ANNÉE 2018

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale MathSTIC

présentée par

Kevin CORRE

préparée à l'unité de recherche IRISA (UMR 6074)
Institut de Recherche en Informatique et Systèmes Aléatoires
ISTIC - UFR en Informatique et Électronique

User Controlled Trust and Security Level of Web Real-Time Communications

Thèse soutenue à Rennes
le 31 mai 2018

devant le jury composé de :

Maryline LAURENT
Professeur, TELECOM SudParis / rapporteur

Yvon KERMARREC
Professeur, IMT Atlantique / rapporteur

Walter RUDAMETKIN
Maître de Conférences, Université de Lille / examinateur

Dominique HAZAEL-MASSIEUX
Ingénieur de Recherche, W3C / examinateur

Vincent FREY
R&D Software Engineer, Orange Labs / examinateur

Olivier BARAIS
Professeur, Université de Rennes 1 / directeur de thèse

Gerson SUNYÉ
Maître de Conférences, Université de Nantes / co-directeur de thèse

“Fide, sed cui vide”.
- Locution latine

“Du père qui m’a donné la vie : la modestie et la virilité,
du moins si je m’en rapporte à la réputation qu’il a laissée
et au souvenir personnel qui m’en reste”.
- Marcus Catilius Severus

Résumé en Français

Contexte

Communication, du latin communicatio [10], est défini par le dictionnaire Oxford comme étant “the imparting or exchanging of information” et “the means of sending or receiving information, such as telephone lines or computers” [11]. Face à une incertitude ou à un risque, un tel échange n’est possible que si une relation de confiance peut être établie entre l’émetteur et le récepteur. Cette relation de confiance doit aussi concerner les moyens utilisés pour communiquer. Les communications écrites ont été développées au travers de plusieurs révolutions : depuis les pictogrammes gravés dans la pierre jusqu’au signaux électroniques répliqués et transmis instantanément. Bien que ces révolutions aient rendu les communications omniprésentes, le besoin fondamental de confiance reste présent.

Depuis l’antiquité, politiciens et généraux ont toujours eu le besoin de communications sécurisées. La stéganographie, le fait de cacher un message à la vue de tous, est l’une des plus anciennes techniques pour sécuriser une communication écrite. Une telle technique peut, par exemple, être réalisée en utilisant une encre invisible ou en utilisant certains bits d’une image numérique pour encoder un message secret. La cryptographie, au contraire, est une technique de transformation d’un message secret, un texte en clair, vers un texte chiffré afin d’en cacher son sens. La scytale, présentée sur la Figure 1, est une technique spartiate de cryptographie par substitution et le premier usage institutionnel de cryptographie militaire. Elle aurait été utilisé en 404 a.v.J.C. [12]. De même, le chiffre de César est l’une des techniques cryptographiques les plus simples. Dans cette technique, les lettres d’un alphabet sont remplacées par d’autres lettres en utilisant un décalage donné. Jules César aurait utilisé une distance de décalage de trois lettres afin d’écrire à ses généraux. En connaissant la valeur de décalage, c’est à dire la clé secrète, le destinataire peut inverser le décalage et récupérer le texte clair.

Figure 1: Une scytale. Une bande de parchemin était enroulée avant d’écrire le message, une lettre par tour. Une fois déroulé, le parchemin était illisible à moins de posséder une scytale du même diamètre.

Les machines à chiffrement électromécanique, telles que la fameuse machine Enigma, ont apporté une impressionnante augmentation des possibilités de chiffrement. Lors de la seconde guerre mondiale, les efforts de cryptanalyse des chercheurs alliés visant à casser les codes électromécaniques menèrent au développement des premiers ordinateurs. Cette percée mena finalement à l’utilisation d’algorithmes cryptographiques fondés sur des problèmes mathématiques complexes. De plus, l’informatique permet le chiffrement de n’importe quel type de données encodées en séquences de bits. Le développement continu de l’informatique et des ’algorithmes de chiffrement publics ont rendu la sécurité des communications accessibles aux petites entreprises et aux individus. Aujourd’hui, les communications sont chiffrées par défaut, et les utilisateurs sont éduqués à prêter attention aux indicateurs de sécurité [13] tels que l’icone HTTPS représentant un cadenas vert 2. Cependant, un tiers de confiance est malgré-tout souvent requis afin d’établir la communication.

