Design of a Secure Mobile Business Communication Platform Utilizing Next Generation Web Technologies

Hannes Stadler
Stadler ITS
Wassermannstr. 14
D-96052 Bamberg, Germany
Email: h.stadler@exxor.de

Marcel Großmann, Udo R. Krieger
Computer Networks Group
Otto-Friedrich-Universität
D-96047 Bamberg, Germany
Email: {marcel.grossmann | udo.krieger}@uni-bamberg.de

***Abstract*—We describe the client-server architecture and browser-to-browser groupware functionality of a secure mobile business communication platform that utilizes advanced web technologies by means of HTML5 and powerful JavaScript frameworks. The design is based on a real-time, peer-to-peer multimedia communication paradigm and adopts WebRTC as basic software architecture of the communication layer. To guarantee confidentiality, integrity, and authenticity of the communication processes and to avoid the strict dependence on existing, purely TLS based solutions, it integrates a newly designed, client-centric security concept with a strict end-to-end encryption of all data flows. We elaborate on the multi-layer architecture of this new software platform that can be applied to support a variety of e-business services in a cloud environment including a heterogeneous population of mobile clients.**

***Keywords*-Business Communication, e-Business Platform, Secure Groupware, HTML5, WebRTC, Meteor.js, Node.js**

I. Introduction

Nowadays, individual collaboration services like social networking, e.g. by Facebook[1] or instant messaging services like WhatsApp[2], and real-time e-business collaboration services like Skype for Business[3] or Bitrix24[4] are evolving very rapidly. They are based on an advanced application logic, which is implemented by client-server and/or peer-to-peer (P2P) architectures using modern web services as application platforms and middleware.

Regarding business scenarios, besides the efficiency and high performance requirements of the data transport processes it is of utmost importance to guarantee confidentiality, integrity, and authenticity of the underlying communication processes and data privacy of the users. These objectives yield stringent service requirements reaching far beyond efficient browser-to-server or browser-to-browser communication patterns of those interactions among different individuals.

To study the possibilities of next generation web technologies evolving from HTML5 and its new multi-media capabilities, a secure groupware platform based on WebRTC has been developed and implemented by a prototype called WappKey [1], [2], [3]. Its design principles and software architecture are described in this paper and illustrated by the prototype WappKey. The development has been based on a systematic design methodology and a market analysis. At the starting point of the development in 2014 the groupware platform was expected to include the following most important service features:

- a simultaneous, efficient real-time communication including different media like text, audio and video
- synchronous and asynchronous instant messaging
- archiving of encrypted messages
- encrypted data transfer among browsers all the time
- use of individual buddy lists
- efficient support of parallel users on the platform even when running on very light weight server hardware
- management support for user administration and independent administration of profiles by the users
- support of the federation of several platforms to enable enterprise wide cooperation and decentralized IoT use cases
- secure integration of a heterogenous population of mobile clients
- easy porting to native applications for Android, iOS or Windows Mobile

The latter items were not covered to full extent by other commercial tools at the time of development. Currently offered commercial or open source collaboration services for mobile environments with improved security profiles like Telegram[5], Threema[6], SMSSecure[7], or Open Whisper Systems' app Signal[8] are most often lacking the possibility to run on-premise and are not restricted to JavaScript frame-

[1] https://www.facebook.com/
[2] https://www.whatsapp.com/
[3] https://products.office.com/de-de/skype-for-business/
[4] https://www.bitrix24.de/
[5] https://telegram.org/
[6] https://de.wikipedia.org/wiki/Threema
[7] http://smssecure.org/
[8] https://whispersystems.org/

works or P2P real-time browser communication. The former requires their users to fully trust a foreign infrastructure, makes security auditing hard or even impossible and denies the option to implement them in custom IoT scenarios.

