Security and Privacy in Unified Communication

THOMAS REISINGER, ISABEL WAGNER, and EERKE ALBERT BOITEN, De Montfort University, Cyber Technology Institute (CTI)

The use of unified communication; video conferencing, audio conferencing, and instant messaging has skyrocketed during the COVID-19 pandemic. However, security and privacy considerations have often been neglected. This article provides a comprehensive survey of security and privacy in Unified Communication (UC). We systematically analyze security and privacy threats and mitigations in a generic UC scenario. Based on this, we analyze security and privacy features of the major UC market leaders, and we draw conclusions on the overall UC landscape. While confidentiality in communication channels is generally well protected through encryption, other privacy properties are mostly lacking on UC platforms.

CCS Concepts: • **Security and privacy → Domain-specific security and privacy architectures**; *Web protocol security*; *Denial-of-service attacks*; • **Information systems → Web conferencing;**

Additional Key Words and Phrases: Unified communication, video conferencing, audio conferencing, instant messaging, cloud service, security, privacy, STRIDE, LINDDUN

ACM Reference format:
Thomas Reisinger, Isabel Wagner, and Eerke Albert Boiten. 2022. Security and Privacy in Unified Communication. *ACM Comput. Surv.* 55, 3, Article 55 (February 2022), 36 pages.
https://doi.org/10.1145/3498335

1 INTRODUCTION

In April 2020, the Zoom video conferencing platform reached over 300 million daily meeting participants, compared to 10 million in December 2019. Similarly, Microsoft Teams reached 200 million daily meeting participants in April 2020, during the beginning of the COVID-19 pandemic. These examples emphasize the extent to which video conferencing is now used, to stay in contact with friends, family, and colleagues [127, 141].

The term ***Unified Communication*** **(UC)**[1] has evolved over time with changes in technologies, market trends, and user requirements. UC started in the 1980s with the trend of replacing circuit-switched telephone systems, provided by telephone companies with **private branch exchange**

[1]There are variations of the term UC including Unified Communications and Collaboration (UCC), Unified Communication & Conferencing, and many more, but from a technology viewpoint the differences are minor.

Authors' address: T. Reisinger, I. Wagner, and E. A. Boiten, Cyber Technology Institute (CTI), De Montfort University, Gateway House, Leicester LE1 9BH, United Kingdom; emails: thomas.reisinger@my365.dmu.ac.uk, {isabel.wagner, eerke.boiten}@dmu.ac.uk.

0360-0300/2022/02-ART55 $15.00
https://doi.org/10.1145/3498335

(**PBX**), with in-house telephone systems. In a further step, PBXs were replaced by **Voice over IP** (**VoIP**) systems and later moved to a software architecture instead of purpose-built hardware components by using existing **Internet Protocol** (**IP**) networks with greater flexibility and cost-saving. Over time these software-based voice systems merged and integrated with other systems such as e-mail clients, presence information, and **Instant Messaging** (**IM**). The wide deployment and availability of mobile devices and smartphones introduced additional challenges into UC environments, including extended user communication options with additional operating systems and challenges related to mobile networks with jitter, packet loss, and bandwidth limits. In recent years, the hosted and cloud-based UC architectures such as **UC as a Service** (**UCaaS**) and web-based conferencing gained popularity, in similarity to other IT areas that operate architectures "in the cloud" instead of on-premises.

Motivation. Early idealist users of the Internet saw it as a technology that would support liberty and democracy. The Arab Spring in the early 2010s is one example of the importance of social media and other internet-based technologies involvement in political changes. Internet and mobile technologies were shown to support communication, association, and organization, with governments a step behind in discovering and interfering with such communications. The use of such technologies is not limited to periods of intense political change; journalists, human rights activists, political activists, and minority groups are also likely to make use of these technologies. Popular social media platforms such as Facebook, Twitter, and WhatsApp are used frequently by these social groups, but may not be designed to meet their requirements, for example, with regard to encryption and anonymity.

Security and privacy features are often not the main focus for providers of UC platforms. Many platforms rely on closed-source code and their services are only accessible via public clouds. As a result, users cannot, for example, verify the implementation of end-to-end encryption and key management, but instead need to trust the claims made by the platform provider [92, 127, 141].

Contributions. Many surveys of UC focus on comparing functionality, for example, minimum video resolution, telephony dial-in support, integration capabilities with other systems (interoperability), recording of meetings, desktop sharing, or meeting controls [19, 123]. Other surveys focus on the jurisdiction, terms of use, and **General Data Protection Regulation** (**GDPR**) compliance of UC architectures [12, 28, 29]. However, to the best of our knowledge, there is no systematic analysis of UC security and privacy issues that discusses requirements, threats, and mitigations. This article closes this gap by presenting a scientific survey of *security and privacy* features of widely used and known UC architectures. In particular, this article makes the following contributions:

(1) We provide an introduction to UC based on a generic UC architecture and give a detailed description of its elements.
(2) We conduct a systematic privacy and security analysis of this generic architecture, following the STRIDE and LINDDUN methodology, to uncover the security and privacy risks in UC systems and known mitigations for them.
(3) We analyze 10 major UC platforms, comparing their features against the identified privacy and security risks and mitigations.
(4) We draw conclusions about the major remaining threats in individual systems as well as in the sector as a whole, and outline directions for future research.

The remainder of this article is organized as follows: Section 2 introduces UC and the generic architecture. The security and privacy analysis of the generic UC architecture is presented in Section 3, followed by a description of the UC landscape (Section 4) and the analysis of 10 major UC platforms (Section 5). In Section 6, we discuss our findings and conclude in Section 7.

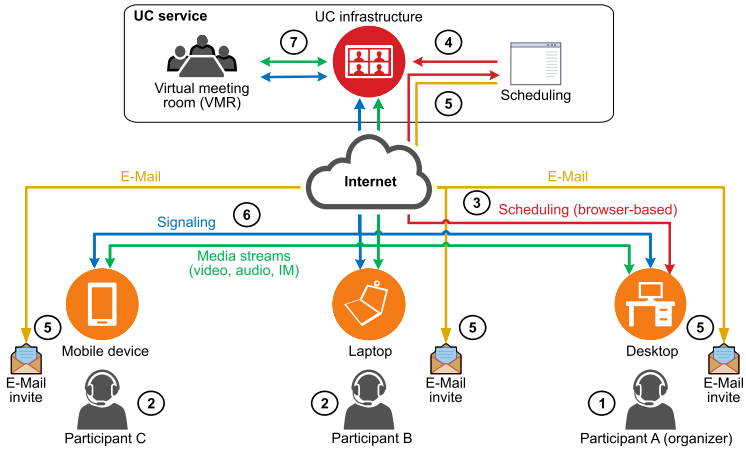

Fig. 1. High-level generic UC architecture—multipoint.

2 A GENERIC UNIFIED COMMUNICATION ARCHITECTURE

UC can be defined as "The interactive use of different real-time communication methods/channels with the integration of applications and processes across multiple devices and media types presenting a consistent unified user interface" [36, 90]. The communication channel can carry different types of information such as voice, video, content sharing, IM, and user presence information.

Besides the components exposed to the end-user, i.e., the UC client or interface, UC systems rely on server-side infrastructure to handle call control, registration, and coordination of the communication channels. This UC infrastructure can either be deployed externally, by a private or public cloud provider, hosted on premises, or mixed in a hybrid model. Figure 1 shows a high-level architecture for a UC multi-party video conference. In this example, one user invites two participants to join a scheduled meeting on a centralized UC infrastructure, using audio and video as communication channels on their desktop, laptop, or mobile device. The example shows a software-based architecture, running in a web browser, without the need for additional software plug-ins or specific hardware.

2.1 Scheduling

Participant A (1) uses a web browser on a desktop system to schedule and invite two other participants, B and C (2), to a meeting on a centralized web-based scheduling platform (3). After participant A completes the planning on the scheduling platform (4), invites are sent out from the platform via e-mail or another communication channel such as **Short Message Service** (**SMS**). The invitation is sent to all participants with the necessary meeting information such as subject, time, message, and joining instructions (5). The reference to the meeting can be a hyperlink or a calendar entry in a format such as iCalendar [25]. At the same time, the meeting is set up on the UC infrastructure (7). The e-mail recipients can add the invite to their calendar system and when the time comes, join the meeting by clicking on the invitation or by following the specified joining instructions.

2.2 Joining a Call

During the joining process, the participant's web browser uses the signaling protocol (6) of the UC infrastructure (7) to establish a call to the centralized UC service and to meet in a **virtual meeting**

room (**VMR**) (7). For authentication of the participants and the presenter/organizer, a PIN (e.g., a six-digit number) could be generated by the scheduling system and shared with the participants. The authentication information is often included in the invite by the scheduling system. After all the participants have joined the VMR the meeting can begin.

2.3 Centralized vs. Decentralized Architecture

A multipoint call (P2M), in contrast to a peer to peer call (P2P), can host more than two concurrent users on a **Multipoint Control Unit** (**MCU**), which handles the media sessions and mixes the different communication channels (e.g., audio and video). A multipoint architecture provides better scalability and additional media transcoding capabilities between different audio and video codecs. However, P2P architectures have the advantage that the media, and part of the signaling is exchanged directly between participants, which means that centralized call control is needed only to organize the call.

3 GENERIC SECURITY AND PRIVACY ANALYSIS

The generic UC architecture described above is subject to a large number of security and privacy threats. In this section, we use STRIDE and LINDDUN to systematically analyze these threats.

3.1 Context for Analysis

3.1.1 Use Scenarios. The users of our generic UC architecture are regular computer users who need not be familiar with the technical components of the UC architecture. No administrator or operator is needed to set up and join a meeting, and the whole process can be realized as an end-user self-service. The number of participants in a meeting can range from two to ten or more, and participants do not need to be from the same organization. Audio, video, and content (screen sharing) are used as communication channels.

3.1.2 External Dependencies. The minimum requirements for the user to use the UC architecture are a computer or mobile device with a supported web browser, a microphone, an Internet connection and optionally, and a camera. Most available UC systems depend on a web browser plug-in or dedicated client to be installed. The requirement to install additional software may be avoided for a better end-user experience and to reduce operating system dependencies.

Web Real-Time Communication (**WebRTC**) is a real-time communication technology, which provides UC services (voice, video, and generic data) built in to most modern web browsers, including Google Chrome 28+, Mozilla Firefox 22+, Safari 11+, and Opera 18+. WebRTC is implemented via JavaScript **Application Programming Interfaces** (**APIs**). For native clients, libraries are available for Android and iOS applications that provide the same functionality [39]. The **World Wide Web Consortium** (**W3C**) and the **Internet Engineering Task Force** (**IETF**) standardized WebRTC as open standard [11]. However, a well-known security and privacy issue is that WebRTC exposes the user's private and public IP addresses regardless of the use of VPN or anonymous network such as Tor. This exposure is caused by WebRTC utilizing the **Interactive Connectivity Establishment** (**ICE**) protocol to solve the problems associated with firewalls and NAT (firewall traversal of UC communication), and results in WebRTC silently leaking peer IP addresses without the user's permission [1, 94, 113, 124]. This issue can be solved by using a browser extension or gateway that examines the WebRTC communication for suspicious requests and asks the user for permission to continue. The disadvantage of using a browser extension is the requirement for individual implementations per browser type, and the disadvantage of the gateway approach is an additional delay for the web communication caused by the verification process [30].

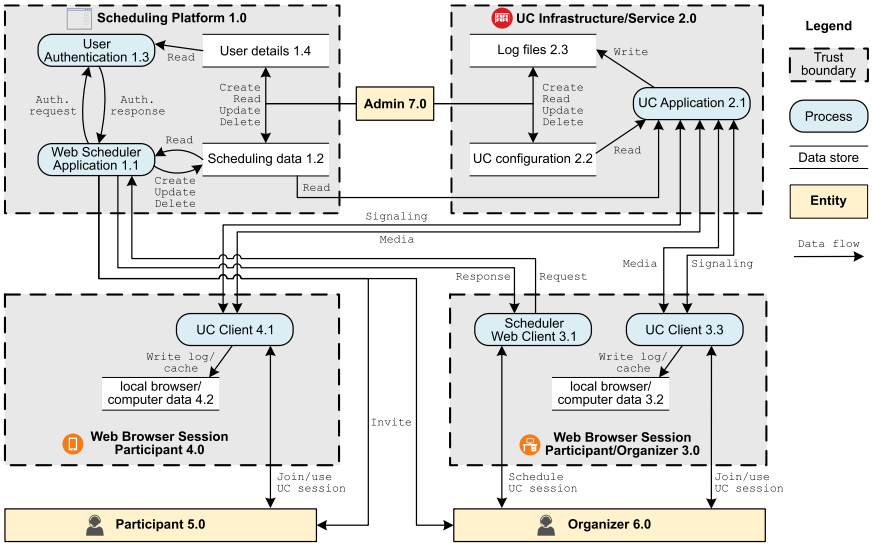

Fig. 2. Data flow diagram (DFD) for a generic UC architecture with multipoint participants.

The UC architecture itself may be hosted in a public or private cloud environment to remove the complexity of operating the various components on-premises and to provide better scalability and resiliency, but deployments on-premises are possible.

We focus our analysis on threats against the UC architecture and its components, and therefore consider out of scope, threats based on a compromise of the underlying hardware or operating system, threats such as social engineering, and intentional or accidental administrator actions. Examples of this include taking pictures of messages and stolen, lost, or hacked devices. We follow the Internet Threat Model guidelines and assume that the attacker has complete control of the communications channel between end systems [96]. Passive attackers are able to read from the network, and active attackers can also write to the network.

3.2 Data Flow Diagram (DFD)

A **dataflow diagram** (**DFD**) is an essential instrument for security and privacy analysis [58] p. 225ff. Figure 2 depicts the dataflow diagram for our generic UC architecture. Entities such as participants or administrators are drawn as filled rectangles, data stores as parallel lines, trust boundaries as dotted line rectangles, processes as filled ovals, and dataflows as arrows. Elements of the DFD are numbered hierarchically, for example, the Scheduling Platform 1.0 contains Web Scheduler Application 1.1, Scheduling details 1.2, User Authentication 1.3, and User Details 1.4.

Trust boundaries are demarcation points in the architecture that describe areas within which system components trust each other, e.g., the processes or dataflows within the trust boundary [115] p. 13. Dataflows between trust boundaries need to be analyzed for correctness and for assurance that no sensitive information is leaked to a potential attacker.

The following describes the DFD components in Figure 2 in more detail, starting with the three main entities, *Participant (5.0)*, *Organizer (6.0)*, and *Administrator (7.0)*. *Participant (5.0)* is a regular computer user joining a UC call on the *UC Infrastructure/Service (2.0)* by receiving an invitation, e.g., via e-mail, from the *Scheduling Platform (1.0)* or ad-hoc using a *WebRTC/UC Client (4.1)* on his computer/mobile device. Participants may belong to the same organization/group or be external. The *Organizer (6.0)* who invites other participants in most cases will have additional UC meeting

control permissions such as the ability to mute all and disconnect participants. *WebRTC/UC Client (4.1)* stores information related to UC calls such as connection details, cache, logs, and potentially sensitive information such as participant names in the data store *local browser/computer data (4.2)*.

The second entity type, *Organizer (6.0)*, is similar to the *Participant (5.0)*, but can additionally schedule UC meetings with a *Scheduler Web Client (3.1)* and invite additional participants via the *Scheduling Platform (1.0)*. Organizers can create, read, update, and delete the *Scheduling details (1.2)* after a successful authentication via *User Authentication (1.3)*.

The third entity type, *Administrator (7.0)*, is responsible for the administration of the *Scheduling Platform (1.0)* and *UC Infrastructure/Service (2.0)*. The administrator has full access to the relevant data stores, *User details (1.4)* and *Scheduling details (1.2)*, within trust boundary *Scheduling Platform (1.0)*. These data stores may contain security and privacy relevant user profile information in *User details (1.4)* (e.g., user/participant name, e-mail address, and authentication credentials) and UC meeting relevant information in *Scheduling details (1.2)* (e.g., who is meeting whom and when, meeting connection information, meeting access codes, and meeting subject). Within trust boundary *UC Infrastructure/Service (2.0)*, the administrator has full access (create, read, update, and delete) to the data store *Log files (2.3)*, which may contain security and privacy relevant information similar to 1.2 and 1.4, as well as technical details and metadata regarding UC meetings (e.g., participant's client IP address, browser used, operating system, and hostname). Data store *UC configuration (2.2)* contains UC system relevant technical configuration details, which are administered by the *Administrator (7.0)*.

