12 November 2014 Study · Web & Cloud · 30 min read

Secure Peer-to-Peer communication

Archived. This content is no longer updated. It remains available for reference.

Source https://technology2.a-sit.at/en/publications/secure-peer-to-peer-communication/

This project proposes a flexible and modular approach for existing peer-to-peer frameworks to enable a secure communication using well-established and proven protocols and algorithms. It introduces an interoperability layer where existing peer-to-peer frameworks and transport security protocols can be plugged in seamlessly and analyses the components of end-to-end security protocols. Finally a proof-of-concept application using the proposed framework is introduced which enables peers to establish secure connections with different types of identities. It can be found in the download area.

Proof-of-Concept Application

The download section contains three *P2P Secure Communication Application* packets. One common file, which includes all necessary libabries and two platform specific files, including the required JRE (Java 8 including JavaFX). All packets include two start scripts *run-client1* and *run-client2*. *run-client1* starts the master node and needs to be connected before *run-client2* is executed. Further details to the application can be found in the report document.

1. Introduction

Peer-to-peer networks are, due to their distributed nature, one of the most promising technology when talking about security and privacy on the Internet. They do not have a single point of failure, cannot be turned off with a single click and the generated traffic cannot be controlled by a single instance. But nowadays security and privacy in peer-to-peer frameworks is tailored to one specific framework or platform or is not considered at all. Looking at existing communication solutions shows that the terms *encryption* or *security* are used as buzz words to raise the confidence of the users. If encryption terminates at a central server and afterwards data is re-encrypted for the recipient(s), it is not end-to-end security, does not © A-SIT, Partner für Sicherheitsfragen protect against compromised servers and does not hinder operators to analyse transmitted data. For peer-to-peer networks various forms of categorizations exist: based on the purpose, centralized or fully distributed or based on the location of the resources. In the context of this project the location and fast discovery of resources is crucial. A resource could be, among others, data, generic endpoints, devices, or users. To achieve a fast communication from one node to another it is most

valuable to have a categorization of peer-to-peer networks based on the location of resources and distinguish between \emph{unstructured} and \emph{structured} peer-to-peer networks. Aberer K. [1] provides an overview of both concepts as illustrated in the remainder of this section. In unstructured peer-to-peer networks particular peers maintain a static amount of connections to other peers. The peers to connect to are discovered by regularly flooding the network with discovery requests and randomly selecting candidates to connect to. This results in a random distribution of connections among all peers. New peers connect to other well-known peers and on their own start a discovery process. The search performance in unstructured networks is rather low, as no peer is aware of resources provided by its linked peers. Therefore, searching for resources results in flooding the network with search requests. It further is not guaranteed that all peers receive a particular search request, as every request has attached some kind of time-to-live and a maximum number of hops. One of the most prominent representatives for unstructured networks is Gnutella [2], which mainly was used for file sharing from various clients. In structured peer-to-peer networks, peers are aware of the resources that neighbour peers offer. Therefore, searching and discovery in structured peer-to-peer networks is much more efficient, as requesting peers can determine in predictable time which peers offer a specific resource. One implementation concept for structured peer-to-peer networks is based on distributed hash tables. Distributed hash tables (DHT) provide a distributed and redundant key-value mapping among all peers of the network. In the context of structured peer-to-peer networks, each node and each resource has a unique resource identifier to uniquely identify the resource or node in the whole network. The resource gets registered in the DHT and associates information on how to reach the resource (e.g. which peer is in charge of the resource). Figure 1 shows the concept of a distributed hash table. The address space is separated in several areas (red area, green area, blue area) and each node in a particular area saves the whole key-value table for the corresponding area to provide redundancy and prevent data loss. The IDs in Figure 1 reference to node IDs, but every node could provide IDs for further resources which are associated with the corresponding area (not necessarily the same area as the maintaining node). The shown lines indicate established connections used for routing. Depending on the strategy of the implementation a node will maintain connections to neighbours and also to far areas to provide a fast connectivity with a minimized number of hops. In case a request is sent from node 1 to node 4, the request does not need to be broadcast to the complete network and there is no need to maintain a separate routing table. The sending node can simply retrieve all required information from the DHT and send the request to the nearest known node in the context of ID distribution.

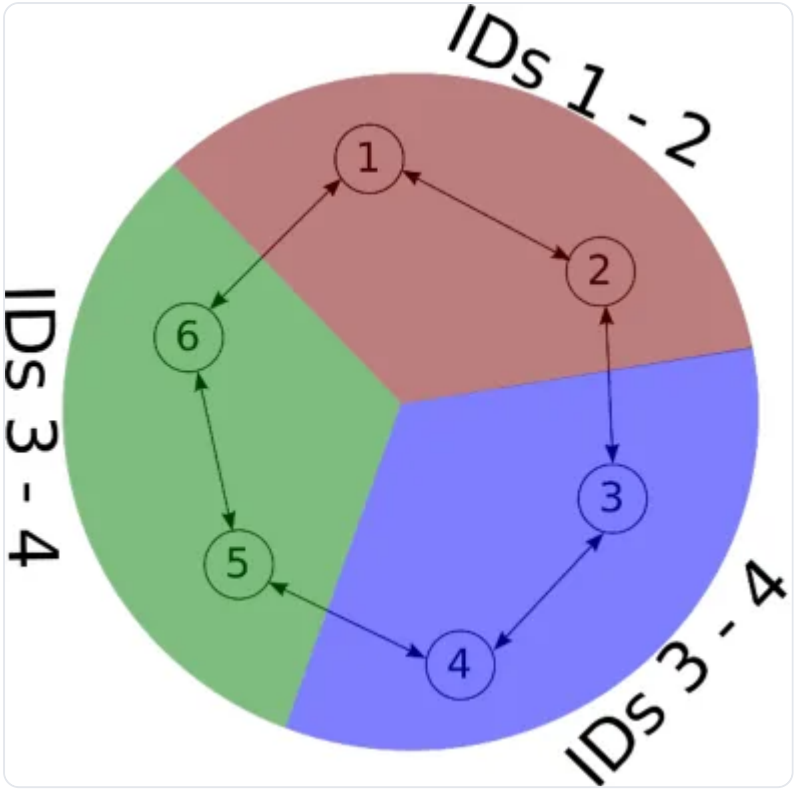


Figure 1: - Distributed hash table

The remainder of this paper is structured as follows. In Section 2 the requirements for a privacy and end-to-end security enabling layer are analysed. Section 3 gives an overview of other existing solutions and how these solution satisfy the requirements. Sections 4, 5 and 6 contain our contribution with a focus on the interoperability layer, end-to-end security and our approach to provide a flexible configuration interface. In Section 7 a proof-of-concept application utilizing SP2P is described. Before concluding in Section 9 some future perspectives are shown in Section 8.