To our best knowledge, the presented well-structured multi-layer architecture of the prototype WappKey provides one of the first modular software platforms that allows the efficient construction of flexible, secure e-business applications on a WebRTC framework with extensible security features and its integration into a cloud environment. It can be used to support a variety of e-business services including smart home, e-health, and industry 4.0 contexts and to implement their components effectively on a heterogeneous population of mobile and stationary clients.

In this regard, the developed secure groupware platform provides a unique starting point for experimentation. Due to page limitations a first performance assessment could not be included in this exposition. It can help us to investigate the underlying software concepts and frameworks, to identify their weaknesses, and to develop enhancements with a significant better energy and performance profile that avoids current processing bottlenecks on mobile devices with ARM processors.

The paper is organized as follows. First, we discuss the software architecture of the secure groupware platform. Then we present the main features of the security layer and the user interface components. Finally, some conclusions and an outlook on future extensions are presented.

II. Software Architecture of a Secure Groupware Platform

We present the software architecture of an experimental groupware platform called WappKey that implements business-oriented communication processes. It has been developed in a systematic manner according to a spiral model with seven design phases inspired by the iterative, incremental agile software development methodology SCRUM [4]. The former process has incorporated a market analysis, a systematic technology assessment as well as a risk analysis based on a SWOT concept [5].

The design of the platform is based on a layered software architecture that comprises four stacked stratums provided by different design patterns that are available in related JavaScript frameworks: a service and API stratum as application layer on top, a web-server sublayer and a communication sublayer as middleware, and a cryptographical stratum combining additional presentation and session layer functionalities above the TCP/UDP/IP based transport protocol stack (see Fig. 1). This hierarchically structured, modular design enables the simple replacement of functionalities or frameworks, the efficient development, maintenance, and enhancement of the system by the loose coupling and strong cohesion among the provided modules [4].

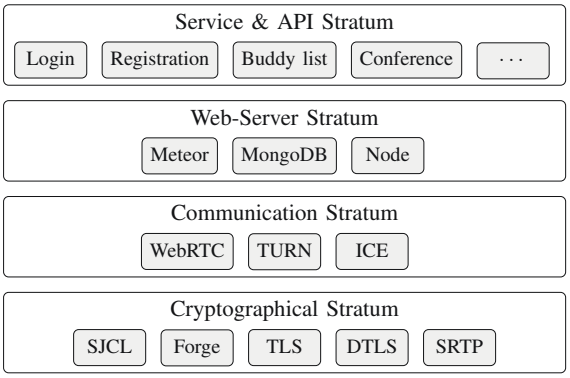

Figure 1. Layering of the secure groupware platform.

A. Server-side and Client-side Software Structures of the Application Program Interface Stratum

The information structure and system-theoretical basis is provided by next generation web technologies based on HTML5 and innovative JavaScript web application frameworks [2]. It is driven by enterprise scenarios of advanced web services that require the dynamic interworking of clients and three-tier web-server architectures, which integrate application logic and database functionality of classical or non-standard DBs (see Fig. 2).

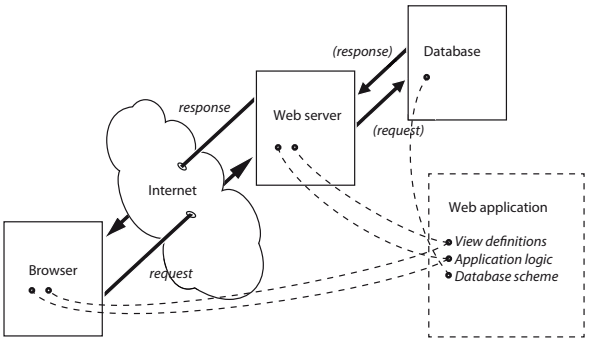

Figure 2. Structure of web services.