The process *UC Application (2.1)* within trust boundary *UC Infrastructure/Service (2.0)* is responsible for handling UC calls with WebRTC clients, including signaling and media (video, audio, etc.), based on the *Scheduling details (1.2)* and the configuration of the service in *UC configuration (2.2)*.

3.3 Threat Modeling

We follow two threat modeling methodologies, STRIDE and LINDDUN, to identify possible security and privacy threats against the generic UC architecture, and to identify generic mitigation strategies. We focus on generic threats against a generic UC architecture, i.e., we do not model threats for a concrete UC system. As a result, we neither discuss concrete attacker motivations and capabilities, nor do we quantify the impact of any attack.

3.3.1 Security Threats. We follow the STRIDE methodology to understand the potential security threats to the generic UC architecture as well as possible mitigations [47, 105]. STRIDE looks at six threat types from an attacker's point of view. ***S**poofing* threats occur when an attacker can participate in the system while pretending to be another participant, possibly violating the system's authentication. For example, an attacker could claim to be a legitimate participant in a UC call. ***T**ampering* threats are based on modifying data on disk, on a network, or in memory, and thereby violating the integrity of the data. In a UC call, an attacker could tamper with the UC chat communication on the network. ***R**epudiation* is claiming that somebody did not do something or is not responsible. For example, a user could claim he did not attend a specific UC meeting. ***I**nformation Disclosure* is the threat of providing information to someone not authorized to see it and violating confidentiality. In a UC call, an attacker could eavesdrop on unencrypted video and audio channels. ***D**enial of Service* threats are based on absorbing resources needed to provide a service to legitimate users and violating availability of the service. For example, an attacker could consume the network resources between UC call participants and UC infrastructure and thereby cause an interruption or unavailability of the service. ***E**levation of privilege* threats allow someone to do something they are not authorized to do. For example, an attacker could gain administrator privileges and modify the configuration of the UC infrastructure.

3.3.2 Privacy Threats. Most privacy properties in the LINDDUN framework [24] are based on terminology proposed by Pfitzmann et al. [88] and are widely recognized in the privacy community. The term ***item of interest*** (**IOI**) denotes subjects, messages, or actions. In the context of UC, IOIs can be the real names of participants or meeting titles containing sensitive information. Pseudonyms are identifiers of a subject other than the subject's real name. LINDDUN covers seven privacy threat categories. ***L****inkability* allows an attacker to infer whether or not two or more IOIs are related within a system. For example, an attacker could link participant names to a specific UC meeting. ***I****dentifiability* violates anonymity and pseudonymity and provides an attacker with sufficient information to identify the subject associated to an IOI (e.g., the sender of a message). Identifiability is a special case of linkability with the subject and its attributes involved. ***N****on-repudiation* guarantees that a user cannot deny (repudiate) that he knows, has done, or has said something. There is a contradiction between repudiation (security threat in STRIDE) and non-repudiation (privacy threat in LINDDUN). The preference for repudiation/plausible deniability or non-repudiation for an architecture depends on the use case. For example, whistleblowers will prefer repudiation so they can deny ever having sent a certain message to protect their safety. On the other hand, a UC meeting could be recorded to prove that a specific conversation happened providing non-repudiation. ***D****etectability* of an IOI means that the attacker can discover whether an item exists or not. For example, a UC video/audio stream encoded via **Real-time Transport Protocol** (**RTP**) is distinguishable from other network traffic based on protocol patterns and traffic volume. *Information* ***D****isclosure* is the same threat as in STRIDE. *Content* ***U****nawareness* means a user is not aware that the system collects or processes user data. For example, a UC meeting participant may incorrectly believe that his real name is not visible to other participants. *Policy/consent* ***N****oncompliance* describes the case where a system does not comply with its own privacy policy. For example, a UC system could record UC meetings without informing the participants, but state in the privacy policy that meetings are not recorded.

3.3.3 DFD/Threat Mapping. Table 1 maps the STRIDE and LINDDUN threats to the elements of our generic UC architecture (see the dataflow diagram in Figure 2). Table cells with a number indicate threats that will be analyzed in detail in the next section. We use the numbers as identifiers for each generic threat. Cells marked with an X and gray background indicate a potential threat that is either irrelevant or does not apply to the corresponding DFD element. Cells marked as *Trusted* represent dataflows that do not cross trust boundaries. We assume that these dataflows are trusted and will not analyze them further.

3.4 Generic Threat Analysis

To analyze the threats to our generic UC architecture, we follow the threat tree patterns from STRIDE [105] and LINDDUN [24]. The item numbers refer to entries in Table 1. When we found significant overlap or dependency of threats and mitigations across threat types or DFD elements, we combined them into a single item. For example, item 3.4.9 covers one STRIDE threat and four LINDDUN threats for all processes in the DFD.

3.4.1 STRIDE - Spoofing - External Entities. The external entities' credentials could be obtained in transit, during change management, or from storage. The use of standard protocols such as TLS **Transport Layer Security** (**TLS**) encryption for the participant's web sessions and SSH (Secure Shell) for the administrator can mitigate these threats in transit [93]. During credential change, strong authentication should be used, and logging and auditing should be in place. Strong authentication could, for example, be two factor or a challenge-response identification technique [20, 56]. File and database access permissions need to be set accordingly for credentials stored on the server. For credentials stored on the client or by a third party, additional authentication factors such as

Table 1. Mapping of STRIDE and LINDDUN Threats to the DFD Elements for a Generic UC Architecture

DFD Item	Security - STRIDE						Privacy - LINDDUN							
	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege	Linkability	Identifiability	Non-repudiation	Detectability	Information Disclosure	Content Unawareness	Policy/consent Noncompliance	
External Entities														
Participant 5.0	1		7				16	19				25		
Organizer 6.0	1		7				16	19				25		
Administrator 7.0	1		7				16	19				25		
Processes														
Web Scheduler Application 1.1	2	4	7	9	12	15	9	9	20	9	9		26	
User Authentication 1.3	2	4	7	9	12	15	9	9	20	9	9		26	
UC Application 2.1	2	4	7	9	12	15	9	9	20	9	9		26	
Scheduler Web Client 3.1	2	4	7	9	12	15	9	9	20	9	9		26	
WebRTC/UC Client 3.3 and 4.1	2	4	7	9	12	15	9	9	20	9	9		26	
Data Stores														
Scheduling details 1.2		5	8	10	13		17	17	21	23	10		26	
User details 1.4		5	8	10	13		17	17	21	23	10		26	
UC configuration 2.2		5	8	10	13		17	17	21	23	10		26	
Log files 2.3		5	8	10	13		17	17	21	23	10		26	
local browser/computer data 3.2 and 4.2		5	8	10	13		17	17	21	23	10		26	
Data Flows														
User Authentication read user details (1.4->1.3)			Trusted						Trusted					
Web Scheduler Application														
Authentication request and response (1.1<->1.3)			Trusted						Trusted					
scheduling details create/read/update/delete (1.1<->1.2)			Trusted						Trusted					
Administrator														
scheduling details create/read/update/delete (7.0 <-> 1.2)	3	6		11	14		18	18	22	24	11		26	
user details create/read/update/delete (7.0 <-> 1.4)	3	6		11	14		18	18	22	24	11		26	
UC configuration create/read/update/delete (7.0 <-> 2.2)	3	6		11	14		18	18	22	24	11		26	
log files create/read/update/delete (7.0 <-> 2.3)	3	6		11	14		18	18	22	24	11		26	
UC Application														
reads scheduling details (1.2->2.1)	3	6		11	14		18	18	22	24	11		26	
reads UC configuration (2.2->2.1)			Trusted						Trusted					
writes log files (2.1->2.3)			Trusted						Trusted					
sends/receives signaling and media from UC client (2.1<->4.1 & 2.1<->3.3)	3	6		11	14		18	18	22	24	11		26	
Web Scheduler Application														
response/requests from web client (1.1<->3.1)	3	6		11	14		18	18	22	24	11		26	
sends invitation to participants (1.1->5.0 & 1.1->6.0)	3	6		11	14		18	18	22	24	11		26	
Participant Y scheduling session with web client (6.0<->3.1)	3	6		11	14		18	18	22	24	11		26	
UC client writes log/cache (3.3->3.2 and 4.1->4.2)			Trusted						Trusted					
Participant uses UC session (5.0<->4.1 and 6.0<->3.3)	3	6		11	14		18	18	22	24	11		26	

the user's IP address, machine fingerprint, or multiple authentication factors could be used. Insufficient authentication threats can occur with null credentials, guest accounts, or anonymous accounts and should be disabled. Predictable credentials should be avoided. For administration of a UC infrastructure, strong authentication and encrypted tunnels for remote access are required. The same can be required for organizers and participants.

3.4.2 STRIDE - Spoofing - Processes. Malware or a manual attack could rename and/or change the link to executable files (such as the UC application) or modify the executable. As a consequence, a manipulated process under the control of the attacker could be started. Signing and verification of executable files can ensure that they are genuine and thereby prevent this kind of attack.

3.4.3 STRIDE - Spoofing - Data Flows. Spoofing the communication between endpoints based on stolen or forged key material can be enabled by weak key generation or via the **Public Key Infrastructure** (**PKI**). To mitigate these scenarios, hardware security modules or secure enclaves, provided by the operating system, could be used to store the key material. Attacks against the PKI or key generation could include legal demands (law enforcement requests), and breaking in.

Spoofing can also be based on weak authentication, **man-in-the-middle** (**MITM**) attacks, and packet injection/modification. This can be mitigated by using machine fingerprints (e.g, MAC &

IP addresses or certificates) in combination with strong authentication and tunneling, e.g., using TLS (mutual TLS authentication between infrastructure machines), SSH, or VPN. Similar to 3.4.1, UC administrator, organizer, and participant communication could be the target of an attack.

3.4.4 STRIDE - Tampering - Processes. If the process input is not appropriately validated, memory modification could result in a denial of service (see also 3.4.12), arbitrary code execution, or SQL injection. In addition, a user or program with write access to memory can tamper processes or attach a debugger. Mitigation strategies should cover input validation or web application firewalls (WAF), separating processes with sandboxing/virtualization, and access controls provided by the operating system, such as shared memory permissions, memory protection, and anti-debugging.

Untrusted code may call trusted code and pass malicious parameters, or called program extensions such as APIs or plug-ins may tamper with memory. Mitigations need to ensure that lower-trusted code cannot execute untrustworthy applications, e.g., via operating system permissions, and that only trustworthy plug-ins are used and trusted processes are called. Input validation on passed parameters is required. In a UC architecture the UC infrastructure/service, scheduling platform, or web browser sessions are potential attack targets.

3.4.5 STRIDE - Tampering - Data Stores. Bypassing protection rules or the protection system are the two main threats for tampering of data stores. If there is no protection or weak protection of the data stores, e.g., on a file system, physical access control and **access control lists** (**ACL**) with appropriate permissions need to be applied. In addition, file and database encryption needs to be enabled.

Another threat category is storage capacity failures where new data cannot be stored (discarded) or the oldest data are deleted or overwritten by new data (wraparound). Additional storage, compression, or moving data to another storage location are mitigation options. Capacity failures could have tampering or denial of service results that are valuable to an attacker. In a UC architecture, access to user/scheduling details, log files, and UC infrastructure configuration can occur.

3.4.6 STRIDE - Tampering - Data Flows. Data flow threats can apply to channels (e.g., e-mail messages that include UC meeting invitations), messages (e.g., **Hypertext Markup Language** (**HTML**) over **Hypertext Transfer Protocol** (**HTTP**) that includes WebRTC), or both. The main attack vectors for both are no or weak integrity, weak key management, MITM, replays (re-sending of a message), reflection (sending a message back to the sender), and collisions (sending a message with the same sequence number as a real message). Mitigations are the same as for dataflow spoofing (see Section 3.4.3).

3.4.7 STRIDE - Repudiation - External Entities, Processes. The first repudiation threat category is account take-over, including accounts of the UC administrator, organizers, and participants. In case the account was compromised, stronger authentication (see Section 3.4.1) could mitigate the situation. If somebody falsely asserts that an account was taken over, strong logging could be used to mitigate. The second threat category is claims that messages were not sent, received, have been altered, or replayed. Mitigations for this threat are digital signatures for signing messages and logging.

3.4.8 STRIDE - Repudiation - Data Stores. Data store transaction repudiation can be mitigated with appropriate logging. Logging should be enabled on transactions, ensuring that sufficient and relevant information is logged. To avoid scattered logs, the logging information needs to be consolidated and all the log entries need a correct time stamp from the involved systems. In case repudiation is a required property for a UC architecture, logged information about scheduled meetings and participants could achieve this requirement (see also 3.4.21).

3.4.9 STRIDE and LINDDUN - Information Disclosure, LINDDUN - Linkability, Identifiability, Detectability - Processes. Side-channel threats are unintentional side effects of computation. The timing of code execution and response times of processes via the network can reveal information about secrets, especially cryptographic key material and other UC infrastructure details. Algorithms should be chosen with constant execution time independent from the secret length, complexity, or protected information (e.g., cryptographic blinding) [60]. Variations of the power consumption of a device (such as microprocessor, encryption device, or smart card) could disclose information about the used cryptographic algorithm and secret keys. Possible mitigation techniques are conditional branches or blinding of cryptographic functions. A hardware redesign depending on the specific type of analysis attack and hardware architecture could be required to optimize the software algorithms.

Protocols can disclose information by the use of banners/headers and their behavior indicating which software and version a specific service is running on or provide details about the UC infrastructure and scheduling platform. These banners should be disabled or changed so as not to disclose this kind of information.

Process logging and virtual memory operations can expose information and can be mitigated by strong logging and use of appropriate system calls.

3.4.10 STRIDE and LINDDUN - Information Disclosure - Data Stores. Information disclosure for data stores can occur by bypassing the protection of an operating system reference monitor through which all access requests pass. Potential targets in a UC architecture are user and meeting scheduling details, log files, architecture configuration, or meeting recordings.

Mitigation options include changing to a stricter and better implemented operating system, encryption, and physical access protection. Metadata of data elements could lead to information disclosure of data stores through names (e.g., "layoff of Bob.txt"), size, or timestamps (creation/last access), and can be mitigated by private directories and permissions.

Physical access to data stores and backups can be protected with physical security and encryption. Non-cleaned storage and reused memory should be manually overwritten and disks destroyed (for decommissioning) rather than reselling the media. Overwriting of spinning media and flash storage does not work reliably and physically destroying them is a secure option.

3.4.11 STRIDE and LINDDUN - Information Disclosure - Data Flows. Information disclosure for dataflows significantly overlaps with dataflow spoofing (see Section 3.4.3) and dataflow tampering (see Section 3.4.6), both in terms of attack vectors and mitigation possibilities.

3.4.12 STRIDE - Denial of Service - Processes. Common DoS threats against processes are realized by consuming resources such as buffers, connections, disk space, or memory. Dynamic resource allocation, quotas, virtualization, and load balancing can help to mitigate these attacks.

Input validation failures can cause processes to crash, for example through buffer overflows, or turn into on elevation of privilege problem (see Section 3.4.15). Mitigation options are input validation within the application itself or a WAF in front of a web application. The WAF can review all incoming traffic and filter out malicious inputs that target security vulnerabilities.

Locks on resources can lead to resource lockout problems if service is denied to other locks. Releasing locks immediately or virtualization/separation of processes can mitigate this attack.

In case of UC web service access, joining a meeting via a provided URL in a web browser (WebRTC), with inappropriate protection such as rate limiting or access verification, the service could be overloaded with requests exhausting the process capacity and impact service availability.

3.4.13 STRIDE - Denial of Service - Data Stores. Denial of service against data stores through squatting (claiming a port, named pipe, etc.) can be limited by appropriate permissions to the

resource object. Access to a data store might be denied by modifying ACLs or taking and holding a resource lock. Appropriate access/lock permissions and moving the data store somewhere an attacker cannot modify permissions and cannot create locks can mitigate this kind of attack.

Exceeding the capacity of data stores by hitting resource quotas, e.g., disk space or bandwidth, can cause a DoS. Adding more disk space or using quotas to limit the DoS to a single application instead of the entire system can help postpone such attacks. Moving processes closer to the system that is accessing the data stores (cloud services or data centers) can improve the bandwidth limit.