2. Requirements

The requirements for a privacy and end-to-end security enabling layer are crucial and without a clear definition and rationale all following analysis and implementations are built on sand. Therefore, in this section a clear goal is defined and two major requirements for the resulting framework (*transport security* and *identity provisioning and authentication*) are identified and further elaborated. The goal of the research on this paper is to enable privacy and security in the form of a framework and interoperability layer for especially (but not exclusive) structured peer-to-peer networks, with the vision that security and privacy should not be a matter of experts, but for everyone.

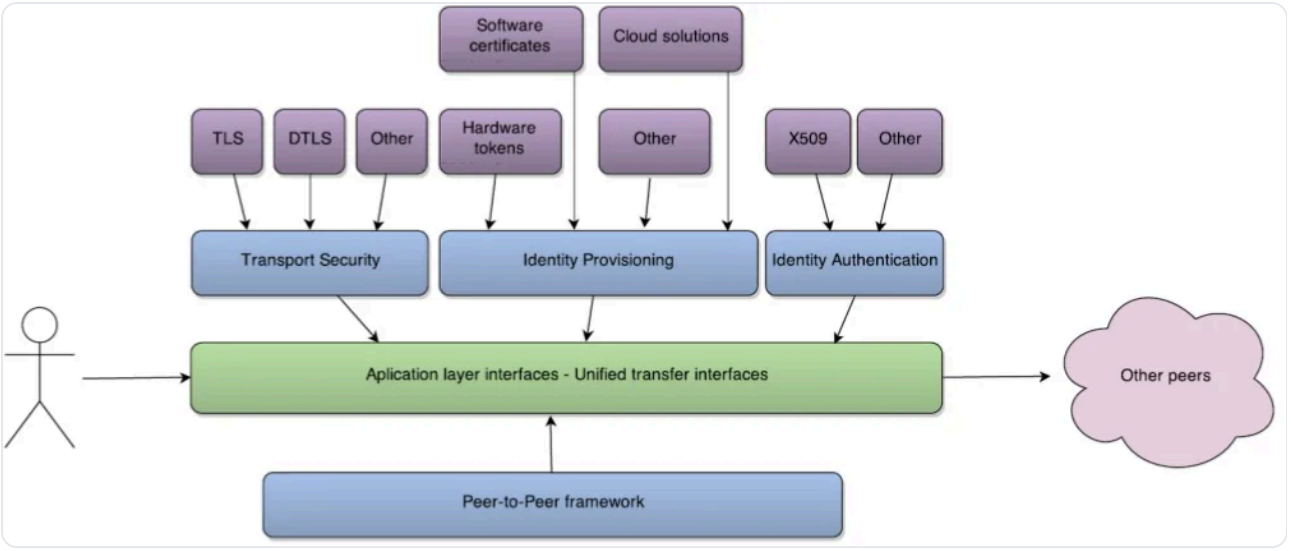


Figure 2: – Component overview Figure 2 shows identified components which are required to provide a privacy and end-to-end security enabled layer, their plug-able nature and the composition to the proposed framework and interoperability layer. In the center of the image, the application layer interfaces and the unified transfer interfaces are shown, as the only part that is tackled by application developers which forms the basis for the provided framework. They provide flexible means of (a) plugging in different peer-to-peer framework implementations and (b) provide several decoupled security related functions on top. A detailed description of the interoperability layer and its interfaces is discussed in Section 4. Transport security: Transport security is a well explored field, and several technical/cryptographic methods are available. End-to-end security can be provided at several levels of the OSI model as seen in Figure 3. IPSec is securing Internet Protocol (IP) communication over potential unsafe media. Secure Socket Layer (SSL)/Transport Security Layer (TLS) provides security on top of reliable communication protocols (e.g. TCP). Other protocols, for example to provide a Secure Shell (SSH) are operating on the Application Layer. Transport security in the context of peer-to-peer networks will always operate on the Application Layer, as peer-to-peer networks rely on other reliable or unreliable communication protocols. Another important topic when talking about end-to-end security is, how to securely provide ones identity and also how to verify it.

Identity provisioning and authentication: In this case identity is not necessarily a person's legal identity, it can also be an anonymous identity. Regardless which type of identity is used, the communicating parties need confidence about each other's identity and that they are really communicating with the same person as last time. Summarizing, for a layer providing seamless secure communication, not only the encryption parts are important, but also to be able to authenticate and have confidence of the identity of the communicating parties. With special attention on these requirements some of the existing solutions are analysed in Section 3 *Figure 3 - OSI Model*

3. Existing Solutions

Skype as one of the most popular communication and voice-over-IP applications today, was originally completely based on a peer-to-peer network, according to the analysis of Baset[2]. It was structured into interconnected super nodes, with potentially higher available bandwidth and ordinary end user hosts connected to one of the super nodes. The authentication took place by using a Skype login server (or multiple login servers). The analysis is based on an old version of the Skype client and protocol, the results state that each communication is end-toend encrypted using the Advanced Encryption Standard (AES) with a key of 256-bit in length, which also is considered secure nowadays. Taking a look at the current version of Skype and their website another picture is shown:

" *...we use TLS (transport-level security) to encrypt your messages between your Skype client and the chat service in our cloud, or AES (Advanced Encryption Standard) when sent directly between two Skype clients. Most messages are sent both ways, but in the future it will only be sent via our cloud to provide the optimal user experience.* "[3] As Skype uses a proprietary protocol it is not possible to conduct a full security analysis. The Bittorrent Chat, which was announced in December 2013, uses a DHT approach to locate other peers and their identity information [5]. There is not much information available yet, but according to the available information the connections are always made directly from sending peer to receiving peer, and the routing capability of the network is only used to locate the identity information. Each peer saves his public key in the DHT, which is later used for the establishment of a session key, but no information is available on how to protect or validate the public key (aka the identity). Other solutions exist which tackle parts of the aimed solution using different approaches. Tor

