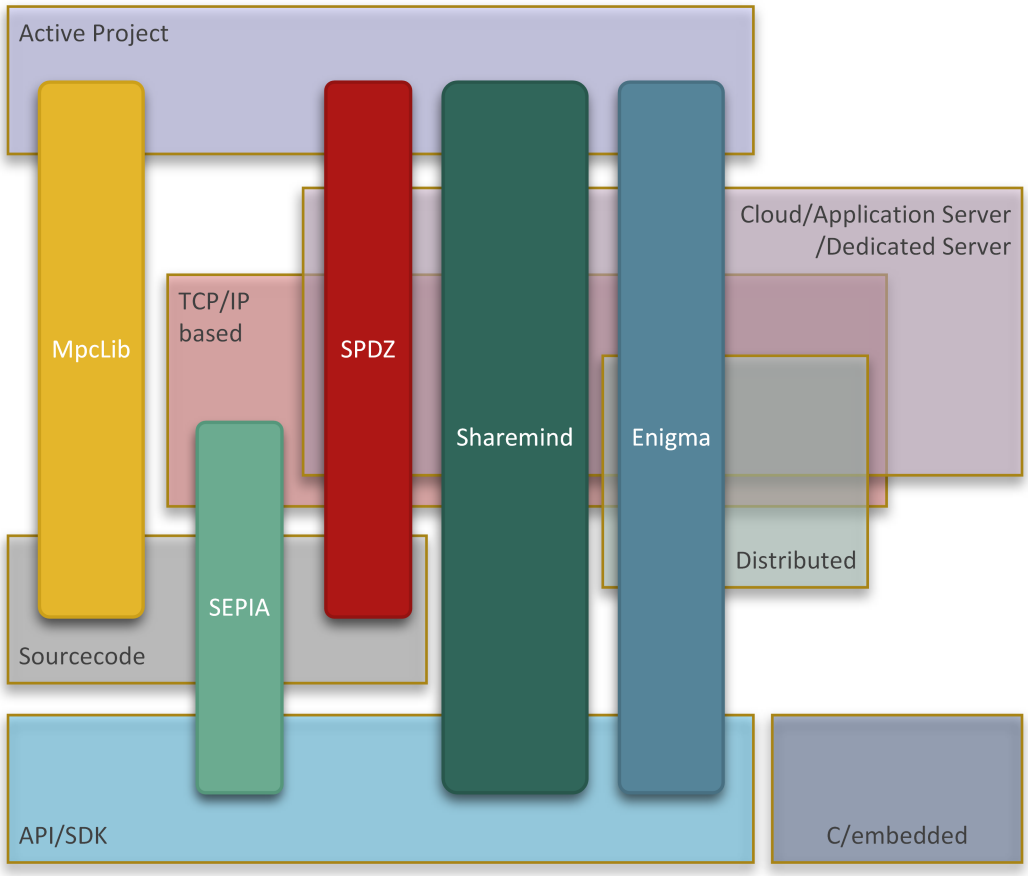


Figure 2.2: Existing SMPC software grouped by properties

booth with a focus on scalable secure data analysis. All solution are based on the Internet protocol suite and require at least locally run server or Internet access.

While all frameworks exceed the requirements regarding the SMPC functionality, they don't provide a solution for local ad-hoc networks without permanently available servers. Also the support for low-level devices is either undocumented or not given through programming language dependencies. The development of a framework with a focus on cross-platform usage, usability for developers without cryptographic research background and applicability for local ad-hoc networks for the described gamification use-cases is therefor justified.

2.2 Mobile Ad Hoc Networks

The framework developed as part of this thesis focuses on providing SMPC for MANETs or MANET-like networks. In this section the network topologies related to MANETs are briefly described (see 2.2.1) and the implementability based on current technology

standards are examined (see 2.2.2).

2.2.1 Network Topologies

Dorri, Kamel, and Kheirkhah (2015) describe a MANET as an "infrastructure-independent network with wireless mobile nodes" (Dorri, Kamel, and Kheirkhah 2015, p. 15). MANETs are similar to mesh networks, but the distinctive feature is the nodes' spatial degree of freedom. In comparison to a star network, there is no central switch dedicated to routing messages. Instead each node provides message passing abilities and acts as a multi-hop relay. The advantage of MANETs is the open network boundary: nodes can freely join and leaving nodes do not affect the functionality of the MANET. The key-features are:

- continuously self-configuring
- self-forming
- self-healing
- infrastructure-less
- peer-to-peer
- mobility of nodes (main difference to mesh network)

The message passing in a MANET can either be done by routing or flooding. Since the nodes can move freely, the neighbors will change often, so maintaining routing tables is expensive. The passing of messages without the availability of authentication protocols like HTTP over TLS (HTTPS) makes the communication also vulnerable against man-in-the-middle attacks. Of course flooding means broadcasting and is not cheap either in regard to message quantity and network load.

The mentioned key-features of MANETs make it a good network choice for a gamification setting based on mobile devices (smartphones, wearables, etc.), because it promises unobtrusive usage for participants without administrative maintenance effort. In the next section the availability and the implementability for Android devices is discussed, because of Androids dominant position as the globally leading smartphone operating system (OS) with a market share of above 80% (see Forni and Meulen (2016)).

2.2.2 Implementability on Android Devices

MANETs are especially of interest for military applications and disaster management but they are also gaining research focus for civil usage for example in context of IoT devices. Demonstrations of the implementability can be found for example in Open Gardens MeshKit software development kit (SDK) (Opengarden.com 2016a), which offers MANET abilities for Android and iOS devices and thereby forming a SPAN. MeshKit is also the foundation for Open Gardens FireChat (Opengarden.com 2016b), which is for example known in context of pro-democracy demonstrations. Since Android does not provide an application programming interface (API) for MANET functionality on Android devices (API 24 at the time of writing) and the MeshKit SDK is not open source and only available through Open Gardens partner program, a simplified (but extendable) implementation of MANET-like behavior is developed in the application layer (compare 3.2.1). Both for Wi-Fi and Bluetooth based connections, there can be limitations in regard to maximum concurrent connections. Vendor specific restrictions (hardware, driver) are hard to compensate reactive at runtime, so this issue has to be addressed proactive in 3.3 Architecture.

2.2.2.1 Bluetooth Based MANET

Usually Bluetooth connections with smartphones require pairing and user actions. This is not a useful process flow to build a MANET-like network, since nodes cannot simply join. Using the Bluetooth protocol radio frequency communication (RFCOMM) an insecure connection can be established, without the need for pairing and user interaction. Andersson et al. (2016) describe RFCOMM as the emulation of serial ports over Logical Link Control and Adaptation Protocol (L2CAP), supporting the emulation of multiple ports between two devices and ports between multiple devices (device dependent). Since multiple simultaneous connection have to share the available bandwidth per node, it takes $\frac{n}{2}$ times longer to share the same amount of data when using only one-to-one connections sequentially. For the targeted number of computation partners in this thesis, this is a tolerable overhead and practical system parameters will be evaluated in 5 and 42. The Bluetooth Special Interest Group has announced mesh networking protocols for upcoming specifications (Hegendorf 2016). This is very promising in regard of system provided MANET features, though it will take time (from experience with Bluetooth LE likely

years) until enough devices are equipped with compliant Bluetooth modules.

2.2.2.2 Wi-Fi Based MANET

Situations in which we can use Wi-Fi (or Global System for Mobile Communications (GSM)) usually provide Internet access, so Wi-Fi is not the primary target technology for this thesis. Generally, the callback-based architecture of the developed framework (compare 3.3 Architecture) enables the usage of different wireless technologies though. Even the interconnection of MANET-like networks is conceivable (as demonstrated with MeshKit), but it complicates the forming of the computation group (compare 3.2.1), because different optional channels between nodes have to be evaluated. With Android 4.0 (API level 14) the Wi-Fi Peer-to-Peer framework was introduced, which complies with the Wi-Fi Alliance's Wi-Fi Direct certificate program. Wi-Fi Direct states that one-to-one or group (many-to-one) connections are possible. One device acts as a group owner (soft access point), so it forms a star topology. To imitate a SPAN with Wi-Fi Direct multi-group communication has to be provided. In Funai, Tapparello, and Heinzelman (2016) limitations of Android in regard of multi-group networking as well as solutions are discussed. Other solutions (compare Thomas (2014)) include usage of custom kernels on rooted smartphones. Even though demonstrations on selected devices have shown the feasibility, such system modifications neglect the target group and the intentions of this framework.