Without limits on UC meeting recordings and appropriate retention periods, data store capacity could fill up and cause a DoS for writing logs or storing further recordings.

3.4.14 STRIDE - Denial of Service - Data Flows. DoS against a dataflow could include initiating a connection or action before a legitimated user does (pre-play), incapacitating services by absorbing CPU cycles, memory, network ports (squatting), bandwidth, or disk space. Dropping or limiting (setting thresholds) for slow or multiple sessions, load balancing, virtualization of processes, appropriate permissions to resources, and additional resources could improve the availability.

Corrupt messages for dataflows can occur in the case of no or weak integrity checks (see also Section 3.4.6). Increasing capacity, connection rate limiting, or secure tunneling can mitigate this attack.

Similar to DoS against processes (Section 3.4.12), a high number of connection requests could exhaust the network bandwidth or sockets, which could lead to service unavailability.

3.4.15 STRIDE - Elevation of Privilege - Processes. Dynamic corruption through input validation failures (e.g., stack smashing or heap overflow) and memory access can lead to elevation of privilege and execution of arbitrary code (see also Sections 3.4.4 and 3.4.12, and for static corruption Section 3.4.5). Insufficient authorization through call-chain issues may be used as an attack vector (see Section 3.4.4).

A potential attacker could gain administrative privileges on the UC architecture and access sensitive information such as user data, meeting recordings, and scheduling information.

3.4.16 LINDDUN - Linkability - External Entities. One precondition for the linkability of two pseudonyms is that dataflows or data stores are not fully protected (e.g., not encrypted) (see also Sections 3.4.10 and 3.4.11). The second precondition requires the linkability of **Personal Identifiable Information** (**PII**) based on temporary user ID, IP address, behavioral patterns such as time, frequency, and location, session ID, communication content, or a combination of these factors. In a case where the attacker has access to the identity management data store, which contains personal identifiers of users, different pseudonyms for the same user can be easily linked. Identity management systems with a focus on privacy, such as Idemix, can help to mitigate this threat [17, 44, 49, 119].

Linkability of external entities in a UC architecture can occur, for example, if a UC meeting participant uses a pseudonym during a meeting and the scheduling database stores the pseudonym together with the user's real name or IP address.

3.4.17 LINDDUN - Linkability/Identifiability - Data Stores. Data store linkability and identifiability requires weak access control to persistent data (e.g., databases) leading to information disclosure, and insufficient data anonymization or the possibility of strong data mining in the data store. This implies that the stored information still contains sufficient references to the corresponding data subject, which makes it possible to link different data items within the same database. Redundancy of data or the possibility of linking data between multiple different databases can cause re-identification. To mitigate, the stored data should be minimized and combined with data anonymization techniques such as k-anonymity [103] or differential privacy [26]. In a UC

architecture, linkability at the data store can occur, for example, when the pseudonym of a participant stored in the scheduling database and the real name stored in the user database can be linked via entries in log files.

3.4.18 LINDDUN - Linkability/Identifiability - Data Flows. The two preconditions for the linkability and identifiability of dataflows are that the dataflow is not protected (e.g., no encryption, see also Section 3.4.11) and that the communication is linkable because of the use of an insufficient anonymity system or the lack of such a system. If there is no anonymous communication system, the same preconditions apply as for linkability of entities. The dataflows can be linked to each other by the user's identifiable information such as IP address, session ID, or computer ID (see Section 3.4.16). With inadequate anonymity systems, traffic analysis allows the extraction of information from patterns of traffic, passive attacks (e.g., traffic correlation, fingerprint, or route selection), and active attacks (e.g., sybil attack or traffic watermarking) [100]. To mitigate, anonymity technologies may include Tor and AN.ON [7, 120].

Linkability and identifiability via dataflows are possible if an attacker can link IP addresses, pseudonyms, or real names that are transferred during UC meetings.

3.4.19 LINDDUN - Identifiability - External Entities. Identifiability of entities can occur in four cases. Firstly, if the real identity is used as a login (e.g., firstname.lastname) and at the same time, the dataflow between the user and login system is not sufficiently protected, the user's identity will be exposed. Secondly, if a secret (e.g., PIN or password) is used for login, the relationship between this secret, and the user can be disclosed if either the passwords are not protected in the identity management database and can be connected to the user (e.g., social insurance number, date of birth), or the passwords can be revealed through replay attacks, key loggers, communication eavesdropping, or by observing the user inputting the secret. The third case is a weak implementation of software or physical token for login. The fourth case is when biometrics, used for login, are retrievable and can be linked to an entity, for example, caused by information disclosure of the identity management database or dataflows containing biometrics that are linkable to data stores. The use of secure pseudonymization for user identifiers and privacy-enhancing identity management systems (PE-IMS) [17, 44, 49, 119] in combination with anonymous communication such as Tor [7, 120] can mitigate these threats. Some UC architectures relay calls via a central infrastructure component where all meeting participants are connected. In this case the UC service provider is able to identify participants, for example, based on their logins or IP addresses. Users of such architectures need to trust the service provider to handle this information accordingly.

3.4.20 LINDDUN - Non-repudiation - Processes. The non-repudiation of a process implies that it cannot be denied that the process has been run. This can happen in a case where the process loses its confidentiality and information disclosure attacks at the process are possible (see also Section 3.4.9) or the process uses logs including all actions, which can be traced back to the user. In the context of a UC architecture, logging could prove that a specific meeting has taken place and cannot be denied.

3.4.21 LINDDUN - Non-repudiation - Data Stores. Non-repudiation is when a subject cannot deny certain data in a data store. This data can be stored by the user himself or by somebody else who has stored data about the subject. There are three threat preconditions. Firstly, when little or weak deniable encryption is used to protect the data, it can be proven that data are encrypted or can be decrypted to a valid plain text. Secondly, when there is weak access control to the database, the stored data are no longer deniable. This can occur when there is a threat of information disclosure at the data store (see also Section 3.4.10). Thirdly, if subjects want deniability, but are not able to edit data in the database to cover their tracks, their data becomes non-repudiable. It can be either

impossible to remove or alter the user's own data or impossible to remove or alter someone else's data concerning the user himself. Non-repudiation is a threat also in the context of the GDPR, which grants a right to erasure, or the right to be forgotten (Art. 17) [29]. Non-repudiation with respect to data stores can happen in a UC architecture if it is not possible for a UC user to delete their account from the user database, including historical logged data with PII.

3.4.22 LINDDUN - Non-repudiation - Data Flows. Non-repudiation of a dataflow implies that the subject cannot deny having sent a message. This can occur when data sources of flows are insufficiently obfuscated, when there is weak deniable encryption, weak MACs, or weak off-the-record messaging are used. Possible mitigations include off-the-record messaging [16], deniable authentication, and deniable encryption [79]. A UC related threat is when an attacker can prove that a user sent or received, via the UC architecture network, a specific message such as an IM.

3.4.23 LINDDUN - Detectability - Data Stores. The knowledge that an item of interest exists, without having access to it, can expose sensitive information. For example, a file on a server of a minority group with the full name as the file name of a person can disclose the information that this person is associated with this organization, even when the file content is inaccessible.

Detectability at the data store can occur when there is insufficient access control, because of information disclosure threats (see also Section 3.4.10) and if insufficient information hiding techniques are applied, such that information is revealed due to weak steganography algorithms, which enable steganalysis attacks [3]. In a UC architecture, detectability at the data store can happen when the attacker can prove that a specific user exists on the system, for example, in a case where user names or other PII is stored in clear text in logged data or the user database.

3.4.24 LINDDUN - Detectability - Data Flows. Knowing that a message has been sent can reveal sensitive information, even if the message contents are unknown. For example, if a user calls a civil rights organization from his private computer, the user could be labeled as an activist even without knowing the content of the call. Detectability of dataflows can occur when a weak covert channel uses too much bandwidth from a legitimate channel. Also, by analyzing protocol signatures or the timing of requests, patterns or characteristics of the communication could lead to detection.

Side-channel attacks such as timing information, power consumption, or electromagnetic leaks can result in detectability of dataflows. Steganalysis attacks are possible when weak information hiding techniques are used [3]. Sufficient dummy traffic on the communication network is required to prevent detectability and to ensure the data appear random, except for the sender and recipient(s).

Finally, a weak spread spectrum communication can result in detectability of dataflows. For example, by allowing eavesdropping caused by inadequate establishment of secure communications, insufficient resistance to natural interference, jamming, or fading.

3.4.25 LINDDUN - Content Unawareness - External Entities. Content unawareness can occur when data subjects provide more information than required, or when they are unaware what data the system stores about them. For example, a UC provider could analyze a user's video stream to infer information from the user's face or background. Organizations commonly use privacy policies to inform users about data collection; however, privacy policies have been shown to be long and difficult to understand, which makes them ineffective [2, 69]. The *Platform for Privacy Preferences Project* [70] was an approach to make privacy policies machine-readable and thereby allow automated matching against user preferences. However, P3P was discontinued in 2018 because of its complexity and low adoption rate by web browser vendors. Recently, advances in machine learning have made automated analysis of privacy policies possible [45]. Personal information feedback

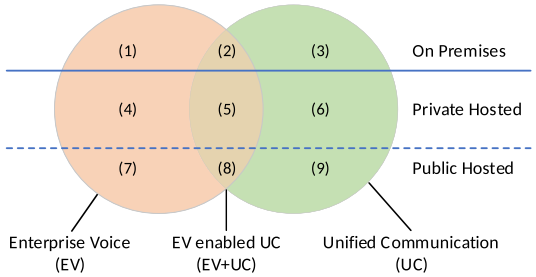

Fig. 3. Classification of UC market segments [46].

tools have also been suggested [66, 83] to help users gain privacy awareness and self-determine which personal data to disclose.

3.4.26 LINDDUN - Policy/consent Noncompliance - Processes, Data Stores, Data Flows. Policy noncompliance, based on legal requirements such as the GDPR, can occur when the internal system behavior does not correspond with the privacy policy provided to the user. This can happen if an attacker tampers with the internal policies or when the policy rules are incorrectly managed by the system administrator. A non-technical approach could be to train administrators and employees of the UC architecture and penalize disclosure of user information in combination with audits.

3.5 Mitigation Controls, Strategies, and Techniques

Table 2 summarizes how the mitigation controls described above, map to the threats and DFD elements from Table 1. The table shows that mitigation controls for security threats on the left-hand side are more represented than mitigations for privacy threats on the right-hand side, except for areas of overlap such as encryption. The most important mitigations by occurrence are authentication, access control, and encryption, i.e., classic techniques from computer security. One reason for this may be a higher interest in security, because it often protects assets with a monetary value compared to privacy which protects personal data. This points to a need for more research on privacy protections, awareness-raising, and adoption of privacy protections in deployed architectures.

For four threats, very limited technical mitigations are available; spoofing of processes (Section 3.4.2), linkability of dataflows (Section 3.4.18), content unawareness (Section 3.4.25), and policy non-compliance (Section 3.4.26). Although some of these threats may be addressed with non-technical mitigations, such as audits and user training, the lack of technical mitigations points to a need for further research. A first step toward a technical solution for policy non-compliance is the automated analysis of flow-to-policy consistency, i.e., the comparison of observed dataflows with policy statements [4].

4 THE UNIFIED COMMUNICATION LANDSCAPE

The UC landscape can be categorized by features, deployment type, availability, and licensing. The three feature classes are **Enterprise Voice** (**EV**), EV-enabled Unified Communication, and Unified Communication. EV provides IP telephony experience on a UC capable platform, but only desk phones or softphones with a headset are deployed (segments 1, 4, and 7 in Figure 3). UC providing a UC-only experience is mostly deployed as a UC software client with no PBX calling features and no access to the **public switched telephone network** (**PSTN**) (segments 3, 6, and 9). The combination of EV+UC describes telephony-enabled UC solutions deployed as UC software clients, with the optional deployment of desk phones (segments 2, 5, and 8).

Table 2. Mapping of Mitigations to Threats and DFD Elements

Mitigation Controls	Threats	3.4.1 Spoofing, external entities	3.4.2 Spoofing, processes	3.4.3 Spoofing, data flows	3.4.4 Tampering, processes	3.4.5 Tampering, data stores	3.4.6 Tampering, data flows	3.4.7 Repudiation, entities/processes	3.4.8 Repudiation, data stores	3.4.9 Information disclosure/Linkability/ Identifiability/Detectability, processes	3.4.10 Information disclosure, data stores	3.4.11 Information disclosure, data flows	3.4.12 Denial of service, processes	3.4.13 Denial of service, data stores	3.4.14 Denial of service, data flows	3.4.15 Elevation of privilege, processes	3.4.16 Linkability, external entities	3.4.17 Linkability/Identifiability, data stores	3.4.18 Linkability/Identifiability, data flows	3.4.19 Identifiability, external entities	3.4.20 Non-repudiation, processes	3.4.21 Non-repudiation, data stores	3.4.22 Non-repudiation, data flows	3.4.23 Detectability, data stores	3.4.24 Detectability, data flows	3.4.25 Content unawareness, ext. entities	3.4.26 Policy/consent non-compliance
Authentication																											
- Strong authentication		x		x			x	x				x															
Access permissions																											
- File system, database, memory, object permissions (ACLs)					x	x				x		x		x		x	x	x			x	x		x			
Encryption																											
- Tunnelling with cryptography and integrity checks (TLS, SSH, etc.)		x		x			x					x			x												
- File, data store, and database encryption						x					x						x					x		x			
- Cryptographic algorithms optimized for the used hardware (timing, power consumption)										x															x		
Secure infrastructure deployment																											
- Signing/verification of executable files			x																								
- Sandboxing/virtualization					x					x			x		x	x					x						
- Physical access/security						x				x		x									x	x		x			
- Input validation/sanitisation/WAF					x								x			x											
- Process call chain (preventing execution of untrusted code)					x											x											
- Decommission physical media											x																
DDoS prevention																											
- Load balancing (network)													x		x												
- Storage and memory capacity mitigations (discard, log-rotation, compress, move)						x							x	x													
- Optimize bandwidth based on location of architecture components														x													
- Limitation of concurrent connections with timeouts and thresholds															x												
Repudiation/(Plausible Deniability)																											
- Strong logging (appropriate information, consolidated, and time synchronized)								x	x	x												x					
- Privacy preserving authentication (off-the-record messaging, deniable authentication)																							x	x			
- Deniable encryption																								x			
- Digital signatures for signing								x																			
Users can securely delete data																											
- Personal information feedback tools																										x	
Secured meeting invitations																											
- Remove protocol specific banners/header, behavioral information										x																	
Anonymous communication																											
- Anonymous Communication (Tor, AN.ON)										x							x		x	x					x		
Undetectability																											
- Steganography, dummy traffic																									x	x	
- Privacy enhancing identity management systems (PE-IMS)																	x				x						
- Pseudonymization																					x						
- Persistent-data anonymization																		x									

Table 3. Overview of the Key Properties and Functionality of 10 UC Platforms

Architecture		Signal Messenger	WhatsApp	Wire	Jitsi	Zoom	Microsoft Skype	Microsoft Teams	Slack	Webex Teams	Google Meet
Owner		Signal Messenger LLC	Facebook	Wire Swiss	jitsi.org	Zoom Video Communication	Microsoft	Microsoft	Slack Technologies	Cisco Systems	Google
Organisation jurisdiction		USA	USA	Switzerland	Open Source	USA	Luxembourg/ USA	USA	USA	USA	USA
Estimated users		not disclosed	2 billion (4)	not disclosed	not disclosed	200 million (5)	300 million (4)	390 million (4)	300 million (4)	not disclosed	100 million (5)
UC market segment		UC 3,6,9	UC 9	UC 3,6,9	UC 3,6,9	EV+UC 5,8	EV+UC 8	EV+UC 8	EV+UC 8	EV+UC 8	EV+UC 8
Deployment type		On-premises Private Cloud Public Cloud	Commercial Public Cloud	On-premises Private Cloud Public Cloud	On-premises Private Cloud Public Cloud	Public Cloud Hybrid Cloud	Commercial Public Cloud	Commercial Public Cloud	Commercial Public Cloud	Commercial Public Cloud	Commercial Public Cloud
Architecture type		Signal/ open source	Signal/ closed source	WebRTC (Signal based), open source	WebRTC/ open source	proprietary/ closed source	proprietary/ closed source	proprietary/ closed source	proprietary/ closed source	proprietary/ closed source	proprietary/ closed source
Licensing		free	free	paid service	free	freemium/ paid service	freemium/ paid service	paid service part of Office365	freemium/ paid service	freemium/ paid service	freemium/ paid service
UC Features (required)	Audio 1:1	yes	yes (3)	yes	yes	yes	yes	yes	yes	yes	yes
	Video 1:1	yes	yes (3)	yes	yes	yes	yes	yes	yes (1)	yes	yes
	Instant Messaging 1:1	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
	Instant Messaging 1:n	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
UC Features (optional)	Audio 1:n	no	yes (3)	yes	yes	yes	yes	yes	yes	yes	yes
	Video 1:n	no	yes (3)	yes	yes	yes	yes	yes	yes (1)	yes	yes
	Content sharing	no	no	yes	yes	yes	yes	yes	yes (2)	yes	yes
	VMR/Scheduling for meetings	no	no	no	yes	yes	no	yes	no	yes	yes

1:1 = a conversation between two participants
1:n = a conversation between multiple participants

(1) only on Windows, macOS, Linux client
(2) only on macOS and Windows client
(3) only on mobile app

(4) estimated monthly active users
(5) estimated daily meeting participants

The second categorization is the deployment type of the architecture. Public Hosted (UCaaS) describes UC solutions deployed in a provider's data center on a multi-tenant platform (segments 7, 8, and 9 in Figure 3). Private Hosted UC solutions are deployed in a provider's data center, with the service dedicated to the enterprise (segments 4, 5, and 6). On-Premises describes UC solutions deployed in the enterprise's data center (segments 1, 2, and 3). The type of deployment is important because it influences which mitigations are available for security threats of the category secure infrastructure deployment and **Distributed Denial of Service** (**DDoS**) prevention (see Table 2).