The developed client-server web service architecture has adopted JavaScript both as server-side and as client-side application programming paradigm as well as HTML5 to specify the views [3]. A server-side architecture based on event-driven programming, asynchronous communication, and non-blocking I/O-API has been adopted to avoid contention and to guarantee efficiency of the processing. Consequently, the *Node.js* [6] framework has been selected as light-weight runtime environment and JavaScript wrapper to implement the business logic of an application at the server-side since it allows a stand-alone web-server operation, optimizes the throughput, and supports the scalability of real-time web applications.

JavaScript frameworks at client-side enable the support of complex control processes peering the corresponding server-side logic. To support the *Model-View-Controller* (MVC) and *Model-View-ViewModel* (MVVM) paradigms [7], *Angular.js* in combination with *Meteor* has been selected as basic client related web application framework. It enables the implementation of complex interaction patterns of the application processes.

B. Software Structure of the Middleware Layer

It is the main objective of the developed groupware platform to provide secure business communication. To achieve this goal, the platform integrates efficient real-time multimedia communication directly among browsers based on a HTML5 setting and a P2P overlay network that supports this interaction paradigm. Furthermore, a purely client-centric security concept is realized that guarantees the confidentiality, integrity, and authenticity of the data flows during a session among multiple user devices.

The server infrastructure of the platform with all required centralized functionalities of the system has been implemented by means of an Ubuntu server but could also run on any other platform, which supports *Meteor*[9].

1) Communication Stratum: To realize these ambitious goals, a communication stratum has been specified as sub-layer of the middleware. It provides the required session, data transport, and signaling functionalities of the P2P overlay network that is established on demand among the interacting users. For this purpose we have adopted WebRTC [8] with its fully specified controls of the media plane and its related protocol-agnostic signaling concept as basic protocol framework. Its components provide the software architecture of the user APIs to interact dynamically in real time in a browser-to-browser fashion, such as the access to locally available multimedia content by the `getUserMedia`[10] primitive [1]. WebRTC supports the connection-oriented multimedia communication along a logical channel between two browsers which can be controlled by the rich methods available at the `RTCPeerConnection` API[11] (see Fig. 3). In the prototype design the communication endpoints of a connection have been encapsulated by a `WebRtcConnection` object with individual identifiers of the caller, callee, and the conference itself. It provides an adequate abstraction of such a virtual connection among communicating entities along a logical channel accessed by WebSockets.

Applying WebRTC, it is possible to control the signaling of an established session between the peers following a request-response principle by the JavaScript Session Establishment Protocol (JSEP) and its generic view on the internal signaling state machine in terms of the parameter interfaces `createOffer()`- and `createAnswer()` [9] (see Fig. 4). Latter are invoked to exchange the SDP descriptors of a connection, also [10]. In the prototype WappKey the

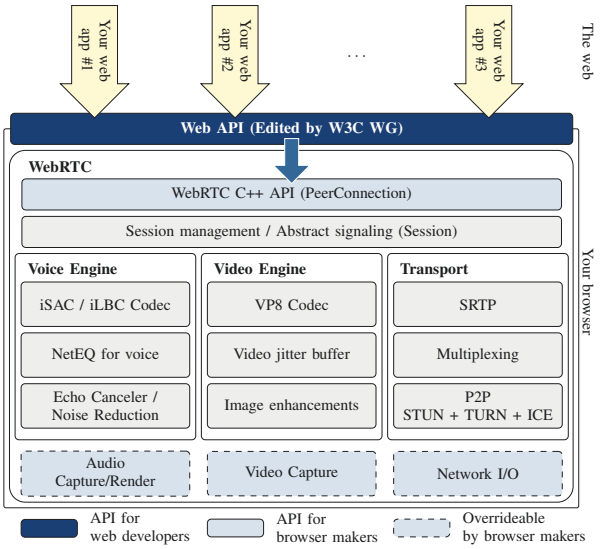

Figure 3. Structure of WebRTC[11].