Chapter 3

Design

Based on the findings in chapter 2 Background and extended with use-cases the requirements for the framework are specified in 3.1 Requirements. In 3.2 Decentralized, Distributed Computing specific requirements in context of complex processes are substantiated with algorithms for decentralized, distributed computing. Finally, a draft design is presented in 3.3 Architecture.

3.1 Requirements

In general this thesis follows the FURPS+ system for requirements as described by Eeles (2005): requirements are categorized into functional and non-functional requirements:

Functionality	functional requirements
Usability	non-functional requirements
Reliability	
Performance	
Supportability	

The functional and non-functional requirements are specified in 3.1.1 and 3.1.2.

3.1.1 Functional Requirements

Functional requirements define the functions the framework has to offer to meet the acceptance criteria. Based on chapter 2 Background we can divide the requirements into

two main fields: features regarding the accurate computation of the SMPC protocols and functions required to compensate the lack of a MANET API and technical limitations. Figure 3.1 presents the general functionality a party - respectively a node - expects from the system: especially the need for a secure channel and the limitation to run the SMPC only with nearby computation partners is caused by the missing multi-hop capabilities.

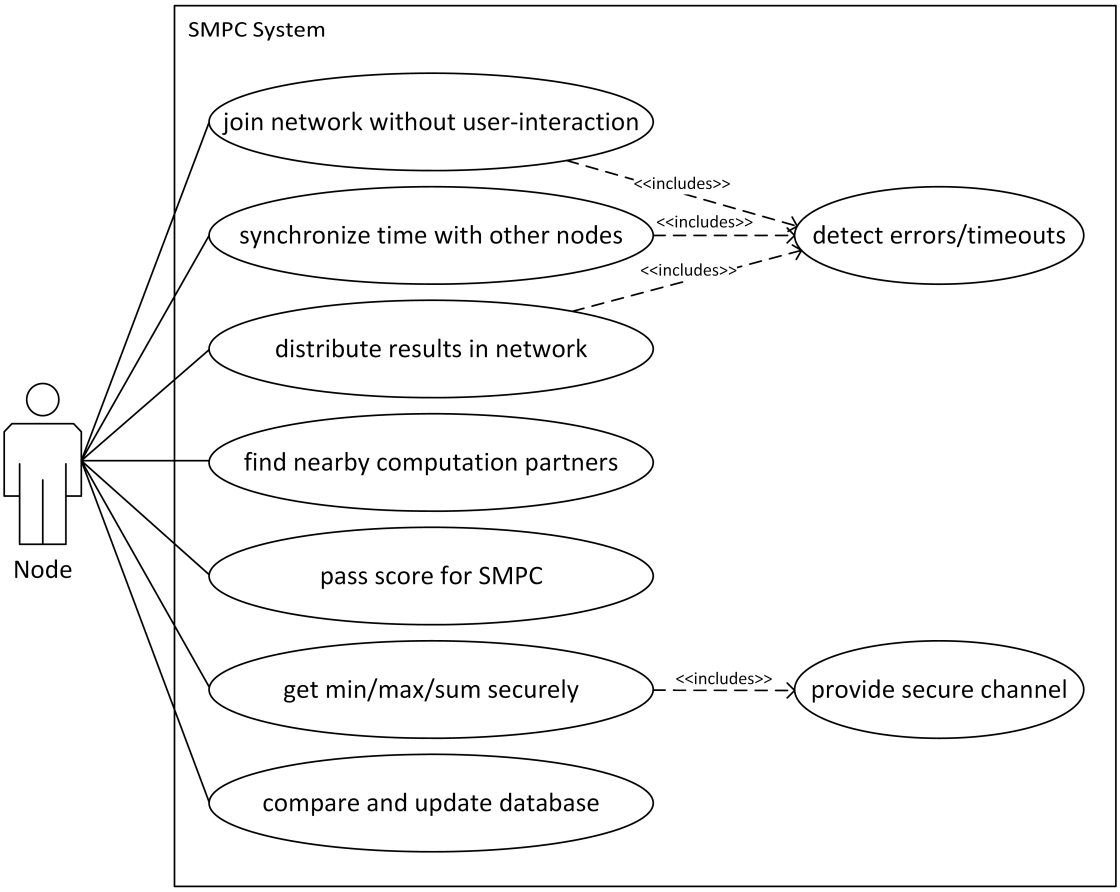


Figure 3.1: UML use-case diagram for the general functional requirements of a node

Since most functions (like the time synchronization and the multi-party computation) require the interaction between nodes, these processes need to be coordinated. In a distributed system there is no central authority, so a node has to become the temporal leader or coordinator for the duration of a process. In figure 3.2 the processes requiring coordination are described as use-cases from the view of a temporal coordinator.

Based on the use-cases functional requirements can easily be identified and specified. In table 3.1 the functional requirements are stated as user-stories, alongside assumptions and targeted tests.

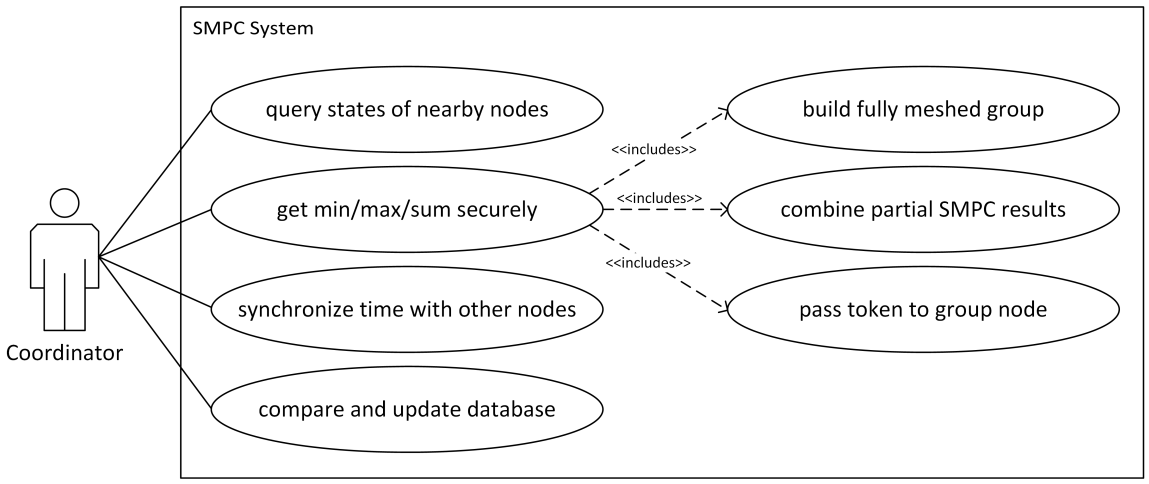


Figure 3.2: UML use-case diagram for the functional requirements for the coordinator

Table 3.1: Functional requirements

Name	**FR01 Pairing-less Connection**
Requirement	As a node I want to join the system without having to pair with other devices so that the system remains unobtrusive.
Assumptions	Device has Bluetooth capabilities with RFCOMM protocol.
Name	**FR02 Heartbeat**
Requirement	As a node I need to inform my coordinator if my computation is running longer than expected so that the system does not assume that the process has failed. As a coordinator I need to inform all group nodes if a computation is running longer than expected.
Assumptions	Hosting system provides system time.
Name	**FR03 Non-termination Detection**
Requirement	As a node I must be able to detect a communication problem so that I can reset my status.
Assumptions	Hosting system provides system time.
Name	**FR04 Coordinator Election**
Requirement	As a node I want to become coordinator for nearby nodes so that communication can be organized.
Name	**FR05 Token-Passing**
Requirement	As coordinator I want to be able to assign a group-member to coordinate a subprocess so that direct communication between group-members can be established.
Name	**FR06 Secure Multi-Party Computation Module**
Requirement	As a coordinator I want to form a group of fully meshed nodes and coordinate the execution of the secure addition and secure comparison protocols using a secure communication channel.
Assumptions	Group size > 2. All group-members are time-synchronized and have a score within the same time-frame limits.
Testability	Unit tests to proof correctness of implementation. Performance-tests with different number of computation partners and validation of result.
Name	**FR07 Clock Synchronization**
Requirement	As coordinator I want to synchronize the clocks of nearby nodes so that computation results are not biased because of different time settings.
Testability	Unit tests to proof correctness of implementation.
Name	**FR08 Database Synchronization**
Requirement	As coordinator I want to compare my database status with nearby nodes and exchange missing entries without having to compare all entries.
Assumptions	Participating nodes are idle and not waiting for a computation.