The third categorization is the availability and the type of licensing; commercial closed source license support model, open source, open source with optional commercial support, and end consumer freemium. Freemium models provide the base functionality for free, whereas premium features are payable, and free users need to accept collection of user data for tracking or advertising.

Table 3 shows the feature type, deployment type, licensing type, and available UC features of the ten architectures, we will evaluate in detail. We focus on UC architectures from market segments EV+UC (segments 2, 5, and 8 in Figure 3) and UC (segments 3, 6, and 9). We exclude enterprise voice-only products and solutions, including handsets. In addition, we exclude systems that require specific video conferencing hardware or telephone handsets. We evaluate only UC architectures that provide audio and video collaboration with two or more participants as well as live chat. Because our focus is on security and privacy, we have included platforms that provide interesting security and privacy properties (Signal, Wire, and Jitsi) as well as a selection of mainstream UC applications to show which security and privacy features are commonly implemented.

5 SECURITY AND PRIVACY OF UNIFIED COMMUNICATION PLATFORMS

We will now evaluate the security and privacy features of ten major UC platforms, based on a review of academic literature and technical documentation. The amount and detail of information available regarding security and privacy controls on the various UC platforms is highly variable. We have included detailed information whenever available, not just to describe the specific tool,

Table 4. Overview of Mitigation Controls Available in 10 UC Platforms

Threats Mitigation Controls	Signal Messenger	WhatsApp	Wire	Jitsi	Zoom	Microsoft Skype	Microsoft Teams	Slack	Webex Teams	Google Meet
Security Focused										
Authentication										
Username/password	**S**	**S**	**S**	**N/A**	**S**	**S**	**S**	**S**	**S**	**S**
PIN/code	**S**	**S**	**N/A**	**S**	**S**	**S**	**S**	**N/A**	**S**	**S**
SMS	**S**	**S**	**S**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**S**
Phone number/IMEI	**S**	**S**	**S**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**
Multi-factor	**S**	**S**	**P** SAML IdP	**N/A**	**N**	**S**	**S**	**S**	**P** SAML IdP	**S**
Privacy preserving authentication	**N** phone number	**N** phone number	**N** phone number e-mail	**N/A**	**N** Username/ password	**N** Username/ password	**N** Username/ password	**N** Username/ password	**N** Username/ password	**N** Google Account required
Access permissions	**N/A**	**N/A**	**S** Role Based	**N/A**	**S** Role Based	**N/A**	**S** Role Based	**S** Role Based	**S** Role Based	**S** Role Based
Encryption										
Data in transit	**S** Signal Curve25519 AES-256-CBC HMAC-SHA256	**S** Signal Curve25519 AES-256-CBC HMAC-SHA256	**S** Proteus TLS 1.2 ChaCha20 Curve25519 HMAC-SHA256 HKDF	**S** WebRTC DHE-RSA&DAS ECDHE-RSA&ECDSA DTLS-SRTP AES-128 GCM SHA256	**P** TLS 1.2 SRTP AES-256-ECB **S** Zoom Version 5.0 TLS 1.2 SRTP AES-256-GCM	**P** Proprietary RSA-1536 & 2048 AES-256 SHA-1	**S** HTTPS based REST signaling TLS 1.2 AES-256 2048-bit SHA256RSA certificate SRTP	**S** HTTPS based REST signaling TLS 1.2 AES-256 SHA2	**S** Signaling: TLS 1.2 HTTPS/WSS ECDHE RSA AES-256 GCM SHA384 Media: SRTP AES CM128 HMAC_SHA1_80	**S** WebRTC TLS 1.3 RSA 2048 Curve25519 AES-128-GCM SHA384
Data at rest	**P** Stored on client	**U** backup	**S** depends on client device	**N/A**	**U**	**P** Stored on client	**S** Encrypted in SharePoint	**S** FIPS 140-2 compliant AES256	**S** same as for data in transit	S Encrypted on GDrive AES
End-to-end	**S**	**S**	**S**	**P** depends on browser	**P**	**P**	**N**	**N**	**P** not for media SRTP	**N**
Secure infrastructure deployment	**U:** Public Cloud **P:** Private Cloud	**U** Public Cloud	**U:** Public Cloud **P:** Private Cloud	**U:** Public Cloud **P:** Private Cloud	**U:** Public Cloud **P:** Private Cloud	**U** Public Cloud	**U** Public Cloud	**U** Public Cloud	**U:** Public Cloud **P:** Private Cloud	**P** Public Cloud
DDoS prevention	**U:** Public Cloud **P:** Private Cloud	**U** Public Cloud	**U:** Public Cloud **P:** Private Cloud	**U:** Public Cloud **P:** Private Cloud	**U:** Public Cloud **P:** Private Cloud	**U** Public Cloud	**U** Public Cloud	**U** Public Cloud	**U:** Public Cloud **P:** Private Cloud	**S** Public Cloud
Privacy Focused										
Repudiation/ (Plausible Deniability)	**S** Signal/OTR	**S** Signal/OTR	**S** OTR	**U**	**U**	**U**	**U**	**U**	**U**	**U**
Users can securely delete data (client)	**S**	**S**	**S**	**N/A**	**P** manual	**S**	**P**	**P** manual	**S**	**S** automatically
Users can securely delete data (server)	**S** automatically	**S** automatically	**S** automatically	**S** automatically	**P** manual	**U**	**P**	**P** manual	**U**	**S** automatically
Secured meeting invitations	**N/A**	**N/A**	**P** hyperlink	**P** Meeting name	**P** email, IM, SMS	**N/A**	**P** email	**N/A**	**P** email	**P** Meeting Code
Anonymous communication	**P** Relay Calls	**N** client end-to-end	**N** full mesh 1:1 encrypted calls	**N** server or client end-to-end	**N**	**P** Relay Nodes	**N**	**N**	**N**	**N**
Undetectability	**N**	**N** client end-to-end	**N**	**N** server or client end-to-end	**N**	**N**	**N**	**N**	**N**	**N**

Legend: **S** = Supported **N** = Not-supported **U** = Unknown **N/A** = not applicable **P** = Partially supported/configurable = concern area

but also to illustrate how a control of that type might work for other tools. However, some details can become outdated quickly due to the fast-paced evolution of these platforms. Table 3 gives an overview of the functionality of the 10 platforms, and Table 4 shows their security and privacy features, structured following the generic analysis in Sections 3.4 and 3.5. A gray background indicates an area of concern, which is discussed in Section 6. The detailed discussion of three of the ten platforms is included in the *online supplementary material*.

5.1 Signal Messenger—Signal Messenger LLC and Signal Foundation

The Signal Messenger client, owned by Signal Messenger LLC and the Signal Foundation, is the successor of RedPhone and TextSecure, which were first launched by Whisper Systems in 2010. The services were stopped shortly after Twitter acquired Whisper Systems in 2011, and resumed when Twitter released TextSecure and RedPhone in 2011 and 2012 under the **GNU General Public License** (**GPL**). In 2013, Moxie Marlinspike left Twitter and founded Open Whisper Systems to continue RedPhone and TextSecure. In 2014 and 2015, Open Whisper Systems merged the two programs into the Signal Messenger and made them available under the GPL and **Affero General Public License** (**AGPL**) for several platforms, including Android, iOS, Microsoft Windows, macOS, and Linux [75].

5.1.1 Authentication and Access Permissions. Signal Messenger requires users to register with a telephone number. This raises some privacy concerns because the end-user has to expose their

phone number to Signal. As a measure to prevent access to user data, Registration Lock adds an additional PIN verification before users can register a new device to their phone number. On Android and iOS, Screen Lock can be enabled to unlock the app with a PIN, passphrase, or biometrics [107].

5.1.2 Encryption. The client provides end-to-end encrypted communication channels for voice and video calling, IM, and file transfer based on the Signal Protocol. Signal updates the session key with every new message in a process called *ratcheting* [117], which provides **perfect forward secrecy** (**PFS**) and protects against MITM attacks. Signal also provides *deniable communication* with the support of **Off-The-Record Messaging** (**OTR**), ensuring that whenever Alice sends Bob a specific message, Bob cannot prove to Mallory (a 3rd party) that Alice sent him the message [50, 74].

5.1.3 Secure Infrastructure Deployment. The central servers are operated by Signal Messenger LLC and the Signal Foundation [108]. A significant difference to other UC platforms and especially other platforms based on the Signal Protocol, such as WhatsApp, is that the owner is a not-for-profit foundation with open-source in mind. Currently there are no indications that Signal Messenger will be monetized, for example, with advertising.

5.1.4 Privacy. In contrast to WhatsApp, Signal Messenger does not send the contact list to the server in clear text (see Section 5.2). Instead, it hashes the phone numbers before sending them to the server. The server responds with the common contacts and discards the query. There is no cloud backup functionality, which could expose the messages. Signal supports disappearing messages, which are removed on the sender's and receiver's devices a specified length of time after they have been seen. On Android, local backups can be enabled by the user [68]. The *Censorship Circumvention* feature allows the bypassing of censorship in countries such as Egypt, Oman, Qatar, and UAE with the help of Domain Fronting. This functionality is based on the country code of the phone number used for registration. However, Google and Amazon stopped this technology by changing their terms of service [76]. Signal is working to provide an alternative for Domain Fronting. Alternatively, censorship can be bypassed by routing the client traffic through the Signal infrastructure. No communication metadata is stored on the central servers. The source code for the client and server components is public and has been formally peer-reviewed [21, 106]. The Signal Messenger received recommendations and positive feedback from prominent security and privacy experts [18, 104].

5.2 WhatsApp—Facebook

WhatsApp Messenger, owned by Facebook since 2014, is a popular free software with 2 billion users as of 2021 [114]. The client is available on many platforms with a focus on mobile devices. The services, based on the Signal Protocol, include IM, audio and video calls, and media sharing capabilities such as sharing pictures with individual users or groups.

5.2.1 Authentication and Access Permissions. During the installation of the WhatsApp client, the client requests a PIN from the server, via SMS, to the phone number of the smartphone. After the PIN has been received, the client requests a unique key from the server, which is stored locally and used for further authentication [22]. The user's phone number is used as a user ID. Two-factor authentication can be enabled by the user to add a PIN for client authentication. To log in from a browser or a tablet, a QR code is displayed and needs to be scanned by the registered smartphone to authenticate the additional device. The WhatsApp client uploads locally stored phone numbers to the WhatsApp/Facebook infrastructure without notifying the user, to find WhatsApp users among the client's contacts. Users cannot select specific phone numbers to upload. Point to

point (1:1) and multipoint (1:n) audio and video calls are supported with up to three participants. All calls are ad hoc, i.e., there is no call scheduling or VMR functionality. To verify what is stored by WhatsApp, the user can request the relevant GDPR information from Facebook. During the registration process, the SMS confirmation, as out of band authentication, can be spoofed or retrieved by social engineering and is prone to identity fraud. The same attack vector applies to the QR code association to bind other devices to a registered smartphone. The requirement to use a phone number to register raises privacy concerns because it exposes the end-user's phone number. A security issue is that a WhatsApp account may be taken over by taking over the phone number. There have been incidental reports of this happening through "retired" and reused phone numbers or attackers taking over the mobile account by talking to the mobile carrier while impersonating the user, either by phone or in a store, and asking to port a phone number to a new service or device ("SIM swapping") [140].

5.2.2 Encryption. In April 2016, Facebook announced end-to-end encryption to provide authentication and confidentiality for all communication, based on the *Signal Protocol* in partnership with Open Whisper Systems (see see Section 5.1) [91, 131]. For audio and video call setup between participants, the initiator transmits the session setup and **Secure Real-time Transport Protocol** (**SRTP**) master secret via the Signal Protocol [10, 131]. Messages are encrypted and deleted from the WhatsApp server as soon they are retrieved by the recipient's client. With the Signal Protocol, the security and privacy risks in transit have been reduced significantly. OTR addresses security and privacy properties such as repudiation and plausible deniability, but the metadata, including mobile phone numbers, groups, timestamps on messages, device types, profile pictures, and contacts are visible and stored by Facebook. The storage of keys, contacts, messages, and media on mobile devices and computers introduces attack vectors that depend on the security and privacy capabilities of the client's operating system. The functionality to backup to a cloud drive, e.g., Google Drive, is another risk, because Facebook provides no further information about possible encryption of backups. Forensic analysis of artifacts left by WhatsApp on Android allows reconstructing of the list of contacts and the chronology of the messages that have been exchanged by users [6]. This is a potential security and privacy threat on the linkability between individuals and conversations by correlating these artifacts.

5.2.3 Secure Infrastructure Deployment. Facebook's closed source and public cloud architecture approach uses a modified **Extensible Messaging and Presence Protocol** (**XMPP**) server running on FreeBSD and Erlang as the core infrastructure deployed globally on hundreds of nodes processing billions of messages per day [22].

5.2.4 Privacy. The client uploads contact information, without knowledge of the user, from the user's device to the WhatsApp infrastructure, which allows reconstructing of the social network relations between users, groups, and sent messages. This information could be sold (data monetization), used for targeted ads, or revealed to government institutions. There are plans by Facebook to use the WhatsApp metadata within Facebook, which gives them the power to create full user profiles with data aggregated from different sources [91]. The EU Commission has fined Facebook €110 million for providing false information during the 2014 acquisition of WhatsApp wherein Facebook claimed that "it would be unable to establish reliable automated matching between Facebook users' accounts and WhatsApp users' accounts". However, Facebook started automatic matching of user identities in 2014 [27]. In August 2016, Facebook changed the terms of service so that WhatsApp users now agree that their data can be shared with other Facebook companies. The only way to avoid that is to delete the WhatsApp account [133]. In January 2021, WhatsApp changed their Privacy Policy for users outside the European Union to allow sharing of

information with other Facebook companies. Users not accepting the new terms are denied the service [132].

WhatsApp is considered standard consumer software whose security properties were enhanced well after it gained mass adoption, which means that its presence on a mobile phone is not likely to arouse suspicion of "subversive" activity. When the user re-installs WhatsApp on a new phone, messages that are waiting to be delivered are automatically re-encrypted and resent by the sender. However, the sender cannot verify whether the new recipient is the originally intended recipient. This behavior of WhatsApp may be an acceptable trade-off if the priority is message delivery [122]. Governments and law enforcement agencies across the world regularly express their unease at the strong confidentiality that WhatsApp provides through its end-to-end encryption, typically presenting the use for organized crime and child sexual abuse as problematic situations. Countries such as China, North Korea, Iran, Syria, Qatar, and United Arab Emirates have banned WhatsApp completely, with additional potential motivations being the protection of local telecommunication companies (economical) and the suppression of dissent (political) [143].