[6] for example, although not directly related to peer-to-peer networks, is described as an onion routing network based on a distributed overlay network to anonymize TCP-based application traffic by tunneling the traffic to the target by using other peers to obfuscate the origin of the transmission. Using the onion routing approach the originator of the transmission choses a path which seems suitable, and consecutively encrypts the transmitted data for all of the peers in reverse order, including some metadata. Using this approach no intermediary node is aware of the complete path and therefore is not aware of the originator or the target and is able to communicate with hosts she may not be able to communicate directly. Off-the

7	**Application Layer** SSH, Peer-to-Peer Security
6	**Presentation Layer** SSL/TLS
5	**Session Layer**
4	**Transport Layer**
3	**Network Layer** IPSec
2	**Data Link Layer**
1	**Physical Layer**

Record Messaging (OTR) is a protocol which enables existing messaging applications to have private conversations providing the following properties: encryption, authentication, deniability and perfect forward secrecy. Various Virtual Private Network (VPN) providers from other countries offer services for users around the globe to overcome the restrictions in their countries.

4. Interoperability Layer

The interoperability layer is one part of our contribution. On one end it provides the interfaces for developers using the framework, and on the other enables peer-to-peer frameworks to be plugged in. It provides means to (a) unify the specifics of a large group of different peer-to-peer frameworks and provide harmonized interfaces and (b) provides a pluggable component system to easily exchange components like Transport Security, Identity Provisioning and Identity Verification. The reference implementation is based on TomP2P [7]: TomP2P is a peer-to-peer library and a distributed hash table (DHT) implementation which provides a decentralized key-value infrastructure for distributed applications. Every node in the network has a unique ID, which is assigned randomly or can be chosen by the connecting party. The ID is abstracted by the framework as it might look different for different frameworks. The framework uses a packet-based approach which is also exposed at a higher level by the interoperability layer. The term “Application Layer Packet” is introduced which is nothing more than a packet which contains data, relevant for the application using the framework. To send an encrypted packet to a remote node the process is basically divided into three stages as seen in Figure 4:

Figure 4 - Connection establishment 1. First the initiator sends a discovery requests to the target, to get all the public identity information provided by the remote peer (refer to Section 5).

2. Then the initiator sends a " *setupConnection* " requests which basically starts the key negotiation process and agrees on a session key. During the agreement process multiple interactions with the initiator (or the user initiating the procedure) may be performed. The initiator also has the possibility to define constraints on the used remote identity, either specify a concrete remote identity ID, as received from the discovery request in step 1, or specify constraints like “only use X509-based identities issued from one specific certificate authority”.

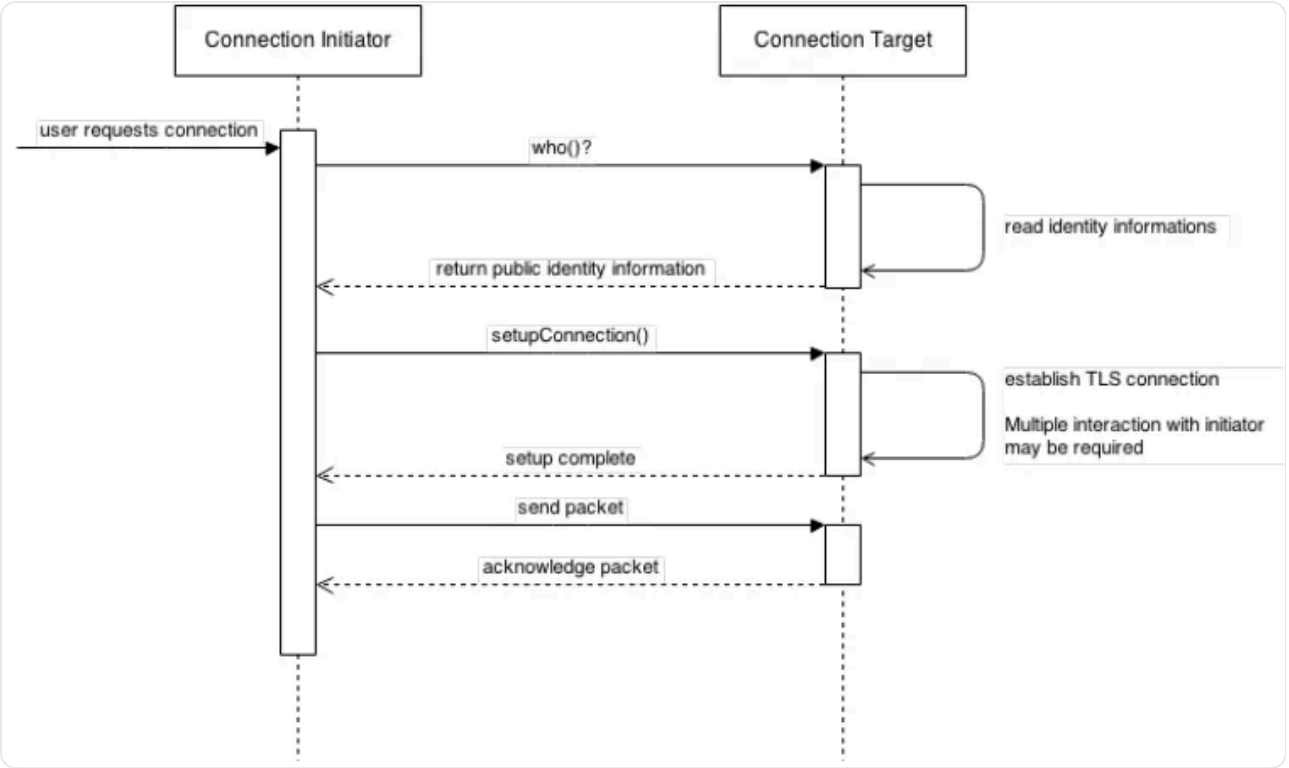


Figure 5: - Interoperability Layer To integrate other security protocols the implementer is only tied to the very basic interface from Figure 5, the communication required by the security protocol does not interfere with the functionality of the interoperability layer. For receiving data from remote peers, SP2P avoids the use of blocking calls and uses callback-based approaches to listen for data of specific peers.