3.1.2 Non-Functional Requirements

Non-functional requirements describe quality attributes the system has to comply to. Two use-cases from a developer view are illustrated in figure 3.3.

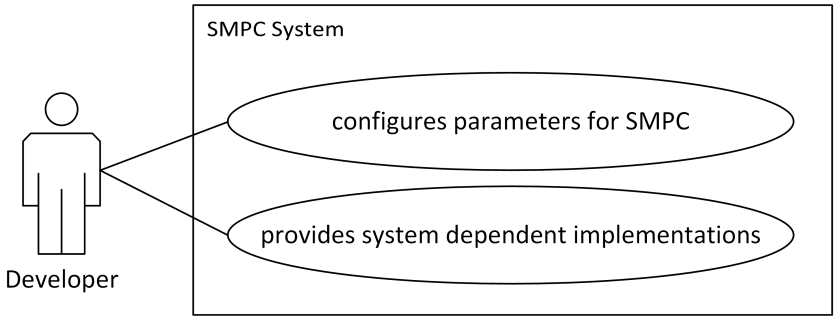


Figure 3.3: UML use case diagram for developer

Based on the use-cases for developers and general demands regarding the maintainability, expandability and performance to make the framework applicable for real-life settings, non-functional requirements can be specified as listed in table 3.2.

Table 3.2: Non-functional requirements

Name	**NFR01 Usability**
Requirement	The framework shall be configurable, so that a developer using the framework can configure the settings for the SMPC.
Name	**NFR02 Maintainability**
Requirement	The framework shall be maintainable, so that the code and documentation make it clear for a developer what callbacks have to be implemented and how the framework can be used in an Android device.
Name	**NFR03 Performance**
Requirement	The framework shall be secure while providing enough performance, that computations can properly terminate for nodes that move at walking speed ($\approx 1\frac{m}{s}$).
Name	**NFR04 Expandability**
Requirement	The frameworks coupling with the wireless technology shall be loosely, so that the system can be extended without having to touch core functionalities regarding the SMPC.

3.2 Decentralized, Distributed Computing

While the protocols for secure addition and secure comparison and thereby the requirement FR06 Secure Multi-Party Computation Module are already well-defined (compare 2.1.1, 2.1.2 and 2.1.3), other functional requirements need further methodical substantiation. FR04 Coordinator Election and FR05 Token-Passing are addressed in 3.2.1, FR02 Heartbeat and FR03 Non-termination Detection are discussed in 3.2.3, an algorithm for FR07 Clock Synchronization is provided in 3.2.2 and finally FR08 Database Synchronization is covered in 3.2.4.

3.2.1 Coordinator Election and Coordinator Role

As discussed in 2.2.2 Implementability on Android Devices fully featured MANETs are currently not provided and mapping it completely in the application layer is beyond the scope of this thesis. Overcoming the technical limitations, the system can be build with sequential communications instead of parallel. As stated in 2.2.1 Network Topologies communication in context of SMPCs is only done in a fully meshed subgroup of the network, which also simplifies the coordinator election.
A node will try to become the coordinator, when

1. it enters the network after longer disconnection: event driven.
2. a new personal score is ready for SMPC: event driven.
3. all SMPC computations for a score are done: event driven.
4. an event driven attempt failed and a certain amount of time passed: timer based.

Extending requirement FR04 Coordinator Election and to avoid situations of competing nodes trying to become coordinator and thereby booth repeatedly failing because neither can acquire enough computation partners, the timer based approach is supported by the exponential backoff algorithm. Ganga et al. (2010, p.67) describe the exponential backoff algorithm for collision detection and retransmission: if a coordinator appointment failed (equivalent to collision detection in original description) a factor for the waiting time till the next attempt is selected uniformly random from an increasing range, reducing the probability for competing coordinator candidates. The process is outlined in form of an UML activity diagram in figure 3.4.

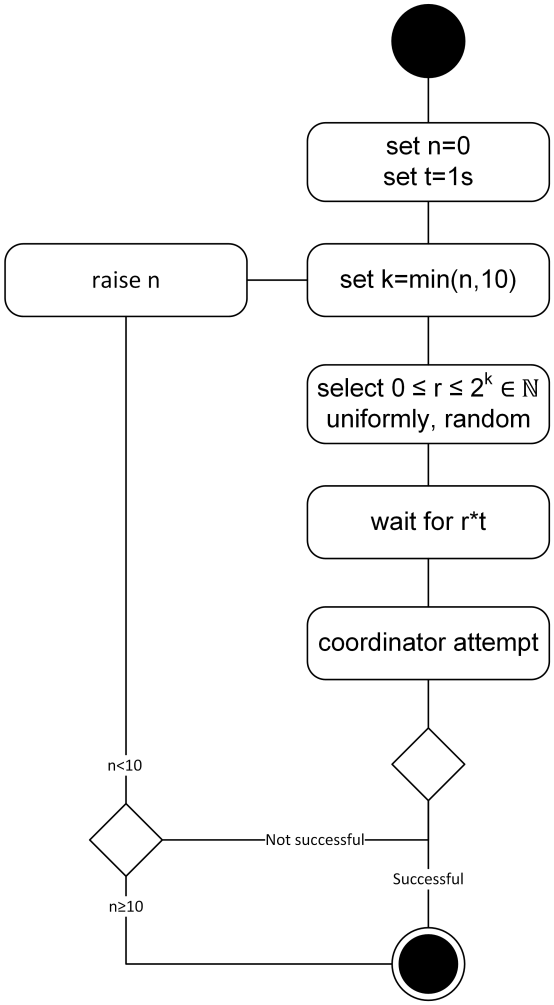
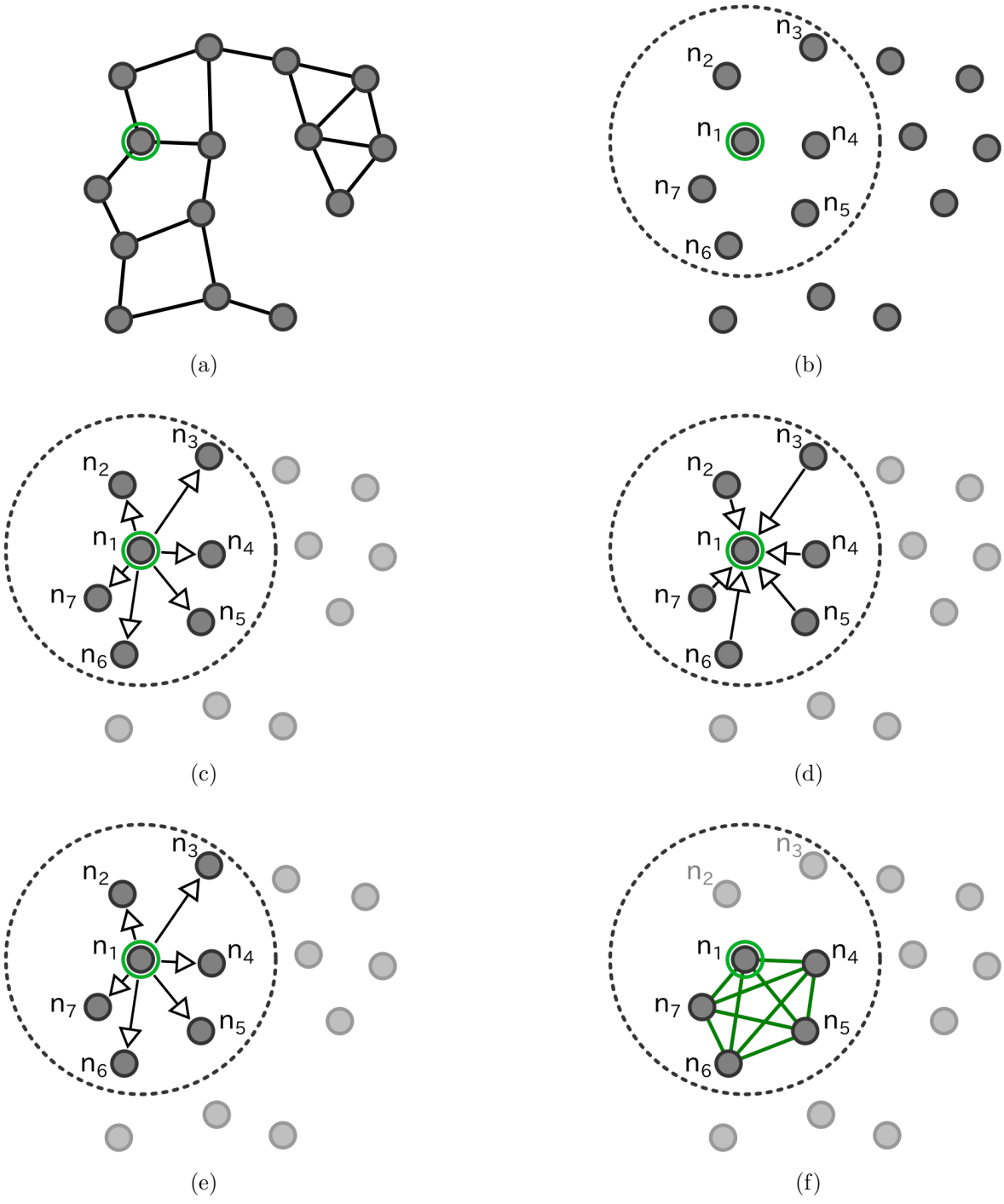