5.3 Wire—Wire Group Holdings GmbH/Wire Swiss

Wire was founded in 2012 by former Skype and Microsoft employees with headquarters in Berlin and offices in Switzerland, Sweden, and the United States. The Wire client is available on Android, iOS, Windows, macOS, Linux, and several web browsers. It features messaging, voice/video calling, file-sharing, and search, all protected by end-to-end encryption. Wire hosts over 1300 enterprise customers and claims to be the fastest growing collaboration platform in the world [135]. Detailed information about Wire's operation, security, and privacy features is available [138, 139].

5.3.1 Authentication and Access Permissions. Registration consists of a mandatory user registration and optional client and push token registrations. Registration by e-mail and by phone are the two basic registration flows, with the verified phone number or e-mail serving as the user identifier. In both cases, the Wire server sends a random verification code to the client, either via e-mail or text message (SMS), and allows three attempts to respond with the correct verification code before a new verification code needs to be requested. Upon successful registration, the client receives a Wire internal ID and authentication cookie. Users have to provide a non-unique profile name, and the user profile additionally stores the profile picture, accent colors, language, cookie label, timestamp of registration, IP geographic location, and web application settings. After user registration, the conversation contents are synchronized across all of the user's devices. Passwords are passed into the *scrypt* key derivation function with a random salt [84, 85]. The resulting hashes are stored along with the salt and parameters on the Wire server. On the client, passwords are kept only in volatile memory.

Client registration is limited to eight client applications per user to limit the computations needed when sending encrypted messages. Wire's protocol overlaps significantly with Signal's protocols, for example, using, Double Ratchet for key management [34, 73, 74, 86]. Prekeys are generated by the client based on Curve25519 and stored locally to initiate cryptographic sessions. During registration, the prekeys are bundled with the client's public key and uploaded to the Wire server. These key bundles can be used by other clients to asynchronously initiate end-to-end encrypted communication even if the recipient is offline. The server removes used prekeys immediately and clients regularly need to upload fresh prekeys. During client registration, Wire collects the device class (mobile, tablet, and desktop), model (e.g., iPhone 8), a label to distinguish clients, cookie, password, timestamp, and IP geographic location. As soon as a new client application is linked to an account, all other clients of the same account are notified via e-mail.

Push token registration allows clients to receive notifications over Google **Firebase Cloud Messaging** (**FCM**) or **Apple Push Notification** (**APN**) in case the device is offline (no data connection).

Wire's authentication protocol uses a combination of short-lived (15 minute) access tokens and long-lived user tokens. Authentication requests via the API against Wire infrastructure resources are protected by access tokens and user tokens are used to continuously obtain new access tokens. All tokens are strings signed with Ed25519 by the server including the **Universally Unique Identifier** (**UUID**) and expiration time sent as HTTP cookies. The user tokens and cookies can be persistent or session-based and are selected during login by the client. The cookies follow the same semantics as those specified by the HTTP protocol. Wire supports two different login types; password login and SMS login. As soon as users add a password to their account or have verified a phone number they are able to login. During login the client chooses the type of login, session, or persistent, with a corresponding cookie placed after successful login. For password login, the client provides an e-mail address or phone number together with the password transmitted to the server via TLS. The server verifies the provided login name and password using scrypt and issues a new user and access token as an HTTP cookie. Users registered with a verified phone number can login via SMS following the same procedure as during registration.

5.3.2 Encryption. Text messages and assets (larger binary entities such as pictures) are end-to-end encrypted between two clients with Proteus, Wire's main cryptographic protocol [136]. Proteus is suitable for use in asynchronous environments through its use of prekeys as described above. Two parties do not need to be online at the same time to initiate an encrypted conversation. The cryptographic primitives used in Proteus are ChaCha20 (cipher), HMAC-SHA256, and Diffie–Hellman Curve25519 (message authentication codes), and HKDF (HMAC key derivation function) [13, 14, 62, 63]. Users can call each other in 1:1 or group conversations. Group conversations are established via a full mesh of end-to-end encrypted 1:1 calls between all pairs of participants. The call signaling is encrypted with Proteus and uses **Session Description Protocol** (**SDP**) to negotiate the client call capabilities [43]. Endpoints establish media flows directly when possible, with the help of ICE and **Traversal Using Relays around NAT** (**TURN**) servers to identify the most suitable transport path [71, 101]. The TURN servers are part of the Wire infrastructure and generic credentials are used for authentication. TURN servers cannot associate user identifiers with a specific call record.

Wire clients use HTTPS with Strict Transport Security, TLS 1.2, with ciphers PFS and certificate pinning. SRTP is used for call media exchange, and negotiation of encryption algorithm, keys, client authenticity verification, and parameters is achieved via the **Datagram Transport Layer Security** (**DTLS**) handshake [10, 77, 97]. For group calls, each call leg of the full mesh is individually encrypted and the keys are not shared among participants. On iOS, local data are protected via sandboxing and not synced to iCloud or iTunes backup. On Android, access permissions protect local data and cached data, and external storage is encrypted using **Advanced Encryption Standard** (**AES**) 128. On desktop clients, Wire recommends the use of full disk encryption such as FileVault on macOS or Bitlocker on Windows.

5.3.3 Secure Infrastructure Deployment. The server infrastructure is operated by Wire, but an on-premises/private cloud deployment is possible on request.

5.3.4 Privacy. The source code for the infrastructure and clients was released in 2016 and is available at https://github.com/wireapp under GNU AFFERO general public license V3, with the limitation that the method in which Wire clients interact with the Wire infrastructure should not be changed. In 2017, Wire sponsored a security audit for the web application, Android/iOS client,

and protocol implementation [137]. The audit identified no critical issues in the core cryptographic protocol Proteus and its Cryptobox API. Some low-, medium-, and high-severity bugs were identified in the iOS, Android, and Web client, which have since been addressed by Wire. For example, the web application used an outdated JavaScript framework vulnerable to injection attacks. Separate research from the University of Waterloo found that unhashed and unencrypted passwords were sent from the web client to the Wire infrastructure via TLS [125]. As a result, server operators could have access to cleartext passwords during authentication, and the recommendation is to use a technology such as password-authenticated key agreement.

5.4 Jitsi-Jitsi.org

Jitsi is a collection of free and open-source multi-platform voice, video conferencing, and IM applications under the Apache license 2.0 based on the OSGi framework. On desktop systems such as Microsoft Windows or Linux, a web browser supporting WebRTC (e.g., Google Chrome or Firefox) can be used as a feature-rich UC client without installing additional plug-ins or software. There are dedicated applications and **Software Development Kits** (**SDKs**) available for Android and iOS.

Jitsi was formed in 2003 as a student project called *SIP Communicator*. With the advancement of WebRTC, the Jitsi Video Bridge was added to allow web-based multi-party video calls based on WebRTC technology. Jitsi operates https://meet.jit.si, a version of Jitsi Meet, hosted for free community use. Other projects include the SIP gateway Jigasi, the library lib-jitsi-meet which allows to securely capture, playback, and stream audio and video flows, and the Chrome extension Jidesha for calendar integration and screen sharing. Jitsi is available at https://github.com/jitsi/jitsi. The project is driven by the open-source community, fully funded by 8 × 8 Inc. (https://www.8x8.com), a commercial cloud-based UC provider that offers a Jitsi-based cloud video conferencing service.

5.4.1 Authentication and Access Permissions. The meeting rooms are ephemeral, generated ad hoc, and can be protected with a password or PIN. Meeting rooms are created when the first participant joins and destroyed when the last participant leaves. No history or messages are kept once a meeting room is destroyed. If somebody joins a meeting with a name that was in use before, a new meeting room is generated with the same name, without any connection to previous meetings. This reduces the potential attack vector of pre-created meetings which would make them easier to identify and target. The meeting room name needs to be shared carefully, for instance it is wise not to share it on social media, and PINs are recommended [53].

5.4.2 Encryption. The default settings for the jitsi-meet server support TLS with cipher suites based on Diffie–Hellman key exchange (DHE-RSA, DHE-DSA) and elliptic curve Diffie-e (ECDHE-RSA, ECDHE-ECDSA) [109]. For WebRTC traffic, **Transmission Control Protocol** (**TCP**) port 443 and **User Datagram Protocol** (**UDP**) port 10,000 must be allowed from the client to the Jitsi server. The WebRTC framework itself does not provide end-to-end encryption between the clients, but Jitsi includes some encryption features on top of WebRTC, depending on the way in which Jitsi operates, P2P or **Jitsi Videobridge** (**JVB**) [95]. The P2P mode is used for one-to-one meetings. In this case, audio and video are encrypted using DTLS-SRTP (Secure Real-time Transport Protocol via DTLS) all the way from the sender to the receiver, even if they traverse network components such as TURN servers [33]. JVB mode is used for more than two participants (1:n) in a meeting. In this case, all audio and video traffic are encrypted on the network using DTLS-SRTP, but packets are decrypted while traversing the JVB. The Videobridge never stores packets in persistent storage and processes them in memory, while they are being routed to other participants in the meeting [52]. Starting with Google Chromium 83 (May 2020), the insertable stream API feature in Chrome,

Edge, Opera, and Brave allows Jitsi Meet to manipulate encoded packets before sending them on the network, and as a result the Jitsi community started new efforts for end-to-end encryption support [51].

5.4.3 Secure Infrastructure Deployment. The Jitsi community team currently operates https://meet.jit.si servers in six regions across the globe. It is possible to install Jitsi on premises or operate it in a private cloud. There is no commercial support available for the Jitsi project, but help is provided by the community via a forum and mailing lists. On the same community platform, individuals and companies can provide or ask for support services. This could be an issue for organizations operating a private instance of Jitsi with the need to receive external support in a timely manner. An alternative for operating a private instance of Jitsi is to use a managed Jitsi service offered by a third party, such as https://jitsi-hosting.eu/ or use the 8 × 8 Inc. UC cloud service.

5.4.4 Privacy. The open source licensing and public availability of Jitsi allows users to host their own Jitsi instance with full control of the infrastructure. Hosted commercial Jitsi services and the community driven https://meet.jit.si are other feasible approaches to using Jitsi. End-to-end encryption is provided with the option to install a Jitsi client on mobile devices or only use a supported web browser.

In the case of https://meet.jit.si all users are moderators and can, for example, mute other participants. For stricter moderator permissions, a private Jitsi Meet instance needs to be deployed. The public https://meet.jit.si service collects some metadata and statistics to improve the service, including an anonymous identifier, bitrate, available bandwidth, SDP offers and answers, utilization, and mobile app crash dumps. In cases where this data collection is not acceptable, a private Jitsi instance can be deployed and operated.

5.5 Zoom-Zoom Video Communications

Zoom, founded by the former Cisco Webex engineer Eric Yuan in 2011, is recognized for its user friendly applications, which are available as closed-source for Windows, Linux, macOS, iOS, and Android. Various plug-ins allow integration, for example, with Microsoft Outlook, IBM Notes, Firefox, Google Chrome, and Skype for Business. Zoom provides free video conferences for a limited number of participants and a limited duration. Larger or longer conferences require a paid subscription. In December 2019, the maximum number of daily meeting participants on Zoom was approximately 10 million. In March 2020, Zoom reached more than 200 million daily meeting participants, which positions Zoom among the top UC service providers [141].

5.5.1 Authentication and Access Permissions. To start a meeting, the meeting host has to authenticate via HTTPS to the Zoom infrastructure using their user credentials (ID and password). A unique per-client, per-session token is used to identify each participant attempting to join a meeting. For each meeting, a unique set of session parameters are generated by Zoom. Authenticated participants must have access to these session parameters in conjunction with the unique session token, to successfully join a meeting. In addition, Zoom supports **Security Assertion Markup Language** (**SAML**) and OAuth as methods for **single sign-on** (**SSO**) [146].

5.5.2 Encryption. All data transmitted from the Zoom client to the Zoom cloud is encrypted in transit with AES via TLS 1.2. The preferred method of communication is HTTPS, leveraging TLS 1.2 encryption and PKI Certificates issued by a commercial certificate authority. Signing into the client, scheduling a meeting, chatting, polling, sharing files, and in-meeting Q&A all use TLS 1.2. Before Zoom client version 5.0 (released May 2020), real-time traffic such as video, voice, and content sharing used **Electronic Code Book** AES-256/(**ECB)** mode, which is considered insecure.

This behavior changed with the introduction of Zoom client 5.0 and the switch to **Galois/counter mode** AES256/(**GCM**)[146, 147].

According to Zoom, in meetings where all participants use Zoom clients and the meeting is not recorded, all video, audio, screen sharing, and chat content is encrypted at the sending client and not decrypted at any point until it reaches the receiving clients. Zoom clients include the Zoom app running on a desktop or mobile device and Zoom Rooms, which are specialized Zoom video conferencing hardware. Zoom claims never to have built a mechanism to decrypt live meetings for lawful intercept purposes, nor one to insert participants into meetings without it being reflected in the participant list [148].

However, because Zoom is closed source, these claims are not easily verifiable. Zoom aims at keeping data encrypted throughout as much of the transmission as possible, Zoom maintains and controls the key management system in the cloud.

Although Zoom claims that its calls are encrypted, it does not use end-to-end encryption [78]. Instead, Zoom encrypts calls with AES-256 and shares the encryption key with Zoom servers around the globe. This potentially gives Zoom servers full access to the audio and video streams, although the company has stated that no user content is available to its employees or servers once encrypted [35, 147].

5.5.3 Secure Infrastructure Deployment. Connectors allow the Zoom Cloud infrastructure to connect with other services, including traditional telephones, Conference Rooms, Skype for Business, Cloud Recording, and Live Streaming. When a user joins a Zoom meeting from a traditional telephone, encryption cannot be applied, which is a general technical limitation. For enterprise customers, an on-premises solution is available for the entire meeting infrastructure. Additionally, enterprise customers have the option to run certain versions of connectors within their own data centers to manage the decryption and translation process themselves [35]. Researchers have also found encryption keys on Zoom servers in China (where the company has development sites) even when no Chinese participants are in the call [72]. This opens the possibility that the Chinese government could eavesdrop on calls. Zoom has reacted to this finding by allowing paying customers to opt out of having data routed through China [87].

5.5.4 Privacy. A number of issues with Zoom have attracted public attention in early 2020, most notably call hijacking or "Zoombombing". Calls that are not set to private or password-protected can be accessed by anyone who inputs the 9 –11-digit meeting code, and researchers have shown how valid meeting codes could easily be identified (something Zoom now says it prevents) [59]. Zoom has also had to make changes to its iPhone and iPad apps to stop Facebook being able to collect data about users [142]. In 2019, Zoom was forced to fix a problem that could have allowed websites to turn Mac users' cameras on without permission [65].

As a reaction to unfavorable news reports in early 2020, Zoom announced a 90-day security plan, starting in April 2020, that delivered several important changes; enabling AES 256-bit GCM encryption standard for data in transit, allowing account admins to designate a data center region of choice and allowing them to set the routing, and implementing the Zoom dashboard, which allows users to see how meetings are connected to Zoom data centers. In addition, meetings now have passwords and waiting rooms turned on by default. Zoom has also begun security and privacy reviews of their services with external experts and prepared a transparency report [99, 144]. Before 2020, Zoom's privacy policy was arguably not user-friendly. By downloading the app, the user granted the company permission to freely collect, process, and share personal data. This improved significantly in 2020 with better disclosures regarding issues of data safety and data rights. However, privacy issues still exist with regard to targeted advertising and third-party tracking [57, 145].

5.6 Skype-Microsoft

Skype was founded by Janus Friis and Niklas Zennström in 2003. After an **Initial Public Offering (IPO)** in 2010, Skype was acquired by Microsoft in 2011. Today, Skype is part of Microsoft as a subsidiary headquartered in Luxembourg. Skype announced 214 billion Skype-to-Skype international minutes in 2014 and 300 million active users per month in 2016 [64, 67, 118]. Skype was the first P2P VoIP client and initially offered free IM and Internet telephony to save costs on international audio calls between computers. Later, Skype added the capability to call landline and mobile numbers from Skype clients via a telephony gateway as a paid service (SkypeOut) [110]. Currently, the Skype client provides free IM, audio calls, and high definition video calls on a public cloud with multiple participants. Skype is available closed-source on many platforms, including Windows, macOS, Linux, Android, iOS, Windows Mobile, Xbox, and Amazon Alexa. Although Skype, Skype for Business, and Teams (the successor of Skype for Business) are all offered by Microsoft, they use different and incompatible underlying technologies. Information about Skype's architecture and protocols is mostly available from the time before Skype's acquisition by Microsoft.