Plusieurs définitions de la confiance ont été proposées. Inspirés par McKnight et Chervany [14], Jøsang et Presti définissent la confiance comme: “the extent to which

one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible" [15]. Dans le sens d'une action, nous dirions que la confiance est la décision par un agent intelligent de coopérer face à une incertitude ou un risque au sujet du comportement (sa capacité ou son intention) d'un autre agent. La confiance est une heuristique basée sur l'historique des interactions, la réputation donnée par une communauté, ou les recommandations transmises par d'autres sources de confiance.

La sécurité des informations vise à protéger les informations échangées lors d'une communication, en s'assurant que certaines propriétés de sécurité sont vérifiées:

- Le secret des échanges, la discrétion (en: privacy),
- La confidentialité des messages échangés,
- L'intégrité du contenu,
- La non-répudiation de la communication,
- L'authentification de l'identité des autres participants.

Les systèmes sécurisés sont complexes à construire et peuvent être peu pratiques pour un usage quotidien. Dans cette situation, la confiance peut être une façon plus confortable d'assurer la sécurité des communications.

Le World Wide Web (le Web) est construit sur un modèle client-serveur. Sa sécurité est basée sur une chaîne de certificat ancrée dans le User-Agent, généralement un navigateur Web. Les sites Web sont authentifiés par leurs certificats et les communications entre navigateurs et serveurs sont sécurisées par des protocoles tels que TLS. L'émergence de nouvelles fonctionnalités du Web offre beaucoup plus de cas d'usage via l'utilisation d'une interface de programmation (API) JavaScript. Ces nouvelles interactions sont souvent tripartites entre un utilisateur et deux autres serveurs: un Service Provider (SP) et un Relying Party (RP). C'est par exemple le cas des scénarios de délégation d'authentification. Dans ces situations, chaque partie ne peut pas monitorer les interactions entre les deux autres parties. La sécurité et la confiance de telles communications peuvent donc être plus dures à évaluer. Chaque entité doit alors baser ses décisions sur des critères subjectifs et sa propre évaluation du risque, donc prendre une décision de confiance.

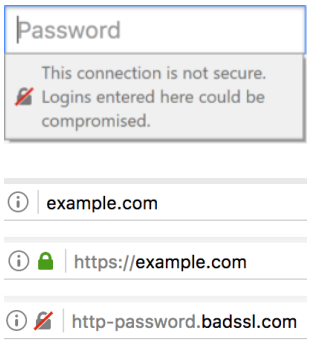


Figure 2: Avertissements de sécurité sur le navigateur Firefox.

.

Les communications en temps réel, sont un autre exemple de ces cas d'usage complexes impliquant plus qu'une simple relation client-serveur. WebRTC est un ensemble d'API Web et de protocoles, spécifiés par le W3C et l'IETF, permettant les appels audio, vidéo et le partage de données en mode pair à pair. Le déploiement de ce nouveau standard pourrait être une opportunité d'ouverture des futurs services Over The Top (OTT) [de l'accès internet]. Puisque WebRTC apporte de nouvelles fonctionnalités à l'écosystème Web, sa sécurité est un facteur critique de son succès. Dans ce but, il est important que la spécification prenne en compte les problèmes rencontrés par les technologies précédentes. C'est notamment vrai concernant la gestion d'identité, en particulier comparée à la qualité de service. Comme les expériences précédentes l'ont prouvé, un modèle d'identité fragile peut avoir un impact important et peut être particulièrement dur à corriger une fois le système déployé.

Défis ouverts

Notre intuition est que les utilisateurs de services de communication temps réel devraient bénéficier de plus d'informations et de contrôle sur la sécurité et le niveau de confiance de leurs sessions. C'est notre objectif global. Pour cela, nous devons construire un modèle

représentant l'architecture de communication, les différents canaux, protocoles et les acteurs impliqués. Ce modèle nous permettra de construire une métrique caractérisant le risque d'utiliser un système de communication, c'est-à-dire son niveau de confiance et de sécurité.