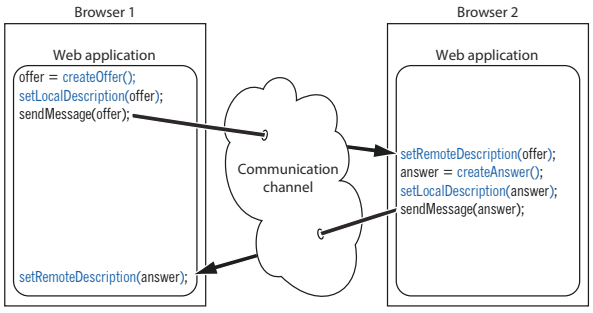

Figure 4. JSEP API used during the connection setup of a WebRTC session.

WebRTC communication sublayer has been realized by a custom implementation of the WebRTC APIs. It is suitable to construct the WebRTC communication channel utilizing *Meteor* and *Node.js* as JavaScript wrapper.

ICE in combination with a *TURN* server have been incorporated in the prototype to handle the signaling across different domains protected by firewalls [11].

2) Web-Server Stratum: The web-server stratum provides web application development, user management, data management and coordination functionalities for the com-

[9]see https://github.com/meteor/meteor/wiki/Supported-Platforms
[10]http://www.w3.org/TR/webrtc
[11]cf. https://webrtc.org/architecture/

munication stratum and the application service and API stratum exclusively at the server-side of the architecture. It utilizes the rich collection of functions given by the *Meteor.js*[12] JavaScript framework. The latter is built on top of Node.js. It supports the development of mobile web applications and integrates *WebSockets*[13] as interface of the message transport. At client and server side, the functionality specified within `Meteor.methods({ ... })` blocks is invoked by corresponding `Meteor.call(methodName, paramter, callback)` primitives.

Moreover, Meteor has the advantage to incorporate the not only SQL (NoSQL) database *MongoDB*[14] with its document oriented structure as data management platform. Furthermore, it supports applications based on the MVC-MVVM-paradigm that require an active data exchange with a server. The provided functions for a continuous, automatic synchronization between the server and a cluster of clients as well as the conflict management and latency compensation mechanisms are used as basic functions by the groupware platform. The latter relies on a *Distributed Data Protocol* (DDP)[15] for the query of structured data and an update of the server-side database and its synchronization among the associated clients. The incorporated control mechanisms are based on a publish–subscribe pattern that uses `Meteor.publish`, `Meteor.subscribe` primitives to disseminate the stored data automatically. The access rights to those data source known as Meteor collections can be controlled by the primitives `Connection.deny, Connection.allow` and are applied to manage the authorization processes among the interacting clients. The bidirectional data flow to the server by WebSockets enables the event-driven updating of state in Meteor, e.g. by database changes, and enables the live updating and dynamic rendering of views known as *full stack reactivity*[12].

The Meteor server assigns unique identifiers to the clients and their conferences and maintains it during a session life time. Any user interaction between their clients is bound to a unique conference and exclusively realized within this abstraction of the client interaction. During the collaboration phase clients create and maintain permanent WebSocket connections to the Meteor server that are realized and protected by HTTPS connections utilizing TLS. They are used to exchange all required status information of a groupware session with the server and its associated functional elements.

Furthermore, the new groupware platform offers additional management functionality to realize a federation among different WappKey servers along a permanent connection safeguarded by means of TLS. Cooperation functions for control, data and management plane such as connection management, key exchange management including the access to public keys of users on the remote domain, and monitoring functions are provided here by corresponding methods of the programming framework.