Figure 3.4: UML activity diagram for exponential backoff algorithm

In regard to the execution of the SMPC protocols in FR06 Secure Multi-Party Computation Module, the coordinator has to find a computation group. In a mesh network with routing and point-to-point encryption as displayed in figure 3.6a, the green marked coordinator can simply broadcast a computation request and responding nodes form the computation group. Caused by the technical limitations (see 2.2.2), the coordinator has to find a fully meshed group within its reach: this guarantees that each node can directly communicate with all computation partners and messages required for securing the channel are not passed through other nodes. First the coordinator n_1 discovers nearby nodes (see figure 3.6b). Then a list of these devices (identified by media access control (MAC) address) is sent to every neighboring node (see n_2 to n_7 in figure 3.6c). Each node responds with the intersection of the received device list with the own list of discovered devices (see figure 3.6d). To reduce the payload of the responses, they only contain a list of booleans, indication if the device with the same index in the received device list is seen by the node. The coordinator then computes the maximum group of fully meshed nodes and sends computation partners an associative array assigning new 8-bit ids (see figure 3.6e), which reduce the payload in following steps. Nearby nodes, that are not part of the computation group, receive an indicator to abort the computation. Each node in the computation group has a list of the group and the assigned ids and can exchange public keys with group members, forming a fully meshed, end-to-end encrypted group (see figure 3.6f).

Since parallel message exchange for the computation group cannot be guaranteed (see 2.2.2), the coordinator controls sequential message exchanges with token passing in accordance with FR05 Token-Passing. For example when n nodes want to exchange n secrets divided into n shares each, the coordinator first requests successively the shares for himself $(s_i, 1)$ from the other $n-1$ nodes, while transmitting his own shares (s_1, j) with the request. Then the communication token gets passed to the next node, which in turn requests the shares for himself from the other $n-2$ nodes while transmitting his own shares and so on. An exemplary share-exchange for $n = 3$ with token-passing is illustrated in figure 3.6.

The combination of processes with the same communication partners, single-digit bytes of payloads and short process termination is a good option to reduces the total message-occurrence in the network: for example when a coordinator requests the states



(a) (b) (c) (d) (e) (f)

Figure 3.5: Formation of fully meshed computation group

of nearby nodes, it can be combined with the clock synchronization.

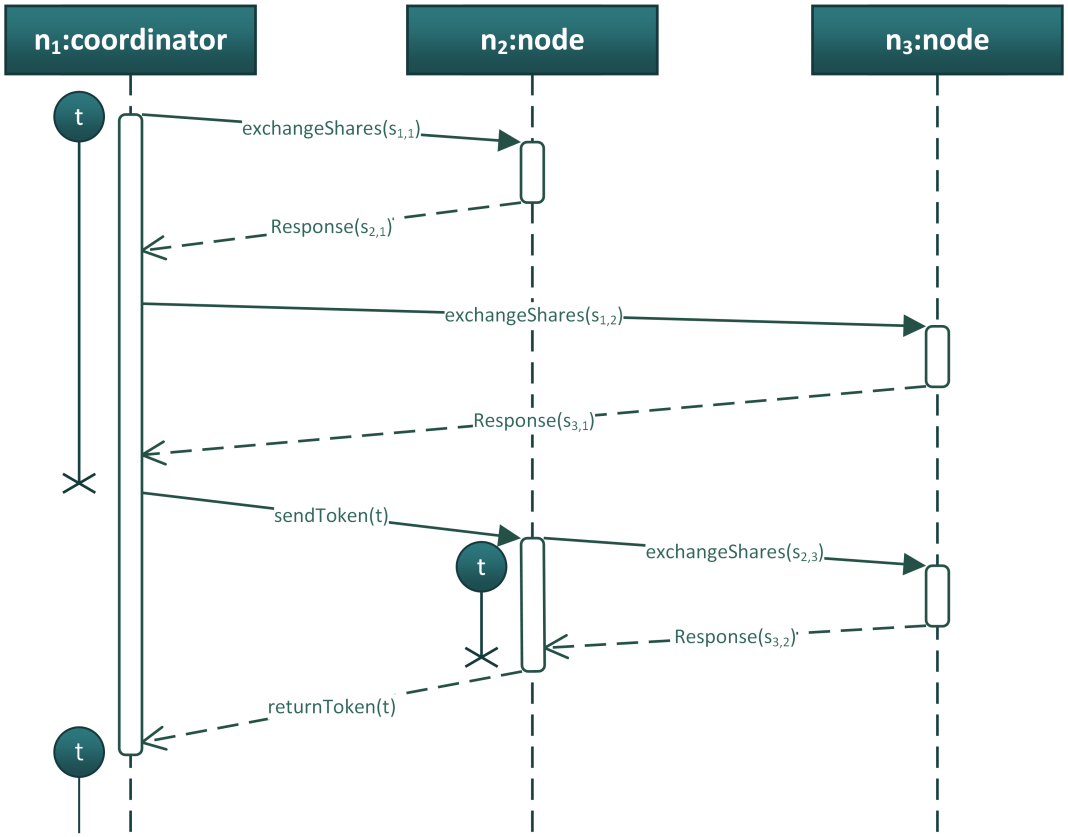


Figure 3.6: UML sequence diagram for passing of communication token t

3.2.2 Clock Synchronization

For statistical data in a gamification system, the sequence of events in infinitesimal time units is not as important as comparing the data for the same durations in Coordinated Universal Time (UTC), so a synchronization of physical clocks is needed as requested in FR07 Clock Synchronization. In this thesis the well known Berkeley-algorithm for internal clock synchronization in distributed systems is used as described in Ghosh (2015).
The coordinator

1. requests the current time values t_i from participating nearby nodes i.
2. computes the average of these values $t_{average}$.
3. reports back the adjustments $\Delta_i = t_{average} - t_i$

Since the communication between the coordinator and a node takes time, the received response is already outdated. This is compensated by observing the Round Trip Time (RTT) and using half of the duration as a correction value (compare 3.1). The RTT is herein the timespan between sending a request to a node and receiving its response (see

figure 3.7).

$$t_i' = t_i + \underbrace{\frac{RTT}{2}}_{\text{correction value}} = t_i + \frac{t_e - t_s}{2} \tag{3.1}$$

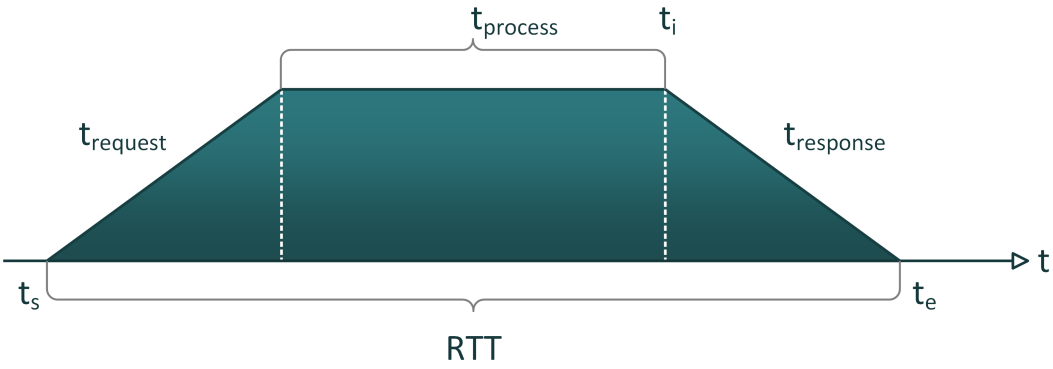


Figure 3.7: Round Trip Time

By sending the adjustments Δ_i instead of the adjusted time, the receiving nodes do not need to compensate the received value with the RTT. Figure 3.8 depicts the computation of the adjustments using Berkeley with RTT correction for three nodes.