Au vu de cet objectif, les questions suivantes sont ouvertes : Comment modéliser une configuration de sécurité d'une architecture de communication constituée de plusieurs canaux, protocoles et acteurs? Le même modèle peut-il intégrer des informations de confiance et de sécurité en une seule métrique? Quels paramètres de l'architecture de communication doivent être utilisés pour construire cette métrique? Peut-on construire ce modèle dynamiquement?

En supposant l'existence d'un modèle de confiance et de sécurité pour une architecture de communication, ce modèle pourrait être utilisé pour agir sur le système afin d'en augmenter le niveau de confiance et de sécurité. Afin de donner ce genre de contrôle aux utilisateurs, nous devons aussi répondre aux questions suivantes: Les utilisateurs peuvent-ils contrôler le niveau de confiance et de sécurité de leurs sessions de communication? Est-il possible de faire confiance à une architecture de communication dans laquelle certains acteurs ne sont pas de confiance? Les utilisateurs peuvent-ils choisir des acteurs de confiance pour l'établissement de leurs communications?

Afin de permettre aux utilisateurs de contrôler la confiance et la sécurité de leurs communications, la diversité des configurations possibles doit être suffisamment élevée. Cependant, les anciens systèmes de communication téléphoniques obéissent à des standards stricts. Il peut y avoir plusieurs acteurs opérant le réseau, mais ils opèrent sous des configurations fermées et similaires. Il y a aussi de nombreux services fournis OTT, mais ils opèrent en mode silo et en mode boîte noire. La diversité de configuration de ces systèmes est donc elle aussi basse. Au contraire, WebRTC est une technologie Web disruptive. Du fait de sa simplicité , il est attendu que le nombre de services WebRTC explose dans un futur proche. La version finale de WebRTC n'a pas encore été publiée et certaines fonctionnalités restent à implémenter dans les navigateurs. C'est donc le bon moment pour étudier l'architecture de sécurité WebRTC.

Pour ces raisons, nous souhaitons focaliser nos questions de recherche sur la spécification WebRTC. Nous voulons comprendre à quel point les utilisateurs peuvent avoir confiance dans une session WebRTC, y compris si l'un des acteurs n'est pas de confiance. Notre première question de recherche est donc la suivante:

Figure 3: Le logo de WebRTC.

- **RQ1**: Quels sont les risques pour l'utilisateur d'une session WebRTC et quelle abstraction peut-on utiliser pour représenter ces risques aux utilisateurs?

Une façon de s'assurer qu'une session est sécurisée est d'interagir avec des tiers de confiance, soit avant d'établir la session, soit lors de la session pour en augmenter le niveau de confiance et de sécurité. Cela pose les questions suivantes:

- **RQ2**: Peut-on agir sur une session WebRTC afin d'en augmenter le niveau de confiance et de sécurité?

- **RQ3**: Peut-on laisser les utilisateurs choisir des acteurs de confiance pour l'établissement de la communication?

Contributions

Dans cette thèse, je propose trois contributions principales:

Discrétion liée à l'architecture d'identité de WebRTC

Dans notre première contribution, nous étudions l'architecture d'identité WebRTC et plus particulièrement son intégration aux algorithmes de délégation d'authentification existants. Cette intégration n'a pas encore été étudiée jusqu'à présent. Dans cette perspective, nous implémentons les composants de l'architecture d'identité WebRTC ce qui nous permet de montrer que cette architecture n'est pas particulièrement adaptée à une intégration aux protocoles de délégation d'authentification existants tels qu'OpenID Connect. Pour répondre à RQ1, nous montrons ensuite comment la position centrale des fournisseurs d'identité dans l'écosystème du Web est renforcée par leur intégration à l'établissement de session WebRTC, posant ainsi un risque supplémentaire contre la discrétion des utilisateurs. Dans l'écosystème Web, la norme est l'architecture des services en silo dont les utilisateurs sont captifs. C'est en particulier le cas des systèmes de délégation d'authentification, pour lesquels la plupart du temps, il n'est pas possible de choisir son fournisseur d'identité. Afin de répondre à RQ3, nous réalisons une étude afin de déterminer pour quelles raisons les utilisateurs ne peuvent pas choisir leur fournisseur d'identité sur Web. Notre étude montre que bien que ce choix soit possible en théorie, l'absence d'implémentation de certains standards par les sites webs et les fournisseurs d'identité empêche ce choix en pratique.