5.6.1 Authentication and Access Permissions. Microsoft runs central login servers for the authentication and registration of clients and users. Skype's privacy settings give the user control over the sharing of their contact details, online status, and who can call or IM them. As soon as a user adds a contact to the contact list, Skype asks for contact details. If the new contact accepts the request, both users are then able to see each other's online status. Contact lists are stored by Microsoft and can be exported via the web interface of Skype. The Skype client keeps local records of contact lists, communication history, and IM content [111]. This introduces attack vectors that depend on the implementation of the client operating system security and privacy capabilities.

5.6.2 Encryption. Skype uses standard cryptographic primitives, such as AES 256-bit integer counter-mode, for all signaling and communication between the clients to prevent spoofing, tampering, and information disclosure. Supernodes and relay nodes do not have the keys to decipher the communication [111]. 1024-bit RSA is used to negotiate the symmetric AES keys. The user's public keys are certified during login, using 1536 or 2048-bit RSA, by central servers, which perform the authentication of the Skype users with their Skype-Name and password [15]. However, the use of proprietary protocols based on standard cryptographic primitives in a "black box" introduces the possibility of backdoors and a master key that can decrypt all communication [89].

5.6.3 Secure Infrastructure Deployment. The Skype architecture consists of three types of nodes; ordinary nodes, supernodes, and relay nodes. The ordinary node is the software installed by the end user. Supernodes are ordinary nodes that are dynamically selected to perform additional functions in the Skype architecture, such as searching for other nodes. To be elected as a supernode, the client needs to meet several requirements, including having a public IP address, sufficient memory, bandwidth, and uptime. The relay hosts relay media and signaling traffic for nodes, which cannot connect to peers directly due to firewalls or **Network Address Translation (NATs)**. The mechanism used by Skype for NAT/firewall traversal issues, such as firewalls blocking new incoming connections, is a variation of STUN [9, 102, 111].

When a Skype client starts, it binds local listening sockets for random high (>1024) TCP and UDP ports, port 443 (TCP/UDP), and port 80 (TCP). The Skype client uses TCP for signaling traffic and prefers UDP for voice, video, and file transfer traffic. In case UDP is not available, Skype can use TCP for media streams which introduces an additional overhead. Skype checks network connectivity during the login process to verify if the outgoing UDP/TCP ports are available and what kind of network address translation is used by the client's network. Client status updates and online status indicators of contacts are also carried out via the P2P architecture [111]. As soon

as a user places a call via the Skype client and the destination's network address is not in the local cache, supernodes are used to search for the recipient's network address and its associated supernode. The caller node can then establish a session for chat, audio, video, file transfer, or authorization requests either directly or through a P2P relay [111].

5.6.4 Privacy. Skype's most significant security and privacy issue is the confirmation by Microsoft that Skype allows eavesdropping by design, by switching from client-to-client to client-to-server encryption. Skype also works with in-country partners, admitting that "there is a possibility that your communications and personal data could be stored, monitored, or blocked and made available to authorized local parties, for instance law enforcement, subject to the local legal standards" [111]. In addition, the Snowden revelations showed that Skype participates in the NSA's PRISM program [82].

5.7 Google Meet - Google

Google Meet launched in March 2017 and replaced Google Hangouts and Google Chat within a single application [54]. The Meet client is available on Android, iOS, and supported on major web browsers, including Chrome, Mozilla Firefox, Microsoft Edge, and Apple Safari. Dedicated hardware products are also available, for example, the ASUS Google Meet Kit. Meet is available for free and surpassed 100 million daily meeting participants in April 2020 [112].

5.7.1 Authentication and Access Permissions. Google Meet offers strong controls for meeting hosts, including admitting, removing, and muting participants. All participants must have a Google account, i.e., anonymous users are not allowed, which, according to Google, makes the platform safer. The Meet codes for joining a meeting are 10 characters long, with 25 characters in the set, e.g., https://meet.google.com/yxo-cvxt-sko, and are not easy to guess or brute-force.

Invitation of participants is organized by e-mail address. Participants are treated as external if their e-mail domain is different from the host's. To join a meeting, external participants need to be included in the calendar invite, be invited by in-domain participants, or submit a request to join the meeting, which must be accepted by an in-domain participant. External participants can only join meetings 15 minutes in advance to reduce the time window for a brute force attack against the meeting code [54, 55].

5.7.2 Encryption. All Meet meetings are encrypted in transit from each participant to the Google Cloud infrastructure, but not end-to-end between the participants. Google, as hub of the communication, is able to access all meeting content. Meeting recordings are encrypted in transit and at rest and stored in Google Drive. Meet uses common security protocols such as DTLS and SRTP. For data in transit, Google Cloud services use the load balancing front end with the BoringSSL open source implementation and TLS1.3, RSA2048 for authentication, Curve25519 for key exchange, AES-128-GCM for encryption, and SHA384 as hash function. All key material is under Google's control. Unique encryption keys are generated by Meet for every participant and meeting, which only live as long as the meeting. The encryption keys are never stored on disk and are transmitted during meeting initialization in an encrypted **remote procedure call** (**RPC**) [38, 55, 112].

5.7.3 Secure Infrastructure Deployment. Google Meet is operated on Google Cloud and claims to be compliant with various regulations including GDPR, HIPAA, and **Children's Online Privacy Protection Act** (**COPPA**). Google claims that Meet data are not used for advertising or sold to third parties. Google provides detailed information regarding security aspects such as staff training, dedicated privacy and security teams, internal audits, collaboration with the security research community, operational security, and data access [41, 42].

5.7.4 Privacy. The requirement to use a Google account for Meet is a privacy disadvantage but improves security by allowing only authenticated users in a meeting. Google's public hosted UC platform is closed source code and only some information is provided, for example, the use of DTLS and SRTP. For administrators, Google Meet keeps audit information for six months, including the event name, description, and participant identifiers. External participant identifiers are obscured or not shown, but for internal participants, most information (including name, e-mail, country, and IP address) is shown in clear text. By using the Google Meet service, the user needs to trust Google and has very limited control of the Meet platform, e.g., key material storage [37, 40].

6 DISCUSSION

Selection criteria for UC platforms and usage guidelines can improve the security and privacy for users. Examples include; enforcing encryption by default and making sure it is end-to-end, locking and password-protecting meetings, holding unauthenticated users in a waiting room so the organizer can check their identity before admitting them to the call, monitoring the participant list to ensure no unknown participant joins, and acquiring consent from participants for meeting recordings, being aware that audio-only participants calling via a regular phone dial-in option or protocol gateways such as SIP/H.323/WebRTC disables end-to-end encryption and being aware that file and screen-sharing capabilities could accidentally disclose sensitive information or be used to spread malicious programs. Organizations such as the UK's National Cyber Security Centre provide further guidance [80, 81].

However, selection criteria and usage guidelines rely on the availability of security and privacy features in UC platforms. In this section, we therefore evaluate the security and privacy features offered by the 10 UC platforms compared to the generic mitigations discussed in Section 3.5. In addition, we discuss open issues and missing features, and outline possible avenues for future work.

6.1 Comparison of Security/Privacy Features

End-to-end encryption and open-source architectures are two fundamental security and privacy mitigations for UC. End-to-end encryption ensures confidentiality and integrity of the communication even against the infrastructure provider or law enforcement requests directed at the provider. Open-source architectures are essential to allow public reviews of security and privacy properties and thereby increase the transparency of the UC service. Among the UC platforms we reviewed, only Signal Messenger and Jitsi are open source, and only Signal, Wire, and WhatsApp fully support end-to-end encryption, while Jitsi supports it on Chromium 83 or newer.

Authentication and access permissions. All architectures use some form of authentication, such as username/password, except for Jitsi, which does not require authentication and instead uses random passphrases for access to ad hoc meetings. This can help provide anonymity or pseudonymity for the users. The *mobile-centered architectures*, i.e., WhatsApp, Signal, and Wire, rely on phone numbers for authentication, which can be spoofed or expose the privacy of the user. Architectures that offer meeting scheduling, such as Wire, Zoom, Teams, Slack, Webex, and Google Meet use role-based access permissions so that authenticated users can schedule meetings, invite meeting participants, and have moderator permissions during the meeting.

Encryption. For data in transit, most platforms use adequate security primitives and algorithms. Zoom updated their encryption during their 90-day security improvement program in 2020. However, the use of proprietary and closed source encryption primitives, for example, in Skype, is problematic because it enables Microsoft's cooperation with the NSA PRISM program. Similar issues

may exist in other proprietary platforms. End-to-end encryption is supported by WhatsApp, Signal, Wire, and Jitsi. The other architectures only encrypt between a client and the cloud infrastructure.

For data at rest, such as shared files, meeting recordings, and backups, details are unclear or unknown for WhatsApp and Zoom. The other architectures use encryption to store the data either in the cloud, or locally on the device and rely on protection mechanisms on the user's device, such as Windows disk encryption.

Secure infrastructure deployment and DDoS prevention. WhatsApp, Skype, Teams, Slack, and Google Meet are only available as public cloud offerings, which means that many implementation details are unknown, for example, details surrounding data separation, backups, or DDoS prevention. Cisco Webex and Zoom support hybrid deployments, where parts of the architecture are deployed on-premises, for example, the key management with Cisco Webex. In contrast, Signal, Jitsi, and Wire are available as publicly reviewed source code and can be deployed in a private cloud or on-premises, which provides significantly more control and responsibility for the implementation and operation of the architecture.

Repudiation/Plausible Deniability. Plausible deniability is supported and documented only by the mobile-centered architectures WhatsApp, Signal, and Wire. For the remaining architectures, information regarding plausible deniability is not available, which means it is most likely not supported.

Secure data deletion on client or server. The mobile-centered architectures, along with Jitsi, support secure data deletion. The other platforms support it partially, typically for data on the client-side. There is no information about server-side data deletion for Skype and Webex.

Secured meeting invitations. Secured meeting invitations are only relevant on platforms with scheduling capabilities, that is, Wire, Jitsi (one-time meeting name), Zoom, Cisco Webex, and Google Meet. Those platforms mainly use e-mail for calendar invites (e.g., in iCal format) and include the joining instructions in plaintext, including a hyperlink to the meeting, meeting passcode or PIN, and telephone dial-in numbers. This approach, while convenient, does not provide a secure way to share meeting details. None of the platforms provides a way to exchange meeting details in encrypted form or on a separate communication channel, although the end user might be able to do this manually.

Anonymous communication and undetectability. Anonymity and undetectability are challenging requirements for UC due to the network requirements for low latency, low jitter, use of UDP, and the large volume of video/audio data. Some architectures (Signal and Skype) support or require relaying of the media or signaling traffic through their infrastructure or through special nodes, which helps to provide anonymity, but does not provide undetectability. Other architectures (WhatsApp, Wire and Jitsi) use end-to-end encryption, which reveals metadata including IP addresses and traffic volumes at UC specific ports. The use of a mixing network such as Tor to provide anonymity is typically seen as infeasible due to latency and lack of support for UDP.

6.2 Open Issues

The discussion above, summarized by the gray areas of concern in Table 4, shows that most current UC platforms provide reasonably good mitigations against security threats. However, mitigations against privacy threats are far less available, and most UC platforms do not provide anonymous communication, undetectability, privacy-preserving authentication, or secure meeting invitations. In addition, formal analysis of the privacy and security properties of UC platforms is an open issue.

Anonymous communication. Anonymity in UC allows anonymous meeting participants. The contents of voice or video streams may break anonymity toward other meeting participants, but anonymity can still hold against the UC provider and passive observers on the network. Anonymity is not limited to protecting identities and personally identifiable data, but also needs to protect metadata that may allow re-identification in combination with other data (e.g., fingerprinting attributes or IP addresses that can be resolved to identities by Internet providers).

Onion routing, most notably Tor, is an established method for anonymous communication on the Internet, but its use for UC is limited because most UC platforms require UDP traffic and also because onion routing introduces additional delays [113, 120, 121]. The first limitation could be overcome, for example, by running WebRTC traffic over a VPN connection in TCP mode, which in turn runs over Tor. However, this creates a complex technology stack which can cause disconnects and a bad user experience in UC sessions [48]. Alternatively, a virtual machine such as Whonix can force all network traffic through the Tor network. The overhead of running a virtual machine may be justifiable for some use cases, but network delays may still lead to a degraded UC experience [134].

In other implementations, WebRTC has been used as a censorship circumvention tool by proxying traffic using WebRTC thereby preventing IP blocking (e.g., Snowflake or uProxy) [31, 32]. However, this approach does not address anonymous communication or undetectability of the WebRTC-based UC communication dataflows.

Undetectability and unobservability. Undetectability in UC would allow participants to hide the fact that they are participating in a meeting, for example, from the UC provider or passive observers on the network. In addition, undetectability could allow users to hide which UC platform was used for a meeting. However, undetectability in the sense of shadow participants who hide their presence from other participants is likely to be an undesirable feature. For example, Zoom publicly disavows that their platform allows shadow participants [148].

Undetectability can be provided by adding dummy traffic or by hiding traffic within other traffic (steganography). For example, VoIP cover traffic has been used to hide other voice traffic encoded with a low bit-rate speech codec, for instance, so that the VoIP cover sounds meaningful and the hidden speech is indistinguishable from the cover [61, 88, 116, 130]. However, due to the large traffic volumes in UC applications that include video traffic, these existing approaches may not be feasible and further research is needed on how undetectability of UC can be supported.

Privacy-preserving authentication. Privacy-preserving authentication for UC could be realized as a lack of formal authentication, which is done, for example, in Jitsi where knowledge of the meeting code constitutes authorization to access a meeting. Other options for privacy-preserving authentication could be to separate authentication to meeting organizers and authentication to the UC provider, and to ensure that the identity of authenticated users is not linkable to their actions on a UC platform (e.g., meeting participation).

Work that provides this kind of unlinkability already exists, for example, in smart metering where fine-grained power consumption is hidden from the electricity provider [98], and in electronic toll pricing where travel routes are hidden from the toll provider [8]. However, we are not aware of work that explores similar approaches for UC.

Secured meeting invitations. Secure meeting invites should be realized so that none other than the invitee can read content of the invite, providing confidentiality and supporting anonymity for meeting participants. In addition, undetectable meeting invites could be a desirable feature, so that none other than the invitee can learn that the message is a meeting invite.

Most platforms rely on e-mail for meeting invites, which could be encrypted with extra effort by the scheduler and participants. However, in practice, e-mail encryption is not widely adopted, particular across organizational boundaries. A first step toward more secure meeting invites would be to have a separate (ideally secure) communication channel for sharing meeting PINs. For example, the UC platform could distribute PINs via SMS. However, SMS is not a secure channel [5] and revealing phone numbers to the UC platform may constitute a privacy risk. Alternatively, PINs could be shared manually via a secure IM channel, which is unlikely to be adopted due to the additional overhead for meeting organizers and participants. Further work is therefore needed to develop solutions for secure meeting invites that can be easily integrated into UC platforms.

Formal analysis. There are two directions in which the evaluation presented here could be enhanced using formal and quantitative models. Firstly, the reference system, the relevant security and privacy properties, and the commercial systems under study have all been described informally here. It would be a worthwhile exercise to develop formal descriptions of the reference system and its properties, as has been done for other areas in privacy and security (e.g., electronic voting [23]). This would allow formal proofs of any inter-dependencies between these, as well the development of tool support for checking practical systems. Because most unified communication platforms are closed-source, and all are of a high complexity, full formal verification of any platform against its required properties is not realistic; however, a focused analysis of the core protocols used is more realistic, as was done for Signal protocols [21]. The second dimension in which we could enhance the analysis is through quantification. The current evaluation presents a qualitative assessment of individual properties only. As a consequence, judgments on the platforms overall security and privacy are largely missing. We include no verdict on the impact of any of the individual privacy and security risks identified, and indeed these will strongly depend on the context of any usage of unified communication. This will also strongly influence the likelihood of particular risks materializing. The measurement of privacy risks in particular is a known challenging problem [129], but a system that allows users to configure the identified risks with impacts and likelihoods, and then combines the individual risk evaluations in some way, would be useful in selecting the right platform for each use case.