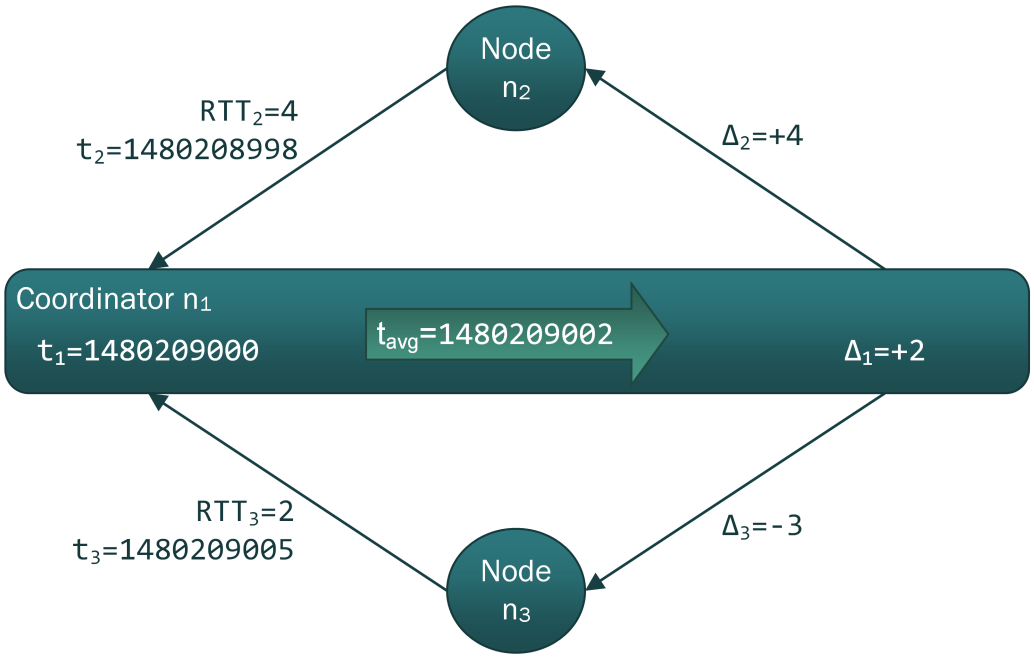


Figure 3.8: Example computation of adjustments with Berkeley

For further improvement of the accuracy the processing duration between receiving a request and sending the response $t_{process}$ can be measured and send to the coordinator.

In this thesis the simple approximation for $t_{response}$ is used, since the additional payload extends the transmission duration. The RTT has to be below an upper bound though, otherwise there is to much uncertainty regarding the influence of $t_{request}$, $t_{process}$ and $t_{response}$. Also bounds for the deviation of the time can be defined to reduce the influence of outliers.

The framework does not change the actual clock setting on the hosting system, but stores the computed time difference Δ_t and applies the value to all time-related actions. To make sure that a node is time-synchronized before scores and computations are acquired, it is reasonable to trigger a synchronization when the node joins the network.

3.2.3 Non-termination Detection

Especially since the coordinator gives temporarily away the message token and goes into a waiting state, there has to be a protocol to detect non-termination for processes. Meeting FR03 Non-termination Detection each request to another node and each local computation initializes the start of timers. The local timer triggers the transmission of a heartbeat message (compare FR02 Heartbeat) to the coordinator, signaling that the process is still intact, but not yet finished. If the coordinator receives a heartbeat message, it informs the other nodes in the computation group (causing them to reset their local timers), and resets its local timeout-timer. If the coordinator reaches a limit for the timer without receiving a heartbeat message, non-termination is assumed and all group members are informed, that the computation failed. The heartbeat protocol for the coordinator waiting for response is outlined in figure 3.10a, while the protocol for a node in possession of the message token is displayed in figure 3.10b.

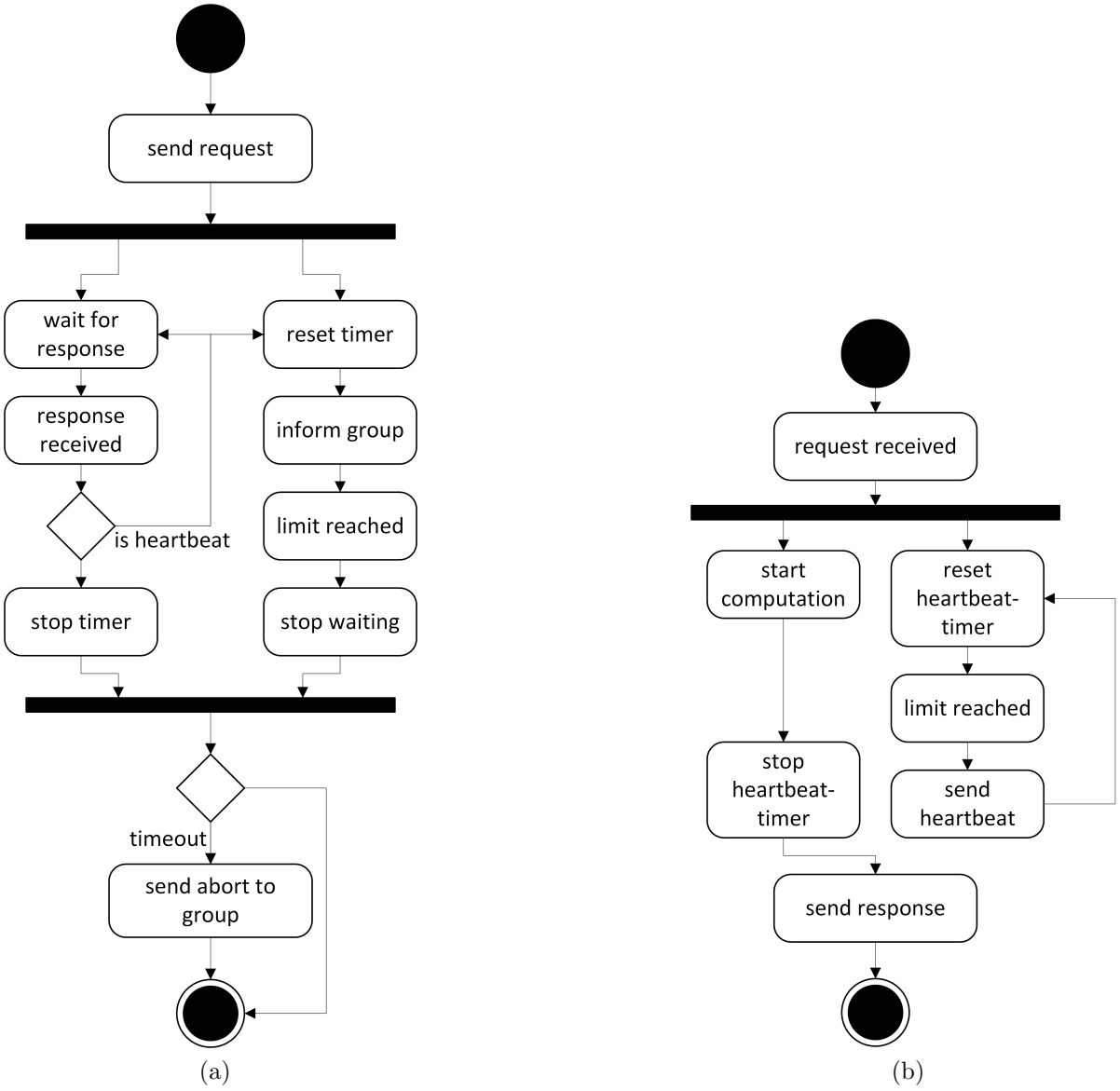


Figure 3.9: Avoidance of false non-termination detection through heartbeat messages

3.2.4 Distributed Databases

A distributed system without central servers, that guarantee availability throughout the network, has to provide a distributed database model. This means, that nodes need to compare their database states with each other and synchronize differences. Since the nodes can enter and leave the network freely, preservation of the data in the system as well as consistency has to be considered.