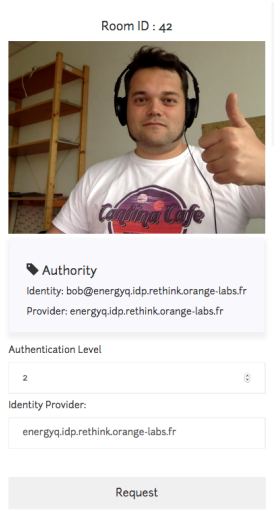


Figure 4: Notre interface de négotiation d'identité sur un service WebRTC.

Controller les paramètres d'identité de WebRTC

Dans notre seconde contribution, nous cherchons à donner plus de contrôle à l'utilisateur. Pour ce faire et en réponse à RQ2, nous proposons une extension de la spécification WebRTC afin de permettre la négociation des paramètres d'identité. Un prototype d'implémentation est proposé afin de valider notre proposition (voir Figure 4). Cette implémentation révèle certaines limites dues à l'API d'identité WebRTC empêchant notamment d'obtenir un retour sur le niveau d'authentification de l'autre utilisateur ainsi que l'impossibilité de changer de fournisseur d'identité en cours de session. Nous proposons ensuite une API Web permettant aux utilisateurs de choisir leur fournisseur d'identité lors d'une authentification sur un site tiers via une interface de sélection d'identité contrôlée par le navigateur. Répondant à RQ3, notre API repose sur une réutilisation de l'architecture d'identité WebRTC dans un scénario client-serveur. Nous présentons une implémentation de notre solution, basée sur une extension du navigateur Firefox, afin d'en démontrer l'utilisabilité (voir Figure 5). Nos résultats montrent qu'à long terme, l'adoption de cette API pourrait réduire la charge d'implémentation pour les développeurs de sites Web et permettre aux utilisateurs de préserver leur discrétion en choisissant des fournisseurs d'identité de confiance.

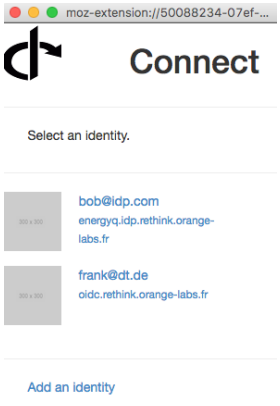


Figure 5: L'interface de sélection d'identité de WebConnect.

Un modèle de confiance et de sécurité pour WebRTC

Enfin dans notre troisième contribution, nous répondons à RQ1 en proposant un modèle de confiance et de sécurité d'une session WebRTC. Ce modèle intègre en une seule métrique les paramètres de sécurité utilisés lors de l'établissement de la session, les paramètres d'encryption des flux média, et les paramètres de confiance de l'utilisateur dans les acteurs de la session WebRTC. L'objectif de notre modèle est dans un premier temps de permettre aux utilisateurs non-experts de mieux comprendre la sécurité de leurs sessions WebRTC. Afin de valider notre approche, nous réalisons une expérimentation préliminaire évaluant la compréhension de notre modèle par des utilisateurs non-experts en sécurité.

Publications

Mes travaux ont été le sujet de plusieurs publications et contributions, en voici la liste:

Publications Scientifiques

[1] Kevin Corre, Simon Bécot, Olivier Barais, and Gerson Sunyé. "A WebRTC Extension to Allow Identity Negotiation at Runtime". *Web Engineering - 17th International Conference, ICWE 2017, Rome, Italy, June 5-8, 2017, Proceedings*. Ed. by Jordi Cabot, Roberto De Virgilio, and Riccardo Torlone. Vol. 10360. Lecture Notes in Computer Science. Springer, 2017, pp. 412–419

[2] Kevin Corre, Olivier Barais, Gerson Sunyé, Vincent Frey, and Jean-Michel Crom. "Why can't users choose their identity providers on the web?" *PoPETs* 2017.3 (2017), pp. 72–86