3. Once the process is complete the application can use the transmission interface as shown in Figure 5 using the send operation and registering for notifications for received packets. The listing in Figure 5 is not complete and only contains the relevant methods required for abstraction.

Identities in the context of SP2P are defined very loosely by the *IdentityInfo* interface as seen in Figure 6. Identity implementations only need to have the functionality to (a) provide a unique identifier among all identities of a single peer and (b) provide a detailed summary of the identity (e.g. output X509 certificate details). This enables the framework to handle today's standards such as X509 certificates or PGP and also enables the framework to support other tokens in future versions. The rest is the responsibility of the *VerificationService* which is tied to one or multiple specific types of identity, and provides, as seen in Figure 6, methods for verifying the identity. The result of the verification process basically provides two evaluations:

1. The *ValidityStatus* evaluates the status at the time of evaluation of the identity among the following values: *VALID_AND_TRUSTED*: The provided identity is valid, a chain to a trusted root was built successfully and the identity is not revoked, *NOT_TRUSTED*: The provided identity is cryptographic valid, but no path to a trusted root could be built, *NOT_VALID*: The provided identity not valid and may have been altered without permission, and *EXPIRED*: The identity has expired or is not yet valid.
2. The *IdentityQuality* contains the asserted quality of the provided identity as specified in Section 5.3.

The current implementation of SP2P utilizes TLS and only supports X509 certificates, because support for e.g. TLS using PGP keys [8] is very limited and is seen as future work.

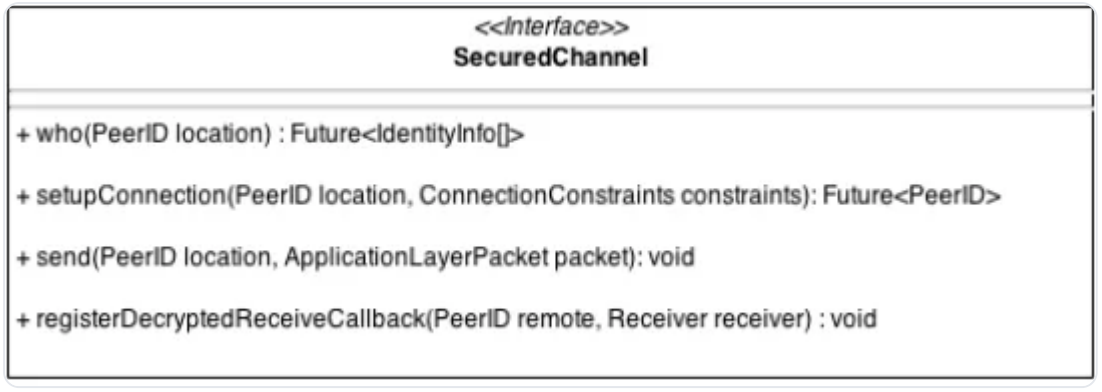
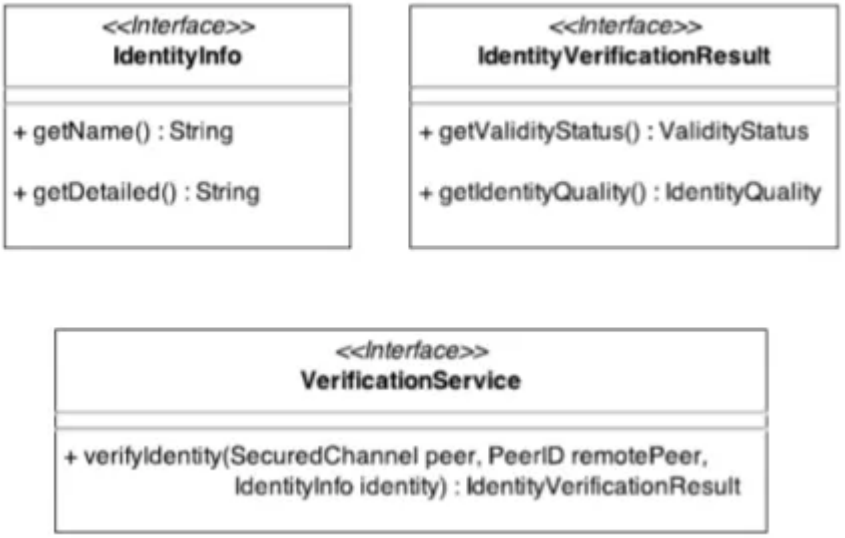


Figure 6: – Identity related interfaces



5. End-to-End Security

A further part of our contribution is to analyse and provide end-to-end security based on the interoperability layer as defined in Section 4. End-to-end security is not only a matter of utilizing secure key agreement protocols to establish a secure communication channel, but also to provide means to supply identities using tamper resistant hardware and securely authenticate the identities to have confidence of the communicating parties. The remainder of this section is organized in three subsections, each tackling one of the aspects Transport Security, Identity Provisioning and Identity Authentication.

5.1. Transport Security

The technology to provide end-to-end transport security is available and well-established. One prominent representative of end-to-end transport security is Transport Layer Security (TLS) [9]: TLS provides a privacy and data integrity layer between two communicating parties and is used to negotiate a particular encryption algorithm and cryptographic keys among two communicating parties. In the course of the research, it was agreed to focus on a single, secure and state-of-the-art algorithm to mitigate the risk of agreeing on a weak or broken algorithm. The SP2P framework currently exclusively uses the algorithm "*TLS_DHE_RSA_WITH_AES_128_GCM_SHA256*" and only operates with server and client authentication:

```
- The key exchange happens using the ephermeral Diffie Hellman (DHE) algorithm. One could also use a static Diffie Hellman (DH), but using DHE has the advantages (1) that the server key does not need to be a DH enabled key and (2) that the algorithm generates a new key pair for every connection which in fact provides perfect forward secrecy (PFS). The PFS property provides assurance, that even if the long-term key, where session keys are derived from, is leaked, recorded session data cannot be decrypted as the key exchange was protected by a temporary generated key pair.

- The symmetric encryption is performed using the Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) [10] [11]. GCM was chosen as it provides an authenticated encryption and achieves very high speeds in both, hardware and software implementations.
```

TLS is designed to operate on reliable transport protocols (e.g. TCP), which have the properties that each byte sent, receives exactly once and in the same order at the recipient. According to RFC6347 [12] connection unreliability creates the following problems for TLS:

“

1. TLS does not allow independent decryption of individual records. Because the integrity check depends on the sequence number, if record N is not received, then the integrity check on record N+1 will be based on the wrong sequence number and thus will fail. (Note that prior to TLS 1.1, there was no explicit IV and so decryption would also fail.) 2. The TLS handshake layer assumes that handshake messages are delivered reliably and breaks if those messages are lost. “ [4] Of course every other protocol beside TLS can be used, but TLS was chosen as it is wellknown and widely used. Although one drawback of TLS is, that only a mutual key-agreement is performed, meaning that no group key-agreement is possible which can be of interest for peer-to-peer networks.