[12]https://www.meteor.com/

[13]cf. https://tools.ietf.org/html/rfc6455

[14]http://www.mongodb.org/

[15]https://github.com/meteor/meteor/blob/master/packages/ddp/DDP.md, https://www.meteor.com/ddp

III. Architecture of the Cryptographical Stratum

To guarantee the strong confidentiality, integrity, and authenticity of the communication processes required by business applications and data privacy of the users, the developed software architecture of the groupware platform integrates a newly designed, mainly client-centric security concept with a strict end-to-end encryption of all data flows. It avoids a strict dependence on existing TLS based solutions with its weaknesses. It is a major objective of the design to guarantee that only encrypted data can leave the browser and all communication processes among the clients and with the groupware server are transported along channels safeguarded by HTTPS. Block ciphering based on temporary valid symmetric keying is applied to achieve confidentiality and authenticity of the message exchange processes that are generated during the interaction of the conference participants.

On the platform only those cryptographical procedures are applied that guarantee the *Perfect Forward Secrecy* (PFS) principle [12]. For this reason a Diffie-Hellman (DH) key exchange protocol is applied as public-key scheme to determine the session keys with limited lifetimes among interacting clients [13]. A session key is used for the encryption of the user data during such information exchange on the groupware platform. Hence, this temporary DH-generation of the session key allows both to authenticate the peers, that intend to interact in a new session, and to secure the confidentiality of the user data exchange processes. The key generation and parametrization process is decoupled from the TLS transport mode. The corresponding prime has been selected originally long enough to avoid a compromising of the exchange process that has been discussed recently in the literature [14].

The first usage phase of the platform with the user registration and later the login to the server by a user requires the bidirectional information exchange between the corresponding client and the groupware server with its registration and login server functionalities. It is exclusively realized by a HTTPS connection. An asymmetric cryptographical scheme is used to support the digital signature of data, the authentication of the interacting client and server entities, and the integrity of the data transport during the registration and login phases.

The prototype utilizes the *Stanford JavaScript Crypto Library* (SJCL[16]) for symmetric encryption by block ciphers

[16]http://crypto.stanford.edu/sjcl/

[15] and the *Forge* project[17] for TLS and asymmetric encryption techniques under JavaScript since it is optimized for Node.js applications. Then the data transport between the browsers is considered to reach a sufficient level of security in combination with HTTPS protected channels [16]. As the latter library is only accessed via an abstraction, its replacement by stronger implementations is always possible.

A. Registration of a Client at the Server

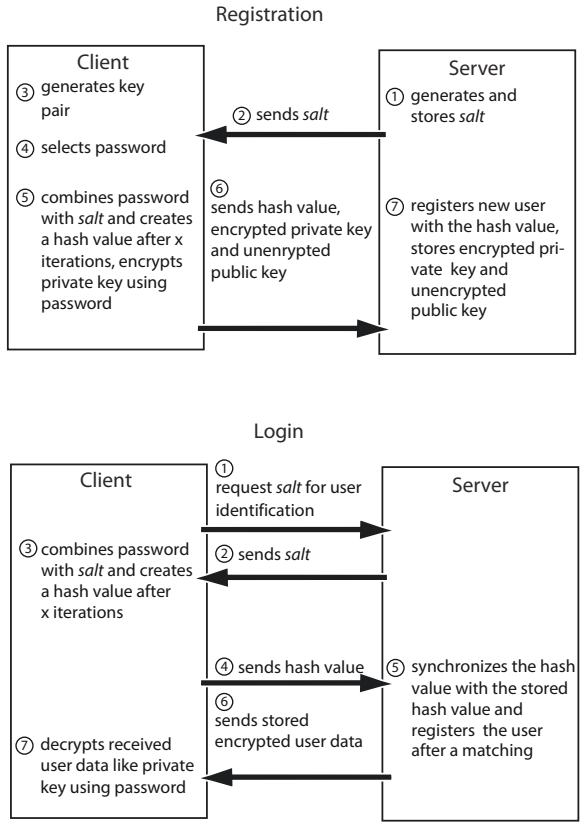

Figure 5. Registration and login processes of a client.