[9] Ibrahim Tariq Javed, Rebecca Copeland, Noël Crespi, Marc Emmelmann, Ancuta Corici, Ahmed Bouabdallah, Tuo Zhang, Saad El Jaouhari, Felix Beierle, Sebastian Göndör, Axel Küpper, Kevin Corre, Jean-Michel Crom, Frank Oberle, Ingo Friese, Ana Caldeira, Gil Dias, Nuno Santos, Ricardo Chaves, and Ricardo Lopes Pereira. "Cross-domain identity and discovery framework for web calling services". *Annales des Télécommunications* 72.7-8 (2017), pp. 459–468

Internet Drafts

[3] Rebecca Copeland, Kevin Corre, Ingo Friese, and Saad El Jaouhari. *Requirements for Trust and Privacy in WebRTC Peer-to-peer Authentication*. Internet-Draft draft-copeland-rtcweb-p2p-idp-auth-00. IETF Secretariat, Sept. 2016

A note on RFC Requests For Comments are documents published by the IETF and describing technical aspects of the internet. RFC are first proposed as *draft* with an expiration date of 6 months, which can be extended by submitting new version of the draft. If an interest emerge, a working group may form. This working group will develop the draft further until a request to publish it as a RFC is submitted. Not all RFCs are however submitted on the standard track. Their status may alternatively be *informational*, *experimental*, *best current practice*, *historic*, or *unknown*.

Brevets

[4] Kevin Corre and Vincent Frey. "Method of managing the authentication of a client in a computing system". WO2017006013 A1 Patent App. PCT/FR2016/051,601. 2016

Rapports Techniques

[5] Rebecca Copeland, Ahmed Bouabdallah, Ibrahim Javed, Eric Paillet, Simon Bécot, Ewa Janczukowicz, Kevin Corre, Jean-Michel Crom, Paulo Chainho, Felix Beierle, Sebastian Göndör, Frédéric Luart, Adel Al-Hezmi, Andreea Ancuta Corici, Marc Emmelmann, Ricardo Lopes Pereira, Ricardo Chaves, and Nuno Santos. *Framework Architecture Definition*. Deliverable D2.1. reThink Project, 2015

[6] Jean-Michel Crom, Kevin Corre, Simon Bécot, Ingo Friese, Felix Beierle, Sebastian Göndör, Ahmed Bouabdallah, Marc Emmelmann, Andrea Ancuta Corici, Ricardo Chaves, and Ricardo Pereira. *Management and Security features specifications*. Deliverable D4.1. reThink Project, 2015

[7] Jean-Michel Crom, Kevin Corre, Simon Bécot, Felix Beierle, Sebastian Göndör, Ahmed Bouabdallah, Saad El Jaouhari, Rebecca Copeland, Marc Emmelmann, Ricardo Chaves Andrea Ancuta-Corici Robert Ende, and Ricardo Pereira. *Implementation of Governance and identity management components for phase 1.* Deliverable D4.2. reThink Project, 2016

[8] Jean-Michel Crom, Kevin Corre, Ingo Friese, Felix Beierle, Sebastian Göndör, Ahmed Bouabdallah, Hao Jiang, Rebecca Copeland, Ibrahim Tariq Javed, Marc Emmelmann, Andrea Ancuta Corici, Robert Ende, Ricardo Chaves, Nuno Santos, and Ricardo Pereira. *Implementation of Governance and identity management components for phase 2.* Deliverable D4.3. reThink Project, 2017

Cette thèse présente le résultat de notre travail de recherche conduit lors des trois dernières années. Il est le résultat d'une collaboration entre l'équipe Identity and Trust Architecture d'Orange Labs et l'équipe DiverSE de l'INRIA. Nous avons aussi contribué au projet H2020 reThink, en particulier à l'architecture d'identité et de sécurité.