Transparency and awareness. Users of UC platforms are often not aware what data these platforms collect, store, and share, and they are also not aware of the potential consequences, for example, following a data breach. Even though regulations such as the GDPR have introduced legal obligations for companies to make collected data available to users, in practice this process is difficult to navigate for users, and not all companies comply with their duties or the mandated timeframes [126].

Personal information feedback tools could help to improve user awareness regarding their processed private data. However, more research is needed on the usability of these tools. In addition, companies may be unlikely to provide their users with these tools unless mandated by laws or regulations. As a result, a promising research direction could also be a client-side approach that automates the entire process for the user, including requesting data from companies, and presenting the information in a usable manner [128].

7 CONCLUSION

This article presents the first comprehensive survey of privacy and security in unified communications. Based on a generic UC architecture, we systematically review possible security and privacy threats, as well as possible existing mitigations, following the STRIDE and LINDDUN methodologies. We then review the security and privacy features of 10 existing UC platforms. While most platforms provide many of the obvious security features, our survey has identified that most

platforms do not provide privacy properties, giving researchers and platform developers opportunities for further work. In particular, there are significant challenges for anonymity and undetectability, as well as challenges to enhance UC features such as meeting invitations with security and privacy features.

The framework we have presented in this article enables users to make informed decisions on the selection of unified communication platforms, particularly with regard to their security and privacy features. Unfortunately, users and organizations increasingly prefer to *consume* UC public cloud services as a commodity without much regard to security and privacy. Ensuring that organizations prioritize responsible security and privacy properties over the convenient use of cloud services is a wider issue of socio-technical systems that also deserves further study.

REFERENCES

[1] Nasser M. Al-Fannah. 2017. One leak will sink a ship: WebRTC IP address leaks. In *Proceedings of the 2017 International Carnahan Conference on Security Technology*. IEEE, 1–5. DOI: https://doi.org/10.1109/CCST.2017.8167801

[2] Ryan Amos, Gunes Acar, Elena Lucherini, Mihir Kshirsagar, Arvind Narayanan, and Jonathan Mayer. 2021. Privacy policies over time: Curation and analysis of a million-document dataset. In *Proceedings of the Web Conference 2021*. ACM, Ljubljana, Slovenia, 22. DOI: https://doi.org/10.1145/3442381.3450048

[3] Ross J. Anderson and Fabien A. P. Petitcolas. 1998. On the limits of steganography. *IEEE Journal on Selected Areas in Communications* 16, 4 (1998), 474–481. DOI: https://doi.org/10.1109/49.668971

[4] Benjamin Andow, Samin Yaseer Mahmud, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Serge Egelman. 2020. Actions speak louder than words: entity-sensitive privacy policy and data flow analysis with policheck. In *Proceedings of the 29th {USENIX} security Symposium ({USENIX} security 20)*. USENIX, 985–1002.

[5] Iosif Androulidakis. 2012. *SMS Security Issues*. Springer US, Boston, MA, 63–74. https://doi.org/10.1007/978-1-4614-1650-0_5

[6] Cosimo Anglano. 2014. Forensic analysis of whatsapp messenger on android smartphones. *Digital Investigation* 11, 3 (2014), 201–213. DOI: https://doi.org/10.1016/j.diin.2014.04.003 Special Issue: Embedded Forensics.

[7] AN.ON. [n.d.]. AN.ON - anonymity.online. Retrieved from https://anon.inf.tu-dresden.de/index_en.html. Accessed: 15-03-2020.

[8] Josep Balasch, Alfredo Rial, Carmela Troncoso, Bart Preneel, Ingrid Verbauwhede, and Christophe Geuens. 2010. PrETP: privacy-preserving electronic toll pricing.. In *Proceedings of the 19th USENIX Security Symposium*. USENIX Association, Washington, DC, 63–78.

[9] Salman Baset and Henning Schulzrinne. 2005. An analysis of the skype peer-to-peer internet telephony protocol. In *Proceedings of the IEEE INFOCOM* (01 2005). DOI: https://doi.org/10.1109/INFOCOM.2006.312

[10] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. 2004. The Secure Real-time Transport Protocol (SRTP). RFC 3711 (Proposed Standard). 56 pages. DOI: https://doi.org/10.17487/RFC3711 Updated by RFCs 5506, 6904.

[11] Cullen Jennings, Henrik Boström, Jan-Ivar Bruaroey, Adam Bergkvist, Daniel C. Burnett, Anant Narayanan, Bernard Aboba, Taylor Brandstetter. 2017. WebRTC 1.0 Real-time.Communication between browsers. https://www.w3.org/TR/webrtc/ Accessed: 30-06-2021.

[12] Berliner Datenschutz Beauftragte. 2020. Hinweise für berliner verantwortliche zu anbietern vonvideokonferenzdiensten. Retrieved from https://www.datenschutz-berlin.de/fileadmin/user_upload/pdf/orientierungshilfen/2020-BlnBDI-Hinweise_Berliner_Verantwortliche_zu_Anbietern_Videokonferenz-Dienste.pdf. Accessed: 24-09-2020.

[13] Daniel J. Bernstein. 2006. Curve25519: New diffie-hellman speed records. In *Proceedings of the Public Key Cryptography - PKC 2006*, Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin (Eds.). Springer Berlin Heidelberg, Berlin, 207–228.

[14] Daniel J. Bernstein. 2008. Chacha, a variant of Salsa20. In *Proceedings of the Workshop Record of SASC*, Vol. 8. 3–5.

[15] Tom Berson. 2005. Skype security evaluation. Retrieved from http://www.anagram.com/berson/skyeval.pdf. Accessed: 18-07-2019.

[16] Nikita Borisov, Ian Goldberg, and Eric Brewer. 2004. Off-the-record communication, or, why not to use PGP. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*. Association for Computing Machinery, New York, NY, 77–84. DOI: https://doi.org/10.1145/1029179.1029200

[17] Jan Camenisch, Maria Dubovitskaya, Anja Lehmann, Gregory Neven, Christian Paquin, and Franz-Stefan Preiss. 2013. Concepts and languages for privacy-preserving attribute-based authentication. In *Proceedings of the Policies and Research in Identity Management*, Simone Fischer-Hübner, Elisabeth de Leeuw, and Chris Mitchell (Eds.). Springer Berlin Heidelberg, Berlin, 34–52.

[18] Dell Cameron. 2020. Edward snowden tells you what encrypted messaging apps you should use. Retrieved from https://www.dailydot.com/layer8/edward-snowden-signal-encryption-privacy-messaging/. Accessed: 04-08-2019.
[19] Dana Casielles. 2020. Cisco, microsoft, zoom: Comparing one to another. Retrieved from https://www.nojitter.com/team-collaboration-tools-workspaces/cisco-microsoft-zoom-comparing-one-another?_mc=NL_NJ_EDT_NJ_weekly_20200901cid=NL_NJ_EDT_NJ_weekly_20200901elq_cid=26974084elq_mid=99292. Accessed: 24-09-2020.
[20] Chi-Tung Chen and Cheng-Chi Lee. 2015. A two-factor authentication scheme with anonymity for multi-server environments. *Security and Communication Networks* 8, 8 (2015), 1608–1625. DOI: https://doi.org/10.1002/sec.1109 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/sec.1109.
[21] Cohn-Gordon Katriel, Cremers Cas, Dowling Benjamin, Garratt Luke, and Stebila Douglas. 2017. A formal security analysis of the signal messaging protocol. In *Proceedings of the 2017 IEEE European Symposium on Security and Privacy*. 451–466. DOI: https://doi.org/10.1109/EuroSP.2017.27
[22] Josh Constine. 2014. The whatsapp architecture facebook bought for 19 billion USD. High Scalability. Retrieved from http://highscalability.com/blog/2014/2/26/the-whatsapp-architecture-facebook-bought-for-19-billion.html. Accessed: 11-05-2019.
[23] Stéphanie Delaune, Steve Kremer, and Mark Ryan. 2009. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security* 17, 4 (2009), 435–487.
[24] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. 2011. A privacy threat analysis framework: Supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering* 16, 1 (2011), 3–32.
[25] B. Desruisseaux. 2009. Internet Calendaring and Scheduling Core Object Specification (iCalendar). RFC 5545 (Proposed Standard). 168 pages. DOI: https://doi.org/10.17487/RFC5545 Updated by RFCs 5546, 6868, 7529, 7953, 7986.
[26] Cynthia Dwork. 2006. Differential privacy. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*. LNCS, Vol. 4052. Springer, Venice, Italy, 1–12.
[27] European Commission. 2017. Commission fines facebook 110 million euro for providing misleading information about whatsapp takeover. (2017). Retrieved from http://europa.eu/rapid/press-release_IP-17-1369_en.htm. Accessed: 19-06-2019.
[28] European Data Protection Supervisor (EDPS). 2020. Outcome of own-initiative investigation into EU institutions' use of microsoft products and services. Retrieved from https://edps.europa.eu/sites/edp/files/publication/20-07-02_edps_euis_microsoft_contract_investigation_en.html. Accessed: 24-09-2020.
[29] European Union. 2016. General data protection regulation. Retrieved from https://eur-lex.europa.eu/eli/reg/2016/679/oj. Accessed: 26-04-2020.
[30] Alexandros Fakis, Georgios Karopoulos, and Georgios Kambourakis. 2020. Neither denied nor exposed: Fixing WebRTC privacy leaks. *Future Internet* 12, 5 (2020), 92.
[31] David Fifield. 2017. *Threat modeling and circumvention of internet censorship*. Ph.D. Dissertation. EECS Department, University of California, Berkeley.
[32] David Fifield and Mia Gil Epner. 2016. Fingerprintability of WebRTC. arXiv:1605.08805. Retrieved from https://arxiv.org/abs/1605.08805.
[33] J. Fischl, H. Tschofenig, and E. Rescorla. 2010. Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS). RFC 5763 (Proposed Standard). 37 pages. DOI: https://doi.org/10.17487/RFC5763
[34] Tilman Frosch, Christian Mainka, Christoph Bader, Florian Bergsma, Jörg Schwenk, and Thorsten Holz. 2016. How secure is textsecure?. In *Proceedings of the 2016 IEEE European Symposium on Security and Privacy*. IEEE, 457–472.
[35] Oded Gal. 2020. The facts around zoom and encryption for Meetings/Webinars. Retrieved from https://blog.zoom.us/wordpress/2020/04/01/facts-around-zoom-encryption-for-meetings-webinars/. Accessed: 11-05-2020.
[36] Gartner I. T. Glossary. [n.d.]. Unified Communications (UC). Retrieved from https://www.gartner.com/it-glossary/unified-communications-uc. Accessed: 30-10-2018.
[37] Google. [n.d.]. Google meet audit log. Retrieved from https://support.google.com/a/answer/9186729?hl=en. Accessed: 15-06-2021.
[38] Google. [n.d.]. Prepare your network for meet video calls. Retrieved from https://support.google.com/a/answer/1279090. Accessed: 14-06-2021.
[39] Google. [n.d.]. Real-time communication for the web. Retrieved from https://webrtc.org/. Accessed: 29-12-2020.
[40] Google. 2017. encryption in transit in google cloud. Retrieved from https://cloud.google.com/security/encryption-in-transit. Accessed: 14-06-2021.
[41] Google. 2019. Google security whitepaper. Retrieved from https://cloud.google.com/security/overview/whitepaper. Accessed: 13-06-2021.
[42] Google. 2021. Google meet security and privacy for admins. Retrieved from https://support.google.com/a/answer/7582940?hl=en&ref_topic=7302923. Accessed: 14-06-2021.

[43] M. Handley, V. Jacobson, and C. Perkins. 2006. SDP: Session description protocol. RFC 4566 (Proposed Standard). 49 pages. DOI: https://doi.org/10.17487/RFC4566

[44] Marit Hansen, Peter Berlich, Jan Camenisch, Sebastian Clauß, Andreas Pfitzmann, and Michael Waidner. 2004. Privacy-enhancing identity management. *Information Security Technical Report* 9, 1 (2004), 35–44. DOI: https://doi.org/10.1016/S1363-4127(04)00014-7

[45] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G. Shin, and Karl Aberer. 2018. Polisis: automated analysis and presentation of privacy policies using deep learning. In *Proceedings of the 27th {USENIX} Security Symposium ({USENIX} Security 18)*. 531–548.

[46] Bill Haskins. 2018. 2018 Worldwide unified communications forecast. Retrieved from https://insight.wainhouse.com/reportaction/UC-FCST18-UCaaS-WW/Marketing. Accessed: 01-11-2018.

[47] Howard and Lipner. 2006. *The Security Development Lifecycle*. Microsoft Press. DOI: https://blogs.msdn.microsoft.com/microsoft_press/2016/04/19/free-ebook-the-security-development-lifecycle/.

[48] David Huerta. 2013. [tor-talk] WebRTC via tor. Retrieved from https://www.mail-archive.com/tor-talk@lists.torproject.org/msg08733.html. Accessed: 12-08-2021.

[49] Hyperledger FABRIC. [n.d.]. MSP implementation with identity mixer. Retrieved from https://hyperledger-fabric.readthedocs.io/en/release-1.4/idemix.html. Accessed: 12-03-2020.

[50] Ian Goldberg, David Goulet, and Jurre van Bergen. [n. d.]. Off-the-Record Messaging. https://otr.cypherpunks.ca/ Accessed: 13-05-2019.

[51] Emil Ivov. 2020. This is what end-to-end encryption should look like. Retrieved from https://jitsi.org/e2ee. Accessed: 27-04-2020.

[52] jitsi.org. [n.d.]. Jitsi meet security and privacy. Retrieved from https://jitsi.org/security/. Accessed: 27-04-2020.

[53] jitsi.org Community. 2018. Information about jitsi. Jitsi Community Forum. Retrieved from https://community.jitsi.org/t/information-about-jitsi/15426/3. Accessed: 02-10-2019.

[54] Scott Johnston. 2017. Meet the new hangouts. Retrieved from https://www.blog.google/products/g-suite/meet-the-new-enterprise-focused-hangouts/. Accessed: 16-06-2021.

[55] Smita Hashim Karthik Lakshminarayanan. 2020. Secure connections: How google meet keeps your video conferences protected. Retrieved from https://cloud.google.com/blog/products/g-suite/how-google-meet-keeps-video-conferences-secure. Accessed: 13-06-2021.

[56] Jonathan Katz, Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. 1996. *Handbook of Applied Cryptography*. CRC press.

[57] Girard Kelly. 2020. What zoom's privacy policy changes mean for you. Common Sense. Retrieved from https://www.commonsense.org/education/articles/what-zooms-privacy-policy-changes-mean-for-you. Accessed: 13-12-2020.

[58] Kenneth E. Kendall, Julie E. Kendall. 2019. *Systems Analysis and Design*. Pearson Education Limited.

[59] Swati Khandelwal. 2020. Zoom bug could have let uninvited people join private meetings. Retrieved from https://thehackernews.com/2020/01/zoom-meeting-password.html. Accessed: 11-05-2020.

[60] Paul C. Kocher. 1996. Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In *Proceedings of the Advances in Cryptology — CRYPTO'96*, Neal Koblitz (Ed.). Springer Berlin Heidelberg, Berlin, 104–113.

[61] Christian Kratzer, Jana Dittmann, Thomas Vogel, and Reyk Hillert. 2006. Design and evaluation of steganography for voice-over-IP. In *Proceedings of the 2006 IEEE International Symposium on Circuits and Systems*. IEEE, 4–pp.

[62] H. Krawczyk, M. Bellare, and R. Canetti. 1997. HMAC: Keyed-hashing for message authentication. RFC 2104 (Informational). 11 pages. DOI: https://doi.org/10.17487/RFC2104 Updated by RFC 6151.

[63] H. Krawczyk and P. Eronen. 2010. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869 (Informational). 14 pages. DOI: https://doi.org/10.17487/RFC5869

[64] Ujjwal Kumar. 2016. Skype has over 300 million monthly active users, microsoft announces at build 2016. Retrieved from https://windowsreport.com/skype-number-of-users/. Accessed: 22-05-2019.

[65] Ravie Lakshmanan. 2019. Zoom security flaw could let websites turn on your mac's webcam without permission. Retrieved from https://thenextweb.com/security/2019/07/09/zoom-security-flaw-could-let-websites-turn-on-your-macs-webcam-without-permission/. Accessed: 11-05-2020.