We claim, running TLS over peer-to-peer networks introduces the problems of unreliable transport protocol and of intermediate nodes which may interfere or modify the communication, which need to be tackled: *Unreliable transport protocol:* Although the peer-to-peer framework of choice may be based on reliable transport protocols (e.g. TCP) it is still up to the peer-to-peer framework to provide a reliable node-to-node connection. If no reliable connection is available it is up to the interoperability layer to provide a reliable connection, or to switch to other protocols beside TLS. Due to the nature of peer-to-peer frameworks, that some nodes may not function as expected and a constant joining and leaving of peers is taking place in the network, sent packets, if not tackled by the peer-to-peer implementation, may not receive in order or even may not receive at all at the destination peer. The SP2P framework intentionally uses a simple Stop & Wait algorithm as illustrated in Figure 7 to provide a transmission reliability layer on top of the used peer-to-peer framework. Each packet is transferred on its own and waits for acknowledgement before sending the next packet. If one acknowledgement is not received after a given time-out the packet is resent and the error counter for the given packet is increased. After a certain amount of errors the packet is dropped and the error is passed to the higher layers. Although the algorithm is not efficient it was chosen due to its simplicity and may require extension in the future. Another option, and still stick to TLS, is to use Datagram TLS (DTLS) [12]. DTLS specifically is designed to provide TLS functionality on (unreliable) datagram networks. It

slightly modifies the TLS structures and adds sequence numbers and a retransmission time-out. At first sight this fits the use case better, we although opted for the first version, as support for DTLS in current libraries is not well elaborated yet:

```
- OpenSSL provides support for DTLS1.0 and rudimentary support for DTLS1.2, but as
we are, because of the selection of TomP2P [7], tied to the Java programming
language, and don't want to include platform specific dependencies we opted to not
go with OpenSSL.

- The Java Secure Socket Extension (JSSE) has no support for DTLS at all.

- In the first run we tried to go with Bouncy Castle, which provides rudimentary
support for DTLS, as long as it does not come to Diffie-Hellman and ephemeral key
exchange which is essential to provide a secure and Perfect Forward Secrecy(PFS)
enabled key agreement.
```

Intermediate nodes may interfere or modify the communication : One core concept of most peer-to-peer networks is, that every node can communicate with everyone, even if no direct connection is possible or allowed. This implies that traffic from one node to another uses the connectivity provided by a third node. As illustrated in Figure 1, traffic from node 1 to node 3 may flow over node 2, therefore node 2 has the possibility to alter the contents of the packet or even discard the packet. Regarding discarding packets the peer-to-peer framework may use multiple-paths or other flow control algorithms, which are out of scope of this document. Thanks to an integrity and privacy enabling TLS (or other) protocol the intermediary node cannot see the plain content of the transmission and all alterations are detected at the recipient and the packet is discarded. An even better option is to completely bypass nodes which are known to drop packets or try to alter the contents, but this needs to happen at the routing level of the peer-to-peer framework.

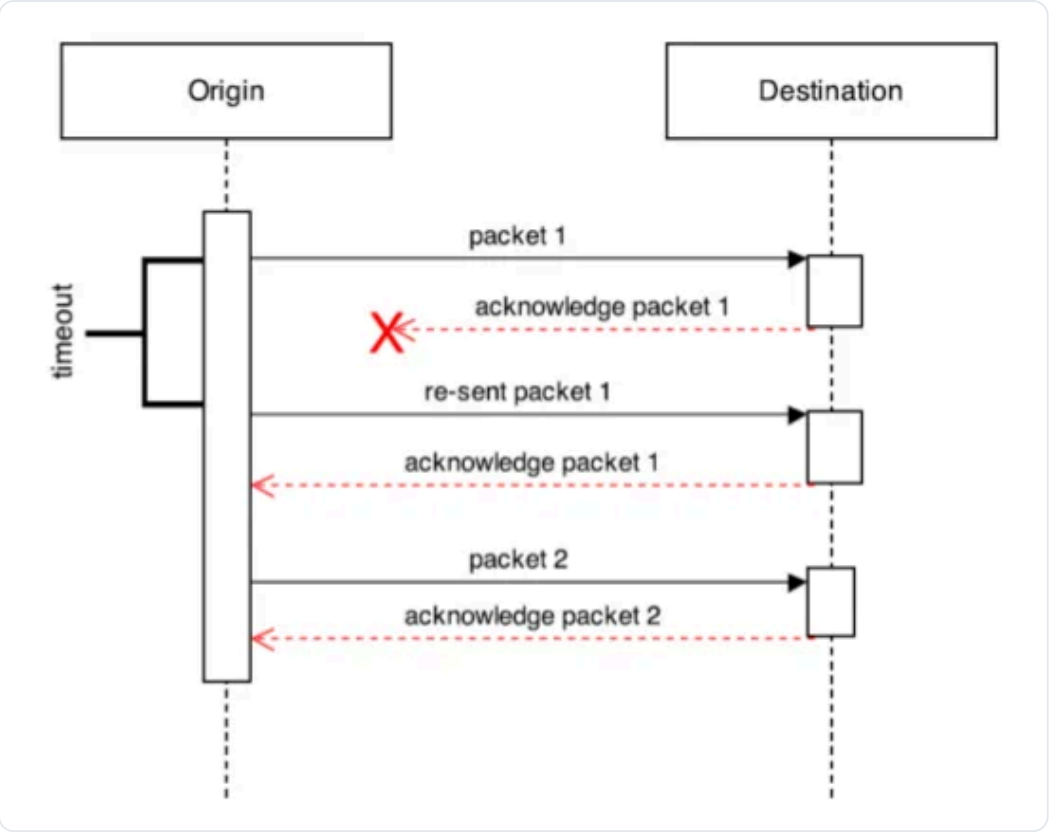


Figure 7: – Identity related interfaces

5.2. Identity Provisioning

Identity provisioning is the process of securely presenting one's identity to another party. In this context we define identities from two perspectives:

1. A digital identity which is directly related to a physical person, its associated data uniquely describes a person.
2. An anonymous digital identity which is directly related to a physical person, and the communicating peers have certainty that the identity is associated to a physical person and the associated key is subject to protection mechanisms, but the communicating peers cannot conclude on the person associated with the identity information.