Abstract

Nowadays, communications transiting through the Internet are encrypted by default, often end-to-end. Secure communication is a commodity accessible to everyone rather than restricted to powerful organisations. Users are being educated to the security risks faced on the internet and to pay attention to security indications. But this does not solve every security issues: a trusted third party is still required in order to setup the communication. Security on the Web is based on a certificate chain anchored into the web browser. The emerging web functionalities offer a lot more possible use-cases than the usual client-server communication. Real-time media and data communication is one of these complex use-cases which involves peer-to-peer and full-duplex client-server communications. On one hand legacy inter-operable communication systems suffer from issues regarding the trustworthiness of their incoming call. On the other hand over-the-top communication networks are all set in a silo model: users are de-facto captive of these services.

WebRTC is a set of standard web API and protocols, which supports peer-to-peer audio-video calling and data sharing. It is envisioned, given the simplicity to deploy a WebRTC services, that the number of WebRTC enabled websites could skyrocket in the near future. To succeed, WebRTC should improve from the issues encountered by previous technologies. A weak identity model may have an important impact later on and is particularly hard to fix once the system is deployed. Considering the various use cases and the possible number of services and other actors, the complexity of a communication setup could be really difficult to assess by non-expert users.

Our intuition is that users should be given more informations and control on the security and trust level of their communications. We want to build a model that could represent the communication setup, the different channels, protocols, and actors in operations. This model would allow us to act on the system in order to raise the trust and security level. At the moment, WebRTC's final version of the specification has not yet been published, and some functionalities are yet to be implemented in Web Browsers. It may be the right time to challenge its security architecture by addressing the following research questions:

- **RQ1**: What are the risks for the user of a WebRTC session and which abstractions can we use to show these risks to the user?
- **RQ2**: Can we act on a WebRTC session to raise the trust and security level?
- **RQ3**: Can users choose actors they trust to participate in the communication setup?

In this thesis, we propose three main contributions:

In our first contribution we study the WebRTC identity architecture and more particularly its integration with existing authentication delegation protocols. This integration has not been studied yet. To fill this gap, we implement components of the WebRTC identity architecture and comment on the issues encountered in the process. In order to answer RQ1, we then study this specification from a privacy perspective an identify new privacy considerations related to the central position of identity provider. In the Web, the norm is the silo architecture of which users are captive. This is even more true of authentication delegation systems where most of the time it is not possible to freely choose an identity provider. In order to answer RQ3, we conduct a survey on the top 500 websites according to Alexa.com to identify the reasons why can't users choose their identity provider. Our results show that while the choice of an identity provider is possible in theory, the lack of implementation of existing standards by websites and identity providers prevent users to make this choice.

In our second contribution, we aim at giving more control to users. To this end and in order to answer RQ2, we extend the WebRTC specification to allow identity parameters negotiation. We present a prototype

implementation of our proposition to validate it. It reveals some limits due to the WebRTC API, in particular preventing to get feedback on the other peer's authentication strength. We then propose a web API allowing users to choose their identity provider in order to authenticate on a third-party website, answering RQ2. Our API reuse components of the WebRTC identity architecture in a client-server authentication scenario. Again, we validate our proposition by presenting a prototype implementation of our API based on a Firefox extension.

Finally, in our third contribution, we look back on RQ1 and propose a trust and security model of a WebRTC session. Our proposed model integrates in a single metric the security parameters used in the session establishment, the encryption parameters for the media streams, and trust in actors of the communication setup as defined by the user. Our model objective is to help non-expert users to better understand the security of their WebRTC session. To validate our approach, we conduct a preliminary study on the comprehension of our model by non-expert users. This study is based on a web survey offering users to interact with a dynamic implementation of our model.