The registration process (see Fig. 5) is executed at the client-side by the JavaScript logic that has been received from the server. It is applied to generate a pair of public and private keys in random fashion. The further process requires the transfer of the public key of the user, the private key that is encrypted by a block cipher, and a hash value over the HTTPS safeguarded channel to the registration server of the platform. The latter key information is generated exclusively by the client, whereas the generation of the hash value incorporates a selected user password and a random information element (salt) that has been transferred to the client beforehand by the server. In this way the new user is registered at the platform. The transferred information element is stored at the server-side in an account associated with the identity of the user to identify its client during the following login processes.

[17]https://github.com/digitalbazaar/forge

B. Login of Clients to the Server

The login procedure (see Fig. 5) relies on a point-to-point key update methodology. It mimics a key establishment protocol using a key transport with challenge-response and symmetric encryption. On request of a registered client the login server sends a random information element (*salt*) over a HTTPS safeguarded channel to the user's browser. The latter uses the information, that stems from the encrypted private key of the user and the user password, and the server's salt to generate a hash value. This procedure is implemented by the Meteor method `Meteor.loginWithPassword`.

A message with this token is returned over the HTTPS safeguarded channel to the login server. After a matching of the hash values, the user is authenticated and logged into the system. The latter returns the encrypted user data, which the client decrypts by the help of the user password.

At any time the developed protocol guarantees that neither the private key of a user nor the user password are transferred in unencrypted manner to the server-side. Furthermore, the overall concept ensures that the secret password is supplied only once by a user to the system to perform a login. In addition, the encryption of user information exclusively at the client-side prevents the ability of a decryption at the server-side.

In this regard a very high level of confidentiality and trustworthiness of the communication processes is guaranteed and the vulnerability at the server-side and therefore also against attack scenarios, where the attacker already compromised the intranet or comes from within company ranks is minimized.

C. Securing the Data Exchange of Clients

If two peers have to exchange data across the P2P overlay network, they first agree both on a common session key using the DH key exchange protocol. Thus, the required information exchange is secured by encryption and authenticated. These session keys are stored exclusively at the client-sides and expire after a session tear-down to obey the PFS principle.

All user data that a user stores permanently on the platform in the MongoDB database are encrypted applying block ciphering and using the secret private password of the user. In this way we prevent the decryption of user data at the server-side or storing of accessible security information at the infrastructure sides. Therefore, compromising activities on the user data at the server-side, e.g. by system

administration staff or attackers, which already compromised the intranet, are made more difficult and data privacy is achieved.

The transfer of the user data in the P2P overlay network can be secured by the application of the SRTP and DTLS session layer protocols below the WebRTC layer [17].

IV. User Interface Structure of the Groupware Platform

The user interface of the groupware platform WappKey is built by HTML5 and uses a responsive design. It allows the adaptation to the display size and supports both desktop and mobile end devices with width smaller than 768 pixels by different menu bars.

A. User Interfaces

The prototype offers a home page for login and 4 different interfaces for those users that have logged into the system, namely, a profile management, a buddy list, a conference list, and the view of a single conference (see Fig. 6). The conference functionality offers the

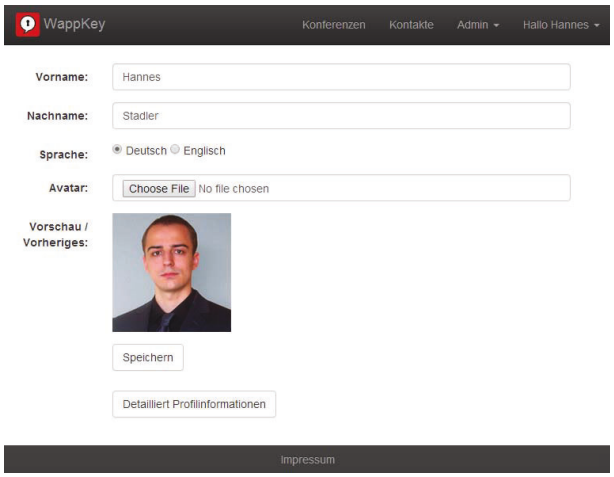

Figure 6. User view on profile management.