The first case is a typical use case, at least for the identification parts, for nationally issued eID solutions (e.g. Austrian Citizen Card [13]). The key material is protected by smart cards or Hardware Security Modules (HSM) and can only be used by presenting the card holder's secret. Some of the deployed eID solutions only contain signature and encryption certificates, and may not contain certificates suitable for establishing TLS connections. Further, smart cards may require the user to enter her personal PIN for each operation performed on the smart card, which essentially includes every operation which involves the secret key. This significantly lowers the user-experience. This is where the emerging field of cloud based HSMs comes in, where the provider has much more fine-grained control on all those parameters. They may enable authentication using the eID solution (e.g. smart card based), and provide certificates and key functionalities with various properties. The keys, like when using smart cards, never leave the provider, and depending on the trust relationship from the user to the cloud-based HSM provider, can be considered safe, with a similar protection level than other solutions (like smart cards). Another consideration when

designing SP2P was, that one node may also provide multiple identities, appropriate to different groups of connecting peers. One for general purpose connection, where the remote peer does not need an association to your real, official eID, but still wants to be sure that your key is protected appropriate, and that she is always talking to the same person, and another identity that is linked to your official eID. This introduces some privacy concerns, as connecting peers may be able to crawl identity information from peers all over the network, therefore every particular identity information can be marked with different flags:

```
- “ _invisible_ ”: The provided identity information is not enumerated on discovery
requests from other nodes. They can only be directly referenced by the provided
ID..

- “ _acknowledge_ ”: The provided identity information needs explicit permission
from the user to be used for authentication by a specific peer..

- “ _userInteraction_ ”: Identity information marked with this flag, require user
interaction before the certificate can be retrieved or before the key can be used (
e.g. for cloudbased HSMs where authentication is required). This enables the
framework to perform for example certificate caching to avoid unnecessary user
interactions.
```

Finally end-to-end security is not only a matter of using decent key-agreement and encryption algorithms but also on providing secured identity information and verification. TLS uses X509 certificates on the server and client side to provide identity information to the other peers, but as described in Section 4 the framework is not limited to X509, it currently is constrained by TLS (and the available implementations) but could also use other technologies and methods, if compatible with the end-to-end security protocol.

5.3. Identity Authentication For the authentication of provided identities, the framework provides interfaces (as described in Section 4) to plug in different authentication systems, as a simple certificate validation and revocation checking procedure may not be sufficient.

Multiple standards defining assurance levels are already available. With STORK QAA[5] four assurance levels are defined: *(1) No or minimal assurance*, *(2) Low assurance*, *(3) Substantial assurance* and *(4) High assurance* . With ISO29115[6] also four levels of assurance are defined, similar to the levels defined in STORK QAA. With eIDAS[7] the levels *low*, *substantial* and *high* are defined. The framework does not address a specific quality level assurance framework, but leaves the decision up to the integrators. As a general guideline we propose to stick to frameworks with four levels of assurance or to omit the lowest level which generally does not provide any confidence in the identity. The following is a general proposal for an assignment of identity types to a four level based assurance framework, but itwill of course depend on the details of the framework:

```
- Level 1: Username and password authentication

- Level 2: Certificate based authentication where the key material is stored (and
generated) locally

- Level 3: Cloud-based HSM where a level 4 token is used to authenticate

- Level 4: Smart card based (or similar systems) identity provisioning utilizing
tamper resistant hardware.
```

In the course of this project we do not assert the lowest level, as no username-password authentication is present and in addition assert to *Level 2* for remote entities storing identities (e.g. cloud-based HSMs) only using username-password authentication.

In the end the assertion is dependent on the authentication service, which may enable the user to e.g. define different cloud-based providers as highly trustworthy and therefore assert to *Level 4* . As a general rule we can say that *Level 4* always involves tamper resistant hardware devices, for providing the relevant key material to establish the secure communication channel or to authenticate at a trustworthy entity.

6. Configuration

For a broad distribution and user acceptance of an application or framework it is key to provide a user friendly, simple and extensible configuration interface. Our approach is to provide a configuration interface which accepts simple JSON formatted configuration files as illustrated in Listing 1. The file is divided into two sections, one for the basic connectivity of the underlying framework which therefore depends on the used peer-to-peer framework, and a securityrelated section containing all the definitions of available identities. In this case, as seen in Listing 1 the peer gets assigned a unique identifier and a locally unique TCP/UDP port to be reachable by other peers. To be part of an existing peer-to-peer network the TomP2P framework requires at least one bootstrap node to connect to, and to be able to discover other nodes and anchor in the DHT as illustrated in Figure 1. In this case the security section just defines a single default file-based identity, but it opens up a flexible way of including identities from smart cards or other (cloud-based) HSMs. Multiple flags, as defined in Section 5 can be applied and add properties to the identities.

{ “connection”: { “peerId”: {“type”:“string”, “value”:“client1”}, “port”:1235, “bootstrap”:[{“peerId”: {“type”:“string”, “value”:“master_peer”}, “url”:“tcp:localhost:1234”}] }, “security”:

[{ “id”:“p12”, “flags”: [“default”], “type”:“p12”, “keystore”:“configuration/test_user_1.p12”, “keystorePass”:“test”, “alias”:“test user 1’s iaik id” }] }

7. Proof-of-Concept Application

The proof-of-concept application is a classical chat application. It utilizes the JSON style configuration as illustrated in Section 6.