[66] Scott Lederer, Jason I. Hong, Anind K. Dey, and James A. Landay. 2004. Personal privacy through understanding and action: Five pitfalls for designers. *Personal and Ubiquitous Computing* 8, 6 (Nov. 2004), 440–454. DOI: https://doi.org/10.1007/s00779-004-0304-9

[67] Dave Lee. 2011. Profile: How skype connected. Retrieved from https://www.bbc.com/news/technology-13350425 Accessed: 21-05-2019.

[68] Micah Lee. 2016. Battle of the secure messaging apps: How ginal beats whatsapp. The Intercept. Retrieved from https://theintercept.com/2016/06/22/battle-of-the-secure-messaging-apps-how-signal-beats-whatsapp/. Accessed: 04-08-2019.

[69] Timothy Libert. 2018. An automated approach to auditing disclosure of third-party data collection in website privacy policies. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, Lyon, France, 207–216. DOI: https://doi.org/10.1145/3178876.3186087

[70] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph Reagle. 2002. The platform for privacy preferences. Retrieved from https://www.w3.org/TR/P3P/. Accessed: 22-03-2020.

[71] R. Mahy, P. Matthews, and J. Rosenberg. 2010. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766 (Proposed Standard). 67 pages. DOI: https://doi.org/10.17487/RFC5766 Updated by RFCs 8155, 8553.

[72] Bill Marczak and John Scott-Railton. 2020. Move fast and roll your own crypto. Retrieved from https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/. Accessed: 11-05-2020.

[73] Moxie Marlinspike. 2013. Advanced cryptographic ratcheting. Retrieved from https://signal.org/blog/advanced-ratcheting/. Accessed: 19-05-2020.

[74] Moxie Marlinspike. 2013. Simplifying OTR deniability. Retrieved from https://signal.org/blog/simplifying-otr-deniability/. Accessed: 19-05-2020.

[75] Moxie Marlinspike. 2014. Free, worldwide, encrypted phone calls for iphone. Signal.org. Retrieved from https://signal.org/blog/signal/. Accessed: 04-08-2019.

[76] Moxie Marlinspike. 2018. A letter from amazon. Retrieved from https://signal.org/blog/looking-back-on-the-front/. Accessed: 04-08-2019.

[77] D. McGrew and E. Rescorla. 2010. Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP). RFC 5764 (Proposed Standard). 26 pages. DOI: https://doi.org/10.17487/RFC5764 Updated by RFC 7983.

[78] Yael Grauer Micah Lee. 2020. Zoom meetings aren't end-to-end encrypted, despite misleading markteing. Retrieved from https://theintercept.com/2020/03/31/zoom-meeting-encryption/. Accessed: 11-05-2020.

[79] Moni Naor. 2002. Deniable ring authentication. In *Proceedings of the Annual International Cryptology Conference*. Springer, 481–498.

[80] National Cyber Security Center. 2020. Video conferencing services: Security guidance for organisations. Retrieved from https://www.ncsc.gov.uk/guidance/video-conferencing-services-security-guidance-organisations. Accessed: 23-04-2020.

[81] National Cyber Security Centre. 2020. Video conferencing services: Using them securely. Retrieved from https://www.ncsc.gov.uk/guidance/video-conferencing-services-using-them-securely. Accessed: 23-04-2020.

[82] NSA. 2012. Users guide for PRISM skype collection. Retrieved from https://edwardsnowden.com/2015/01/05/users-guide-for-prism-skype-collection/. Accessed: 22-05-2019.

[83] Sameer Patil and Alfred Kobsa. 2009. Privacy considerations in awareness systems: Designing with privacy in mind. In *Proceedings of the Awareness Systems*. Springer, 187–206.

[84] Colin Percival. 2009. Stronger key derivation via sequential memory-hard functions. Retrieved from http://www.bsdcan.org/2009/schedule/attachments/87_scrypt.pdf. Accessed: 19-05-2020.

[85] C. Percival and S. Josefsson. 2016. the scrypt password-based key derivation function. RFC 7914 (Informational). 16 pages. DOI: https://doi.org/10.17487/RFC7914

[86] Trevor Perrin (editor) and Moxie Marlinspike. 2016. The double ratchet algorithm. Retrieved from https://signal.org/docs/specifications/doubleratchet/. Accessed: 19-05-2020.

[87] Jay Peters. 2020. Zoom will let paying customers pick which data center their calls are routed from. Retrieved from https://www.theverge.com/2020/4/13/21219835/zoom-data-center-call-routing-china-security-privacy-encryption. Accessed: 11-05-2020.

[88] Andreas Pfitzmann and Marit Hansen. 2010. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. Retrieved from https://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf. Accessed: 30-06-2021.

[89] Philippe Biondi, Fabrice Desclaux. 2006. silver needle in the skype. Retrieved from http://www.oklabs.net/wp-content/uploads/2012/06/bh-eu-06-Biondi.pdf. Accessed: 23-07-2019.

[90] Blair Pleasant. [n.d.]. UC cutting through the hype - what UC is and isn't. Retrieved from http://viewer.media.bitpipe.com/1206484657_637/1206511483_362/SearchUC-v5.pdf. Accessed: 04-11-2018.

[91] Nidhi Rastogi and James Hendler. 2017. Whatsapp security and role of metadata in preserving privacy. In *12th International Conference on Cyber Warfare and Security*. Academic Conferences and publishing limited, 269–275.

[92] Thomas Reisinger. 2020. Zoom security: I've researched problems with video conferencing for years – here's what you need to know. Retrieved from https://theconversation.com/zoom-security-ive-researched-problems-with-video-conferencing-for-years-heres-what-you-need-to-know-136330. Accessed: 25-07-2021.

[93] E. Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Proposed Standard). 160 pages. DOI: https://doi.org/10.17487/RFC8446
[94] E. Rescorla. 2019. security considerations for WebRTC. IETF. Retrieved from https://tools.ietf.org/html/draft-ietf-rtcweb-security-12. Accessed: 29-12-2020.
[95] Eric Rescorla. 2019. WebRTC security architecture. Retrieved from https://tools.ietf.org/html/draft-ietf-rtcweb-security-arch-20. Accessed: 28-04-2020.
[96] E. Rescorla and B. Korver. 2003. Guidelines for writing RFC text on security considerations. RFC 3552 (Best Current Practice). 44 pages. DOI: https://doi.org/10.17487/RFC3552
[97] E. Rescorla and N. Modadugu. 2006. Datagram transport layer security. RFC 4347 (Proposed Standard). 25 pages. https://doi.org/10.17487/RFC4347 Obsoleted by RFC 6347, updated by RFCs 5746, 7507.
[98] Alfredo Rial and George Danezis. 2011. Privacy-preserving smart metering. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*. ACM, New York, NY, 49–60. DOI: https://doi.org/10.1145/2046556.2046564
[99] Colleen Rodriguez. 2020. Zoom hits milestone on 90-Day security plan, releases zoom 5.0. Retrieved from https://blog.zoom.us/wordpress/2020/04/22/zoom-hits-milestone-on-90-day-security-plan-releases-zoom-5-0/. Accessed: 29-04-2020.
[100] Burton Rosenberg. 2010. *Handbook of Financial Cryptography and Security*. CRC Press.
[101] J. Rosenberg. 2010. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245 (Proposed Standard). 117 pages. DOI: https://doi.org/10.17487/RFC5245 Updated by RFC 6336.
[102] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. 2008. Session Traversal Utilities for NAT (STUN). RFC 5389 (Proposed Standard). 51 pages. DOI: https://doi.org/10.17487/RFC5389 Updated by RFCs 7350, 8553.
[103] P. Samarati. 2001. Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering* 13, 6 (Nov. 2001), 1010–1027. DOI: https://doi.org/10.1109/69.971193
[104] Bruce Schneier. 2016. Comparing messaging apps. Retrieved from https://www.schneier.com/blog/archives/2016/06/comparing_messa.html. Accessed: 04-08-2019.
[105] Adam Shostack. 2014. *Threat Modeling: Designing for Security (1 Ed.)*. John Wiley & Sons Ltd, New York.
[106] Signal.org. [n.d.]. Signal github repository. Retrieved from https://github.com/signalapp. Accessed: 03-08-2021.
[107] Signal.org. [n.d.]. Signal support - features. Retrieved from https://support.signal.org/hc/en-us. Accessed: 04-08-2019.
[108] Signal.org. [n.d.]. Signal Terms & Privacy Policy. Retrieved from https://signal.org/legal/. Accessed: 04-08-2019.
[109] Varun Singh. 2018. Explaining the WebRTC Secure Real-Time Transport Protocol (SRTP). Retrieved from https://www.callstats.io/blog/2018/05/16/explaining-webrtc-secure-real-time-transport-protocol-srtp. Accessed: 11-07-2020.
[110] Skype Communications SARL. [n.d.]. What is skype? Retrieved from https://www.skype.com/en/about/. Accessed: 21-05-2019.
[111] Skype Limited. 2010. Skype IT administrators guide - skype for windows version 4.2. Retrieved from https://download.skype.com/share/business/guides/skype-it-administrators-guide.pdf. Accessed: 07-07-2019.
[112] Javier Soltero. 2020. Google meet premium video meetings–free for everyone. Retrieved from https://blog.google/products/meet/bringing-google-meet-to-more-people. Accessed: 13-06-2021.
[113] Stackexchange. [n.d.]. Does tor work with WebRTC? Retrieved from https://tor.stackexchange.com/questions/876/does-tor-work-with-webrtc. Accessed: 10-08-2021.
[114] Statista Research Department. 2021. Whatsapp - statistics and facts. Retrieved from https://www.statista.com/topics/2018/whatsapp/#dossierKeyfigures. Accessed: 2021-11-09.
[115] Peter Stavroulakis and Mark Stamp. 2010. *Handbook of Information and Communication Security*. Springer Berlin Heidelberg, Germany. Retrieved from https://www.springer.com/gp/book/9783642041167.
[116] Shanyu Tang, Qing Chen, Wei Zhang, and Yongfeng Huang. 2016. Universal steganography model for low bit-rate speech codec. *Security and Communication Networks* 9, 8 (2016), 747–754. Retrieved from https://onlinelibrary.wiley.com/doi/full/10.1002/sec.1183.
[117] Editorial Team. 2016. End-to-end whatsapp: An opinionated series on why signal protocol is well-designed. Retrieved from https://www.praetorian.com/blog/whatsapp-end-to-end-encryption-why-signal-protocol-is-well-designed/. Accessed: 01-07-2021.
[118] TeleGeography. 2014. Skype traffic continues to thrive. Retrieved from https://www.telegeography.com/products/commsupdate/articles/2014/01/15/skype-traffic-continues-to-thrive/. Accessed: 22-05-2019.
[119] The Linux Foundation. [n.d.]. Hyperledger indy. Retrieved from https://www.hyperledger.org/projects/hyperledger-indy. Accessed: 24-03-2020.
[120] Tor Project. [n.d.]. Tor project. Retrieved from https://www.torproject.org/. Accessed: 15-03-2020.
[121] Torproject. 2017. UDP over tor. Retrieved from https://gitlab.torproject.org/legacy/trac/-/issues/7830. Accessed: 10-08-2021.

[122] Zeynep Tufekci. [n.d.]. In response to guardian's irresponsible reporting on whatsapp: A plea for responsible and contextualized reporting on user security. Retrieved from http://technosociology.org/?page_id=1687. Accessed: 15-09-2019.
[123] Nate Drake; Brian Turner. 2020. Best video conferencing software in 2020. Techradar. Retrieved from https://www.techradar.com/best/best-video-conferencing-software. Accessed: 26-09-2020.
[124] J. Uberti. 2019. WebRTC IP address handling requirements. IETF. Retrieved from https://tools.ietf.org/html/draft-ietf-rtcweb-ip-handling-12. Accessed: 29-12-2020.
[125] University of Waterloo department Cryptography, Security, and Privacy (CrySP). 2018. Wire. Retrieved from https://crysp.uwaterloo.ca/opinion/wire/. Accessed: 04-08-2020.
[126] Tobias Urban, Dennis Tatang, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. 2019. A study on subject data access in online advertising after the GDPR. In *Proceedings of the Data Privacy Management, Cryptocurrencies and Blockchain Technology (Lecture Notes in Computer Science)*, Cristina Pérez-Solà, Guillermo Navarro-Arribas, Alex Biryukov, and Joaquin Garcia-Alfaro (Eds.). Springer International Publishing, Cham, 61–79. DOI: https://doi.org/10.1007/978-3-030-31500-9_5
[127] Vicki Turk. 2020. Zoom took over the world. This is what will happen next. Wired. Retrieved from https://www.wired.co.uk/article/future-of-zoom. Accessed: 20-10-2020.
[128] Isabel Wagner. 2022. *Auditing Corporate Surveillance Systems: Research Methods for Greater Transparency*. Cambridge University Press.
[129] Isabel Wagner and Eerke Boiten. 2018. Privacy risk assessment: from art to science, by metrics. In *Proceedings of the 13th International DPM Workshop on Data Privacy Management*, Vol. LNCS 11025. Springer, Barcelona, Spain, 225–241. DOI: https://doi.org/10.1007/978-3-030-00305-0_17
[130] Chungyi Wang and Quincy Wu. 2007. Information hiding in real-time VoIP streams. In *Proceedings of the 9th IEEE International Symposium on Multimedia*. IEEE, 255–262.
[131] WhatsApp. 2017. Whatsapp encryption overview. Retrieved from https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf. Accessed: 07-05-2019.
[132] WhatsApp. 2021. Whatsapp privacy policy. Retrieved from https://www.whatsapp.com/legal/updates/privacy-policy/?lang=en. Accessed: 11-01-2021.
[133] WhatsApp FAQ. [n.d.]. How we work with the facebook companies. Retrieved from https://faq.whatsapp.com/general/26000112/?eea=1. Accessed: 15-09-2019.
[134] Whonix.org. [n.d.]. whonix.org. Retrieved from https://www.whonix.org/. Accessed: 12-08-2021.
[135] Wire Swiss GmbH. [n.d.]. About wire. Retrieved from https://wire.com/en/about/. Accessed: 23-07-2020.
[136] Wire Swiss GmbH. [n.d.]. Proteus. Retrieved from https://github.com/wireapp/proteus. Accessed: 13-06-2020.
[137] Wire Swiss GmbH. [n.d.]. Wire audits. Retrieved from https://wire.com/en/security/#audits. Accessed: 23-07-2020.
[138] Wire Swiss GmbH. 2018. Wire privacy whitepaper. Retrieved from https://wire-docs.wire.com/download/WirePrivacyWhitepaper.pdf. Accessed: 19-05-2020.
[139] Wire Swiss GmbH. 2018. Wire security whitepaper. Retrieved from https://wire-docs.wire.com/download/WireSecurityWhitepaper.pdf. Accessed: 19-05-2020.
[140] Gabriel Wood. 2018. phone porting: How hackers can hijack your mobile phone number. NextAdvisor. Retrieved from https://www.nextadvisor.com/phone-porting-how-hackers-can-hijack-your-mobile-phone-number/. Accessed: 03-11-2019.
[141] Eric Yuan. 2020. A message to our users. Retrieved from https://blog.zoom.us/wordpress/2020/04/01/a-message-to-our-users/. Accessed: 11-05-2020.
[142] Eric Yuan. 2020. Zoom's use of facebook's SDK in iOS client. Retrieved from https://blog.zoom.us/wordpress/2020/03/27/zoom-use-of-facebook-sdk-in-ios-client/. Accessed: 11-05-2020.
[143] Ali Zafar. 2018. Countries where whatsapp is banned. Retrieved from https://www.privacyend.com/countries-where-whatsapp-banned/. Accessed: 21-05-2019.
[144] Zoom. 2020. 90-Day security plan: Key updates. Retrieved from https://blog.zoom.us/wp-content/uploads/2020/07/Security-90-day-Plan-Key-Updates.pdf. Accessed: 13-12-2020.
[145] Zoom. 2020. Privacy policy. Retrieved from https://zoom.us/privacy Accessed: 11-05-2020.
[146] Zoom. 2020. Security guide. Retrieved from https://zoom.us/docs/doc/Zoom-Security-White-Paper.pdf. Accessed: 15-07-2020.
[147] Zoom. 2020. Zoom encryption. Retrieved from https://zoom.us/docs/doc/ZoomEncryptionWhitepaper.pdf. Accessed: 11-05-2020.
[148] Zoom. 2021. government requests guide. Retrieved from https://zoom.us/docs/en-us/government-requests-guide.html. Accessed: 18-08-2021.

Received February 2021; revised August 2021; accepted November 2021