sharing of multi-media content including separate audio and video streams generated by an attached camera and microphone. The screen may be shared as video stream among conference participants and a selective sharing of audio exclusively is possible. Messaging among conference participants is supported by two methods with different types. An *Only online* (nur online) mode encrypts a message using the session key previously constructed by DH. It can be decrypted only by online users and is depicted as elusive message (flüchtige Nachricht). An *Online & offline* transport mode triggers a permanent message that is encrypted by means of an asymmetric keying scheme using the public keys of all users (see Fig. 7). The message

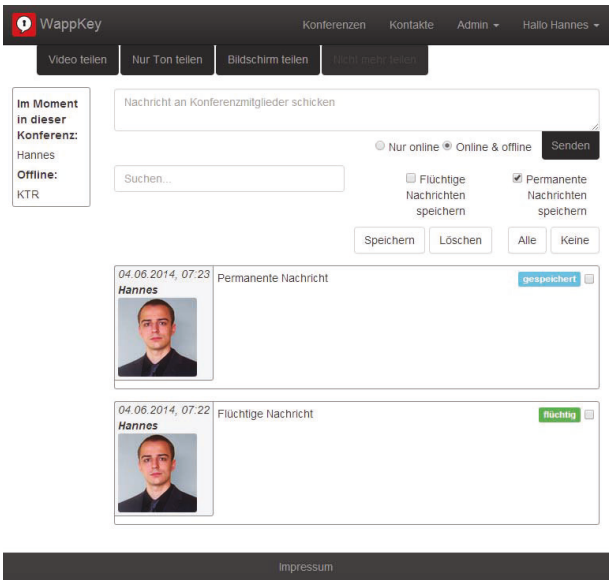

Figure 7. User view on a conference.

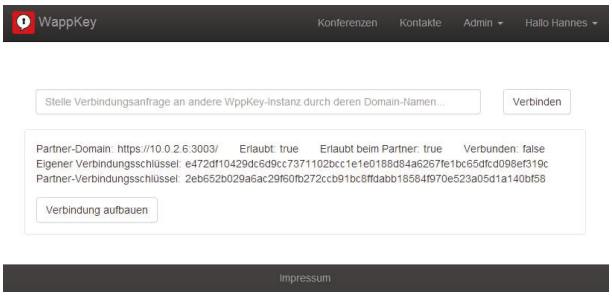

Figure 8. Administrator view of the connection management.

management offers filtering by supplied criteria and the permanent saving of elusive and permanent messages. In latter case the message is encrypted by means of a symmetric keying scheme using the secret password of a user. Local erasure is possible, but sent messages cannot be deleted at storages of others. Business extensions enable the exchange and archiving of files.

B. Management Interfaces

The groupware prototype offers two additional views for administrators, namely, a user monitor depicting all registered users and a connection management for other WappKey instances (see Fig. 8).

It enables the request to a complete domain by the *Connect* (Verbinden) method. In the initiator view it generates status information of the generated own key of the connection and the peer connection key at the callee side. An active called peer can now synchronize the keying along an independent signaling channel and authorize the connection

request after a successful authentication of the requestor. Then caller and callee are enabled to establish a connection in their user and signaling planes. Triggered by a setup method (Verbindung aufbauen) (see Fig. 8), a common connection between the authenticated users is instantiated in the signaling plane and after that both associated platforms automatically synchronize their basic user data. Thereafter, all common user and conference interactions are enabled across both domains of the different platforms. One WappKey instance can connect to multiple other instances in this fashion. In this regard, it is also perfectly suitable for other decentralized use cases or IoT, which is a subject of future research.