The framework deals only with entry-sets of the database and lets the hosting system handle the actual storage. Since each node hosts its own database, transactions for concurrent access is not an issue.

An entry consists of:

- Hash over the entry
- Unix timestamp
- size of computation group
- indicator for min, max or sum
- value

The combination of hash and Unix timestamp generates a key for the entry that is most likely collision free. The size of the computation group is needed to compute the arithmetic average from multiple entry-sum-values in a specified time-window:

$$\left.\begin{array}{l} \underbrace{s_1, s_2, s_3, s_4, s_5}_{n_1=5} \\ \underbrace{s_6, s_7, s_8}_{n_2=3} \end{array}\right\} \begin{array}{l} v_1 = \sum_i s_i \\ v_2 = \sum_i s_i \end{array} \Bigg\} \overline{v}_i = \frac{v_1 + v_2}{n_1 + n_2}$$

Since the framework offers three types of SMPCs the entry must reflect the source of the value. By comparison and updating, each node will have eventually all entries, so a distributed database has eventual consistency. To meet with the requirement FR08 Database Synchronization each node holds the sum of the entries' hashes within a specified time-window. This value is used to compare the database-states between nodes: if the values are equal, the databases are likely consistent (collisions are possible though but only for short durations until new SMPC results are generated or collision free nodes are encountered), otherwise entries are compared and exchanged. First the coordinator request the hash-sum. If they match an acknowledgment is send, otherwise up to n (predefined upper bound) hashes of the entries in anti-chronological order are send in an array to the node. The node response with an array of booleans, representing if the hashes are known. If the response-array contains zeros, then the unknown entries are transmitted. After an entry-exchange the hash-sums are compared, to determine if consistency is reached (coordinator request hash-sum if needed, compare figure 3.10). If the hash-sums do not match, the node sends up to n entry-hashes to the coordinator, skipping already evaluated entries. This is repeated until consistency is reached or a request times out and the process is aborted. Figure 3.10 displays the basic process for $n = 2$, with ASCII-values as mock-up hashes:

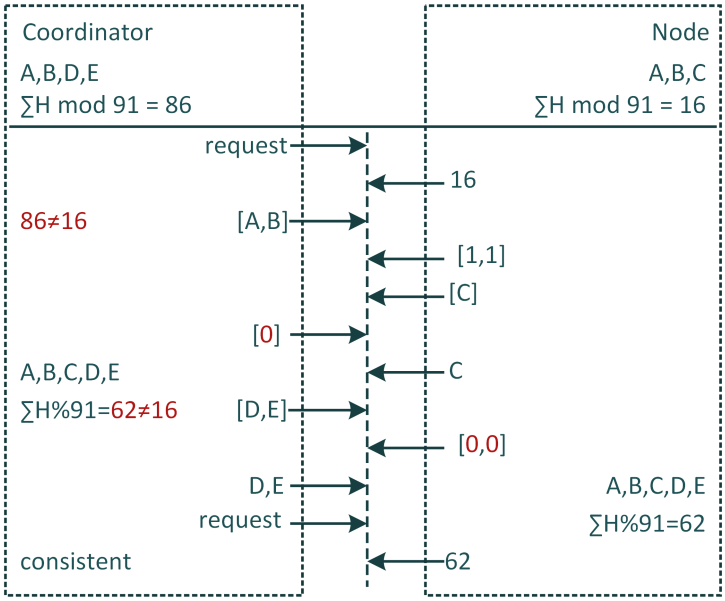


Figure 3.10: Database synchronization scheme

3.2.5 Securing the Communication Channel

As requested in requirement FR06 Secure Multi-Party Computation Module and noted in 2.1.2 the SMPC protocols need secure communication channels. Listen in on wireless communication means receiving the radio signals, so for common wireless technologies this is easily accomplished. Since the physical layer is more or less public, the communication needs encryption. For this framework two kinds of encryption are used: first asymmetric cryptography is used to exchange a session-key, which is then used to secure messages with symmetric encryption, as displayed in figure 3.11. This principle is well known from TLS encryption used in HTTPS. For the symmetric encryption the AES as described by Delfs and Knebl (2015, pp. 19-25) is used. For the asymmetric encryption the public-key cryptosystem RSA as described by Delfs and Knebl (2015, pp. 49-76) is used. AES encrypts and decrypts faster than RSA, because RSA requires long keys (2048 bit and longer recommended) for proper security. But AES needs sender and receiver to know the cipher, and the exchange of this key over an insecure channel only with AES is not possible.

The basis for the cryptosystem by RSA is the prime-factorization, which requires super-polynomial time. RSA is asymmetric, since there is one key for encryption and one key for decryption. In this setting, the public key is used for encryption, so only the receiver with the private key can decrypt the secret. For the key generation two large prime numbers are selected: $p, q, p \neq q \in \mathbb{P}$. The product $n = p \cdot q$ is computed.

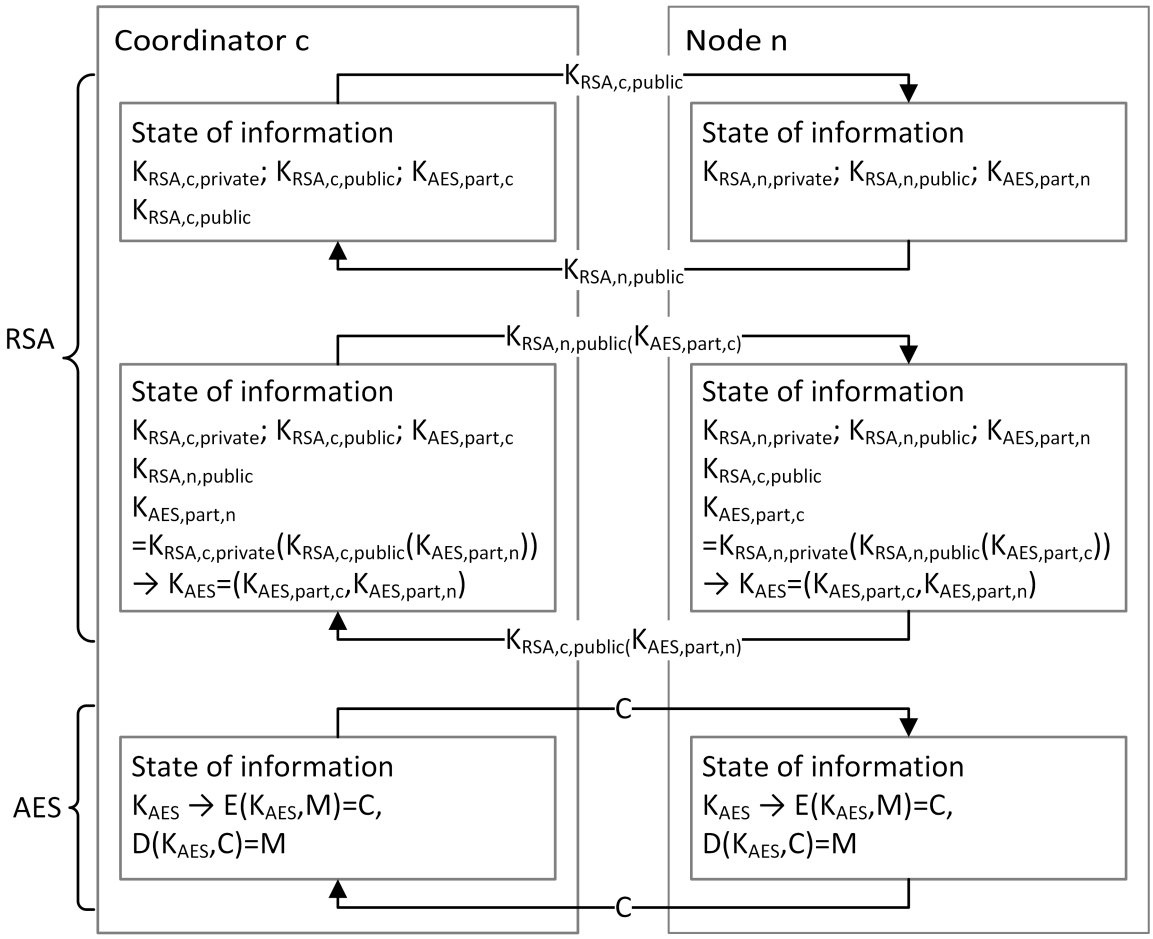


Figure 3.11: Securing communication with RSA and AES

Euler's Phi function $\phi(n) = (p-1)(q-1)$ is computed and a coprime integer e is selected $1 < e < \phi(n)$. A common value for e is 65537. n and e form the public key. The private key is formed from n and d, where d meets $e \cdot d \equiv 1 \mod \phi(n)$. For the symmetric encryption, booth partners use the same key. AES is an iterated block cipher with a block length of 128 bits and key length of 128, 192 or 256 bits. The iterations (called rounds) follow the Rijndael algorithm. A detailed description of the algorithm can be found in Delfs and Knebl (2015, pp. 20-25).