Contents

Résumé en Français **i**

Abstract **vii**

Introduction **1**

I Context **9**

1 WebRTC Trust and Security Architecture **11**
1.1 WebRTC Overview . . . 11
1.2 Security on the Web . . . 14
1.2.1 The Trusted Computing Base . . . 15
1.2.2 The Same Origin Policy . . . 15
1.2.3 The HyperText Transfer Protocol Secure . . . 16
1.2.4 Public Key Cryptography . . . 17
1.3 WebRTC Security . . . 19
1.3.1 Confidentiality and Integrity of the Media Path . . . 19
1.3.2 Availability of the Communication . . . 20
1.3.3 User Authenticity . . . 20
1.3.4 WebRTC Identity Path . . . 22
1.3.5 Considered Protocols for WebRTC Peer Authentication . . . 24
1.3.6 Alternative Key Management Protocols . . . 28
1.4 Trust . . . 29
1.4.1 Introduction on Trust . . . 29
1.4.2 The WebRTC Trust Model . . . 30
1.5 Privacy of the Call-Session . . . 31
1.5.1 Attack and Threat Mitigation . . . 31
1.5.2 Regulations . . . 33
1.5.3 Privacy Considerations for WebRTC . . . 34
1.5.4 Tor: Onion Routing . . . 34
1.6 Signalling Architectures . . . 34
1.6.1 Voice over LTE and WebRTC Interconnectivity . . . 35
1.6.2 Matrix . . . 37
1.6.3 reThink . . . 38
1.6.4 Distributed Signalling Architectures . . . 39
1.7 Summary . . . 41

2 State of the Art **43**
2.1 VoIP Security Research - 2012 . . . 43
2.1.1 Threats Classification and Methodology . . . 43
2.1.2 Keromytis Survey Summary . . . 44
2.2 VoIP and WebRTC Security Research - 2012+ . . . 50
2.2.1 Methodology . . . 50
2.2.2 Observations on VoIP Security Research since 2012 . . . 50
2.2.3 Survey of WebRTC Security Research . . . 51
2.2.4 Observations . . . 56
2.3 Summary . . . 59

II Contributions **61**

Foreword on Methodology **63**

3 Privacy Implications of the WebRTC Identity Architecture **65**
3.1 WebRTC Identity Architecture Implementation . . . 65
3.1.1 Local Authentication Implementation . . . 66
3.1.2 IdP Proxy with OpenID Connect . . . 67
3.1.3 Observations . . . 70
3.2 RQ1.1 Additional Privacy Considerations . . . 72
3.2.1 Audience Issue . . . 73
3.2.2 IdP in a Central Position . . . 74
3.3 Why Can't Users Choose their Identity Providers on the Web? . . . 75
3.3.1 The Study: OAuth Request Collection . . . 75
3.3.2 RQ3.1: Do RP require specialised API? . . . 76
3.3.3 RQ3.2: Is dynamic discovery and registration commonly available for RP? . . . 79
3.3.4 RQ3.3: Do RP requires a trust relationship with the supported IdP? . . . 80
3.3.5 Developer Survey . . . 81
3.4 Summary . . . 84

4 Controlling the WebRTC Identity Parameters **85**
4.1 An SDP Extension to Allow Identity Negotiation . . . 86
4.1.1 Recommendation Sources . . . 86
4.1.2 SDP Extension . . . 87
4.1.3 Validation on the current specification . . . 88
4.2 WebConnect . . . 92
4.2.1 Implementation . . . 93
4.2.2 Analysis . . . 97
4.2.3 Validation . . . 99
4.3 Summary . . . 100

5 WebRTC Trust and Security Model **101**
5.1 Methodology . . . 102
5.2 Building the WebRTC Trust and Security Model . . . 105
5.2.1 Session Confidentiality . . . 105
5.2.2 Signalling Path Security . . . 105
5.2.3 Identity Path Security . . . 106
5.2.4 Media Path Confidentiality . . . 107
5.2.5 Overall Trust and Security Tree, Instantiation and Computational Models . . . 107
5.3 Validation . . . 110
5.3.1 WebRTC Trust and Security Model Survey . . . 110

5.3.2 Discussions 117
5.4 Summary 119

III Conclusion and Perspectives 121

6 Conclusion 123

7 Perspectives 125
7.1 On the WebRTC Trust and Security Model 125
7.2 On the IdP Proxy Interface 126
7.3 On WebConnect and the WebPayment Working Group 128