![- Pending key verification Figure 9 - Pending key usage authorization As the selected identity constraints require a specific key from the remote peer which is protected by the flag " acknowledge “, the remote peer application asks the user for consent of using the key as shown in Figure 9. After acknowledgement both peers verify the provided identity information as shown in Figure 10. Beside performing a certificate validation and revocation checking the " Verification Service " performs an additional verification of the identity information. The current implementation limits to ask the user about the trust state. As future work the connection to trusted verification services is planned, which also take into account aspects as described in Section 5.3. After positive verification at both sides the connection is established and the peer view is presented to the users as seen in Figure 11. The view is composed of three sections. The first one shows the local and the remote identity, the second is used for sending messages to the remote peer and the third show the raw communication as sent over the network. The application is not production ready and is meant to show the basic working of the provided framework.](abbildung-8.webp “Figure 10: - Pending key verification Figure 9 - Pending key usage authorization As the selected identity constraints require a specific key from the remote peer which is protected by the flag " acknowledge “, the remote peer application asks the user for consent of using the key as shown in Figure 9. After acknowledgement both peers verify the provided identity information as shown in Figure 10. Beside performing a certificate validation and revocation checking the " Verification Service " performs an additional verification of the identity information. The current implementation limits to ask the user about the trust state. As future work the connection to trusted verification services is planned, which also take into account aspects as described in Section 5.3. After positive verification at both sides the connection is established and the peer view is presented to the users as seen in Figure 11. The view is composed of three sections. The first one shows the local and the remote identity, the second is used for sending messages to the remote peer and the third show the raw communication as sent over the network. The application is not production ready and is meant to show the basic working of the provided framework.”)

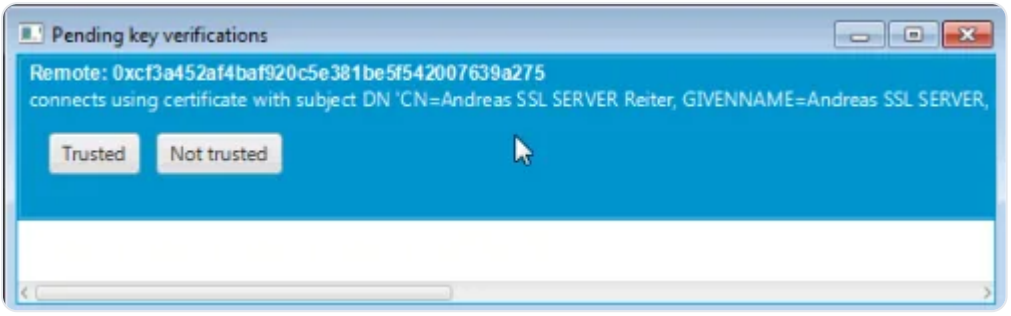
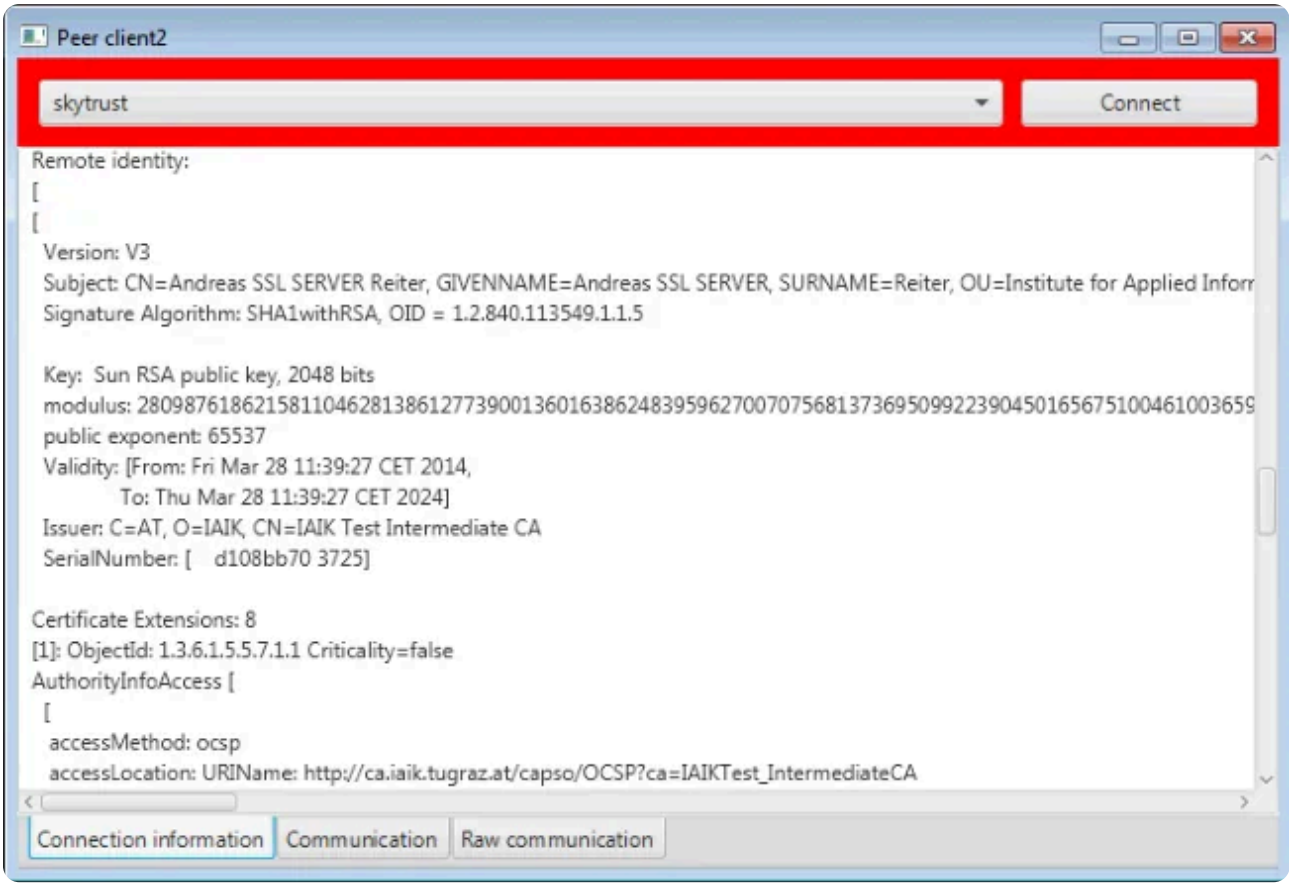


Figure 11: - Peer view with established connection

![- Application main window The main window of the application is shown in Figure 8. The main element is a list which contains all other added peers. New peers can be added by entering in the provided field. By clicking on the foot-line of the window, a peer overview opens which shows all discovered peers and their connectivity information. On double-clicking on a list-entry the peer-view opens which shows the connection-status to the peer and the public available identity information. For this show-case we manually add an identity constraint by selecting " add manual identity constraints " and entering the name of a remote constraint which requires user acknowledgement. Clicking " Connect " starts the TLS process.](abbildung-10.webp “Figure 8: - Application main window The main window of the application is shown in Figure 8. The main element is a list which contains all other added peers. New peers can be added by entering in the provided field. By clicking on the foot-line of the window, a peer overview opens which

shows all discovered peers and their connectivity information. On double-clicking on a list-entry the peer-view opens which shows the connection-status to the peer and the public available identity information. For this show-case we manually add an identity constraint by selecting " add manual identity constraints " and entering the name of a remote constraint which requires user acknowledgement. Clicking " Connect " starts the TLS process.")