3.3 Architecture

Based on 3.2.1 it is sensible, that the central element in this framework is a node component. Figure 3.12 displays the UML component diagram for the framework design and illustrates the basic conjunctions between the components, as well as key-functionalities. The node component can also act as the coordinator and in either state communications and computations let it pass through different states of activity. This framework therefor uses the state pattern: the current state determines the behavior and abilities of the node. In regard to the hosting system the node component utilizes a API component, which uses callbacks to bind the communication layer and the system clock to the framework in accordance with NFR02 Maintainability. To handle the message encryption a cryptography module is needed, providing the functionality described in 3.2.5. As described in 3.2.3 a handler for timeout detection and heartbeat message triggering is provided. Parameters for communication, cryptography and SMPC need to be accessible in a central component to meet NFR01 Usability.

Since it is likely, that the technical limitations described in 2.2.2 will be overcome in future releases, the framework's core functionality is independent from the efforts to provide the self-forming network abilities (avoidance of code smell change preventer/shotgun surgery). So in case of full MANET implementation only the node component has to be adopted.

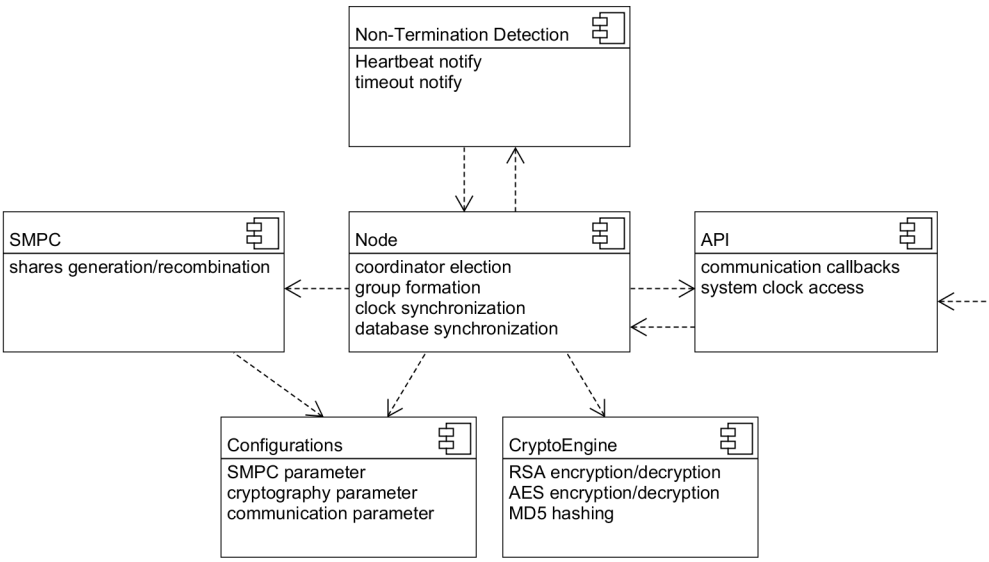


Figure 3.12: UML component diagram

Chapter 4

Implementation

While the algorithms and protocols described in chapters 2 and 3 are language-independent, it is sensible to use C for the development, since it is widely supported on most OS's including embedded OS's. In this chapter the development environment and tools are introduced in 4.1, followed by descriptions of the implemented modules in 4.2. Important structures like the messages passed between nodes or the database entries are also described in 4.2. For the encryption of the messages the open-source library WolfCrypt is embedded (see 4.2.3). In 4.3 Interfacing the Library the implementation tasks required for the usage of this library in a C project (in a Linux environment) as well as in an Android project are demonstrated.

4.1 Development Tools

4.2 Module Structure

The framework follows a modular code design: functions are separated into modules based on affiliation.

4.2.1 Node Module

4.2.1.1 Message Protocol

4.2.1.2 Database Entry

Hashing with wolfCrypt: Embedded Crypto Engine

4.2.2 Non-Termination Detection Module

4.2.3 Cryptography Module: WolfCrypt

RSA/AES with wolfCrypt: Embedded Crypto Engine wolfssl.com (2016)

4.2.4 API Module

4.2.5 Configuration Module

4.2.6 SMPC Module

4.3 Interfacing the Library

4.3.1 Configuration

4.3.2 Usage in C

4.3.3 Usage in Android

Chapter 5

Evaluation

5.1 Testing Tools

5.2 Examination of Computation Time Dependent on Computing Power

5.3 Examination of Computation Time Dependent on Number of Participants

Chapter 6

Discussion

Based on the gamification related target group, assumptions regarding the significance of dated data can be made: for example it is of interest to compare the general average for the previous year with the personal scores, maybe also on month or week level. In contrast a more fine-grained resolution of dated data will be rarely used while requiring extensive storage, compared to summarized data.

Chapter 7

Conclusion

References

Andersson, Christian et al. (2016). *RFCOMM WITH TS 07.10*. Bluetooth Special Interest Group. [online] Available at: URL: https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=263754 (visited on 11/25/2016).

Aumann, Yonatan and Yehuda Lindell (2007). "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries". In: *Theory of Cryptography: 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007. Proceedings*. Ed. by Salil P. Vadhan. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 137–156. ISBN: 978-3-540-70936-7. URL: http://dx.doi.org/10.1007/978-3-540-70936-7_8.

Burkhart, Martin et al. (2012). *SEPIA library*. [online] Available at: URL: http://www.sepia.ee.ethz.ch/index.html (visited on 12/08/2016).

Clifton, Chris et al. (2002). "Tools for Privacy Preserving Distributed Data Mining". In: *SIGKDD Explor. Newsl.* 4.2, pp. 28–34. ISSN: 1931-0145. URL: http://doi.acm.org/10.1145/772862.772867.

Cramer, Ronald, Ivan Bjerre Damgård, and Jesper Buus Nielsen (2015). *Secure Multiparty Computation and Secret Sharion*. Cambridge University Press.

Delfs, H. and H. Knebl (2015). *Introduction to Cryptography: Principles and Applications*. Information Security and Cryptography. Springer Berlin Heidelberg. ISBN: 9783662479742.

Dorri, Ali, Seyed Reza Kamel, and Esmaeil Kheirkhah (2015). "Security challenges in mobile ad hoc networks: A survey". In: *arXiv preprint arXiv:1503.03233*.

Eeles, Peter (2005). *Capturing Architectural Requirements*. IBM. [online] Available at: URL: http://www.ibm.com/developerworks/rational/library/4706.html#N10073(archived at: http://web.archive.org/web/20161129163620/http://www.ibm.com/developerworks/rational/library/4706.html) (visited on 11/28/2016).

Forni, Amy Ann and Rob van der Meulen (2016). *Gartner Says Five of Top 10 Worldwide Mobile Phone Vendors Increased Sales in Second Quarter of 2016*. Gartner, Inc. [online] Available at: URL: http://www.gartner.com/newsroom/id/3415117 (visited on 12/06/2016).

Funai, Colin, Cristiano Tapparello, and Wendi B. Heinzelman (2016). "Supporting Multi-hop Device-to-Device Networks Through WiFi Direct Multi-group Networking". In: *CoRR* abs/1601.00028. URL: http://arxiv.org/abs/1601.00028.