V. Conclusion

We have described the client-server architecture and browser-to-browser groupware functionality of a secure business communication platform that utilizes advanced web technologies by means of HTML5 and effective client-side and server-side JavaScript frameworks. The design is based on a real-time, peer-to-peer multimedia communication paradigm adopting WebRTC as basic protocol architecture and API software structure. To guarantee the confidentiality, integrity, and authenticity of the communication processes and the data privacy of the users, we have also integrated a newly designed, mainly client-centric security concept with a strict end-to-end encryption of all data flows. It avoids a strict dependence on existing, purely TLS based solutions with their weaknesses.

We have elaborated on the multi-layer architecture of this new software platform WappKey that can be applied to support a variety of e-business services in a cloud environment and include a heterogeneous population of mobile and stationary clients.

We are convinced that the proposed design offers a very broad potential to other application areas like smart home, e-health, and industry 4.0 contexts. The investigation of those items is part of our future research and development.

More information on the business product line of WappKey is available at URL: https://stadler-its.de/page/projekte

References

[1] A. Bergkvist, D. C. Burnett, C. Jennings, and A. Narayanan. (2013) WebRTC 1.0: Real-time Communication Between Browsers. W3C Working Draft 10 September 2013. [Online]. Available: http://www.w3.org/TR/2013/WD-webrtc-20130910/

[2] M. Pilgrim, *HTML5*. Up and Running. O'Reilly, 2010.

[3] R. Berjon, S. Faulkner, I. Hickson, T. Leithead, E. D. Navara, E. O'Connor, and S. Pfeiffer. (2013) Html5.1. A vocabulary and associated APIs for HTML and XHTML. [Online]. Available: http://www.w3.org/TR/2013/WD-html51-20131029/

[4] J. Goll, *Methoden und Architekturen der Softwaretechnik*, 1st ed. Vieweg+Teubner, 2011.

[5] H. Mintzberg, *The Rise and Fall of Strategic Planning*. The Free Press, 1994.

[6] S. Tilkov and S. Vinoski, "Node.js: Using javascript to build high-performance network programs," IEEE Internet Computing, Vol. 14, No. 6, pp. 80–83, 2010.

[7] I. Minar. (2012) MVC vs MVVM vs MVP. [Online]. Available: https://plus.google.com/+AngularJS/posts/aZNVhj355G2

[8] Google Inc. (2014) WebRTC. [Online]. Available: http://www.webrtc.org/

[9] J. Uberti and C. Jennings. (2013) Javascript Session Establishment Protocol. Internet Engineering Task Force. [Online]. Available: http://tools.ietf.org/html/draft-ietf-rtcweb-jsep-05

[10] C. Jennings, J. Rosenberg, J. Uberti, and R. Jesup. (2011) RTCWeb Offer/Answer Protocol. Internet Engineering Task Force. [Online]. Available: http://tools.ietf.org/html/draft-jennings-rtcweb-signaling-01

[11] J. Rosenberg. (2010) Interactive Connectivity Establishment. A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. Internet Engineering Task Force. [Online]. Available: http://tools.ietf.org/html/rfc5245

[12] D. Fox, "Perfect Forward Secrecy (PFS)," DuD - Datenschutz und Datensicherheit, 11, p. 729, 2013.

[13] J. Schwenk, *Sicherheit und Kryptographie im Internet*, 3rd ed. Vieweg+Teubner, 2010.

[14] D. Adrian et al. "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice," CCS'15, October 12—16, 2015, Denver, Colorado, USA, 2015.

[15] E. Stark, M. Hamburg, and D. Boneh. (2009) Symmetric Cryptography in Javascript. Stanford University. [Online]. Available: http://crypto.stanford.edu/sjcl/acsac.pdf

[16] Matasano Security. (2013) Javascript cryptography considered harmful. [Online]. Available: http://www.matasano.com/articles/javascript-cryptography/

[17] E. Rescorla. (2014) WebRTC Security Architecture. Internet Engineering Task Force. [Online]. Available: http://tools.ietf.org/html/draft-ietf-rtcweb-security-arch-09