8. Future Work

Peer-to-peer frameworks are best suited to not only support single-peer to single-peer communication but also to enable a multi-peer communication (e.g. group chat, conference call, file sharing...). For the security protocol parts we only focused on TLS, which is not capable of establishing session keys between more than two endpoints. Therefore, future work should go into the direction of implementing mechanisms (and possibly also extend the provided interfaces) in a way to enable multi-peer key agreements. Another session key related topic is the support for PGP or other mechanisms, as currently SP2P totally relies on X509 certificates. Mavrogiannopoulos et al. [8] specify the use of OpenPGP keys for TLS authentication, this is one option, although we think that research should definitely consider the key agreement among multiple peers. The focus on TLS also introduces the problem of reliability on transport level. Currently SP2P uses a Stop & Wait algorithm as illustrated in Section 4, more efficient and a boost in performance would be some kind of sliding window algorithm where multiple packets can be transmitted without waiting for the acknowledgement. Another option are selective repeat algorithms where multiple packets are transmitted without acknowledgement and the receiver can specifically request to retransmit erroneous packets. This is not only of interest for the security protocol, but also for the developer using the framework and knowing that the transmission is reliable. On the other side research and implementation aspects should go in the

direction of providing security and key- agreement algorithms based on unreliable networks and with the possibility of packet loss. For video- or audio-streaming applications for example, it may not be of highest importance that each packet is received, and the codec may be able to handle a certain amount of packet loss without the user recognizing.

9. Conclusion

In this paper SP2P - a flexible and modular interoperability layer for existing peer-to-peer frameworks is proposed. It enables secure communication using well-established and proven protocols and algorithms. Special attention is put on the TLS protocol to analyse the problems and benefits of using TLS over (possible unreliable) peer-to-peer frameworks. Further end-toend security as a whole, composed of transport security, identity provisioning and verification is analysed with the conclusion that end-to-end security is not only a matter of using decent key-agreement and encryption algorithms but also on providing secured identity information and verification. In the course of the research activities, a security enabling framework for peer-to-peer frameworks with an adapter for TomP2P [7] was created, which provides the ability to establish secured connections from one node to another using TLS with different types of identities. Further an interoperability layer is proposed, providing application layer and unified transfer interfaces. It provides a plug-able infrastructure to plug-in transport security algorithms, provide different types of identities and according to the type of identity provide different types of verification services. The verification service extends the standard certificate validation process by a certificate quality dimension from the ISO 29115 standard [14]. It defines four levels of assertion which are integrated in SP2P. The framework can easily be extended to support other protocols than TLS, which may also support unreliable transport for applications like audio- or video-streaming. By using and providing adapters for SP2P, peer-to-peer framework developers gain the advantage that they can use proven and well-established security mechanisms and algorithms out of the box. It raises the overall utilization of cryptographic methods and works against the current massive data collection practices.

10. References

[1] Aberer, K., Cudré-mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., & Schmidt, R. (2003). P-Grid : A Self-organizing Structured P2P System, 32(5005), 29–33.

[2] Services, C. D. S. (n.d.). The Gnutella Protocol Specification v0.4.

[3] Baset, S. A., & Schulzrinne, H. G. (2006). An analysis of the Skype peer-to-peer internet telephony protocol. In Proceedings - IEEE INFOCOM. doi:10.1109/INFOCOM.2006.312

[4] Does Skype use encryption? (n.d.). Retrieved August 12, 2014, from https://support.skype.com/en/faq/FA31/does-skype-use-encryption

[5] Update on BitTorrent Chat. (2013). Retrieved August 12, 2014, from http://engineering.bittorrent.com/2013/12/19/update-on-bittorrent-chat/

[6] Dingledine, R., Mathewson, N., & Syverson, P. (2004). Tor: The second-generation onion router. SSYM'04 Proceedings of the 13th Conference on USENIX Security Symposium, 13, 21. doi:10.1.1.4.6896

[7] TomP2P. (2014). A P2P-based high performance key-value pair storage library. Retrieved August 05, 2014, from http://tomp2p.net

[8] Mavrogiannopoulos, N., KUL, & Gillmor, D. (2011). RFC 6091 - Using OpenPGP Keys for Transport Layer Security (TLS) Authentication.

[9] Dierks, T., & Rescorla, E. (2008). RFC 5246 - The transport layer security (TLS) protocol Version 1.2. In Network Working Group, IETF (pp. 1–105).

[10] Mcgrew, D. A., Drive, W. T., Jose, S., & Viega, J. (n.d.). The Galois / Counter Mode of Operation (GCM).

[11] Salowey, J., Choudhury, A., & McGrew, D. (2008). RFC 5288 - AES Galois Counter Mode (GCM) Cipher Suites for TLS. In Network Working Group.

[12] Rescorla, E., RTFM, I., Modadugu, N., & Google, I. (2012). RFC 6347 - Datagram Transport Layer Security Version 1.2.

[13] Leitold, H., Hollosi, A., & Posch, R. (2002). Security Architecture of the Austrian Citizen Card Concept. In Proceedings of 18th Annual Computer Security Applications Conference (ACSAC'2002), Las Vegas, 9-13 December 2002. pp. 391-400, IEEE Computer Society, ISBN 0-7695-1828-1, ISSN 1063-9527.

[14] ITU-T Recommendation X .1254 | International Standard ISO / IEC DIS 29115 Information technology — Security techniques — Entity authentication assurance framework. (2011).