Ganga, Ilango S. et al., eds. (2010). *IEEE Standard for Information Technology — Telecommunications and Information Exchange Between Systems — Local and Metropolitan Area Networks-Specific Requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment 4: Media Access Control Parameters, Physical Layers and Management Parameters for 40 Gb/s and 100 Gb/s Operation*. New York, NY, USA: LAN/MAN Standards Committee. URL: http://standards.ieee.org/about/get/802/802.3.html.

Ghosh, Sukumar (2015). *Distributed Systems: An Algorithmic Approach, Second Edition*. Chapman & Hall/CRC Computer and Information Science Series. CRC Press. ISBN: 9781498760058.

Hasan, O. et al. (2013). "A Privacy Preserving Prediction-based Routing Protocol for Mobile Delay Tolerant Networks". In: *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 546–553.

Hegendorf, Steve (2016). *Get ready for Bluetooth mesh!* Bluetooth Special Interest Group. [online] Available at: URL: http://blog.bluetooth.com/__trashed/(archived at: http://web.archive.org/web/20161125191028/http://blog.bluetooth.com/__trashed/) (visited on 11/25/2016).

Keller, Marcel et al. (2016). *Bristol University — Department of Computer Science*. [online] Available at: URL: https://www.cs.bris.ac.uk/Research/CryptographySecurity/SPDZ/ (visited on 12/08/2016).

Opengarden.com (2016a). *Mesh networking made easy - Open Garden*. Open Garden. [online] Available at: URL: https://www.opengarden.com/meshkit.html(archived at: http://web.archive.org/web/20161126105839/https://www.opengarden.com/meshkit.html) (visited on 11/26/2016).

Opengarden.com (2016b). *Start Something - Open Garden*. Open Garden. [online] Available at: URL: https://www.opengarden.com/firechat.html(archived at: http://web.archive.org/web/20161126110144/https://www.opengarden.com/firechat.html) (visited on 11/24/2016).

Shamir, Adi (1979). "How to Share a Secret". In: *Communications of the ACM*.

sharemind.cyber.ee (2011). *Sharemind - analyze confidential data without compromising privacy*. Cybernetica AS. [online] Available at: URL: https://sharemind.cyber.ee/ (visited on 12/08/2016).

Sheikh, Rashid, Beerendra Kumar, and Durgesh Kumar Mishra (2009). "Privacy Preserving k Secure Sum Protocol". In: *CoRR* abs/0912.0956. URL: http://arxiv.org/abs/0912.0956.

Thomas, Josh (2014). *The SPAN Project*. [online] Available at: URL: https://github.com/ProjectSPAN (visited on 11/25/2016).

wolfssl.com (2016). *wolfSSL - Products — wolfCrypt Cryptography Engine*. wolfSSL Inc. [online] Available at: URL: https://www.wolfssl.com/wolfSSL/Products-wolfcrypt.html (visited on 11/28/2016).

Yao, Andrew C. (1982). "Protocols for Secure Computations". In: *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*. SFCS '82. Washington, DC, USA: IEEE Computer Society, pp. 160–164. URL: http://dx.doi.org/10.1109/SFCS.1982.88.

Zamani, Mahdi (2016). *GitHub - mahdiz/mpclib: MpcLib - A Multi-Party Computation Library*. [online] Available at: URL: https://github.com/mahdiz/mpclib (visited on 12/08/2016).

Zyskind, Guy, Oz Nathan, and Alex Pentland (2016). *Enigma*. [online] Available at: URL: http://www.enigma.co/ (visited on 12/08/2016).

To do...

- ☐ 1 (p. i): bad word high acceptable here?
- ☐ 2 (p. 1): 5-10%, including motivation, general audience
- ☐ 3 (p. 1): mention IoT problems (Distributed Denial of Service (DDoS) and botnets), to emphasis usefulness of connected but not online
- ☐ 4 (p. 3): 10-15%; thorough review of the state of the art; informed audience
- ☐ 5 (p. 3): general idea
- ☐ 6 (p. 5): describe number off messages, usage of threshold as trade-off between security and performance -¿ START
- ☐ 7 (p. 5): describe number off messages, usage of threshold as trade-off between security and performance -¿ END
- ☐ 8 (p. 7): Write about differential privacy?
- ☐ 9 (p. 10): secure addition with verification (Cramer, Damgård, and Nielsen 2015); number of messages
- ☐ 10 (p. 19): 15-20%; explains complete processing chain; explains what methods are used; for someone that wants to know what was done in detail
- ☐ 11 (p. 22): visualize finding fully-meshed nearby group in mesh network
- ☐ 12 (p. 24): Reduction of active connections; compare number of additional rounds needed; discuss timeouts

- ☐ 13 (p. 35): state machine; state pattern; client server architecture; UML state diagrams for 1. joining network (get time; set clock delta), 2. finding computation partners, 3. running computation, 4. compare database
- ☐ 14 (p. 37): 15-20%; details on the implementation; for someone who wants to continue the work
- ☐ 15 (p. 37): use doxygen for documentation
- ☐ 16 (p. 37): mention: development tools?
- ☐ 17 (p. 37): UML class diagram
- ☐ 18 (p. 37): modular programming: separating interface from implementation
- ☐ 19 (p. 37): describe components documentation style
- ☐ 20 (p. 37): describe the node-states: state machine in combination with state struct
- ☐ 21 (p. 37): describe the message protocol detailed/bit level
- ☐ 22 (p. 37): describe how the library encrypts the messages; flag for message to signal encryption (first Byte of payload 0/1)
- ☐ 23 (p. 37): describe the database entry struct
- ☐ 24 (p. 38): change label in component diagram
- ☐ 25 (p. 38): describe how external system can provide better seeds for the public key system
- ☐ 26 (p. 38): change label in component diagram

- ☐ 27 (p. 38): describe the module for creating shares; describe generation of communication partner matrix; describe secure addition module; describe secure maximum module; describe secure minimum module
- ☐ 28 (p. 38): external system: extend on RFCOMM; widespread
- ☐ 29 (p. 38): describe configuration.h: what can be configured, override of illegal configurations/sanity checks
- ☐ 30 (p. 38): describe usage of framework in a C environment with example for Linux with BlueZ stack-RFCOMM
- ☐ 31 (p. 38): describe library is used in raspberry and in xadow
- ☐ 32 (p. 38): describe how library is used with android Native Development Kit (NDK); describe Java wrapper
- ☐ 33 (p. 38): describe usage of framework in a Java environment with example for Android with Android Bluetooth API (Bluedroid stack)
- ☐ 34 (p. 38): https://developer.android.com/reference/java/security/SecureRandom
- ☐ 35 (p. 39): 5-15%; outcome; how was it tested; for supervisor
- ☐ 36 (p. 39): Unity (Unit test for C); JUnit; Android based multi-device tests
- ☐ 37 (p. 39): centralized client-server test application for android: trigger test runs, report results (measured execution time, correctness)
- ☐ 38 (p. 39): test on: xadow (IoT); RaspberryPi 3 (SBC); Android 4 (single core, low RAM), Android 5 (multi-core, 2 GB RAM)

- ☐ 39 (p. 39): test on pc: define CPU limited processes, use pipes as communication channel, run scalability tests
- ☐ 40 (p. 39): n devices, n shares (highest security)
- ☐ 41 (p. 39): n devices, k (¿n/2) shares (adjustable security)
- ☐ 42 (p. 40): 5-15%; outcome for a design-reader
- ☐ 43 (p. 40): combine requirements with evaluation
- ☐ 44 (p. 40): extend protocol: implement merging of results, to reduce probability of not finding computation partner; implement alternative protocols
- ☐ 45 (p. 40): implement optional verification for addition, if performance is good enough for real life application
- ☐ 46 (p. 40): computation group, implement fairness: after rejection as member of computation group either reduce backoff time or raise a priority indicator a do a weighted group
- ☐ 47 (p. 40): outlook: bt 5.0, mesh network
- ☐ 48 (p. 40): extend: combine dated data to reduce storage needs; incremental hash instead of sum
- ☐ 49 (p. 40): QoS control needed
- ☐ 50 (p. 41): 5-10%; outcome for an introduction-reader
- ☐ 51 (p. 41): describe how statements in the introduction were met: SMPC over self-forming, low maintenance/infrastructure-less network; framework offers reduced complexity, useful computation results for

many gamification settings and other applications; performance for real life usage with good levels of security; outlook: will gain more significance for SIoT: secure Internet of things (revisit topic of insecure Internet); next steps: host public as open source to gain to use open source community to improve project further