Author's Publications 130

References 131

Glossary 140

List of Figures 141

List of Tables 142

Introduction

Context

Communications, deriving from the Latin *communicatio* [10], is defined by the Oxford dictionary as "the imparting or exchanging of information" and "the means of sending or receiving information, such as telephone lines or computers" [11]. In case of uncertainty and risk, such exchange is made possible if a trust relation can be established between the senders and receivers. This trust relation must also cover the channels used to communicate. Written communications were developed through several revolutions: from immobile chiselled pictograms to electronic signals instantly replicated and transmitted. While these revolutions made communications more and more pervasive, the fundamental issue of trust remains. Politicians and military commanders may always have had a need for secure communications, and for long the concept of secure communication was equivalent to the confidentiality of a message. Steganography, the fact of hiding a message in plain sight, was the earliest technique to secure a written communication. It can be achieved by using an invisible ink or by inserting a message inside another document, for instance, using bits of a numerical image to encode a secret text. Cryptography, on the other hand, is a technique transforming a secret message, called a plaintext, into an encrypted cyphertext in order to hide its meaning. The first known evidence of cryptography is an engraved pottery from 1900bc. The Spartan scytale, shown in Figure 6, was the first known institutional military cypher, reported as being used in 404 B.C. [12].

Caesar's cypher may be one of the simplest cryptographic technique. In this technique, letters are replaced by another letter at a given distance in the alphabet. Julius Caesar was reported as using this cypher with a shift of three letter to write secret messages to his generals. Knowing the shift value, *i. e.* the secret key, a recipient of an encrypted message could reverse the shift to uncover the plaintext message. Alberti invented around 1467 a polyalphabetic substitution cypher where the key is modified at random through the text. Though later works derive from Alberti's invention, such cryptography technique was already known by the 8th century's mathematician Al-Kindi. Substitution cyphers are vulnerable to letter frequency attack. Indeed, a cryptanalyst can guess the key of a simple substitution cypher by comparing the letter frequencies of the cyphertext to the letter frequencies of another plaintext known to be written in the same language. Polyalphabetical substitutions try to alter the frequency of letters by continuously changing the substitution alphabet. These are however still vulnerable to more complex frequency analysis. The development of substitution cyphers was thus a race between cryptographers inventing new keying mechanisms and cryptanalysts.

Figure 6: A band of parchment was first wrapped around the scytale, then the message was written one letter per convolution. Once unfolded, the message would be unreadable except for the receiver which would have another scytale of the same diameter.

Rotor machines, such as the famous Enigma machine, brought a large increase in the number of possible substitutions. Cryptanalysis efforts by allied researchers during the second world war resulted in the development of the first programmable computer. This breakthrough also allowed the use of much more complex cyphers based on complex mathematical problems. Furthermore, computers allow the encryption of any kind of data encoded as a sequence of bits. The development of computer and public

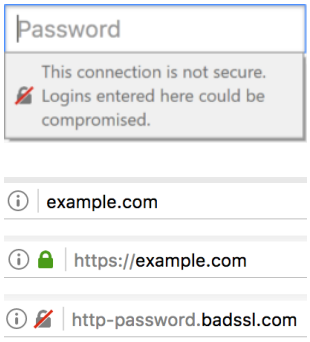


Figure 7: By displaying warning on insecure HTTP webpage, the Firefox web browser is educating its users to the faced risk, and at the same time pushing web developers to implement security for HTTP.

cryptographic algorithms made security accessible to small companies and individuals. Everyday communications are now encrypted by default, and users are being educated to pay attention to security indications [13] such as the one presented in Figure 7. This, however, does not solve every issue: a trusted third party is still often required to set up the communication.

Several definitions of trust have been proposed. Inspired by McKnight and Chervany [14], Jøsang and Presti define trust as: "the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible" [15]. In the sense of an action, we would say that trust is the decision of intelligent agents to cooperate in the face of risk and uncertainty about the behaviour (capability or intention) of other agents. This heuristic is often based on prior interaction history, reputation from a community or recommendations from other trusted sources.

Information security, on the other hand, aims at protecting information exchanged during a communication, by ensuring that certain security properties are met:

- The secrecy of the exchange, *i. e.* the *privacy*.
- The *confidentiality* of the exchanged message.
- The *integrity* of the content.
- The *non-repudiation* of the communication.
- The *authentication* of the other peer's identity.

Fully secure systems are difficult to build and may be impractical for everyday usage. In this situation, trust may be a more relaxed way of ensuring the security of the communication.

The World Wide Web (in short: the Web) is built on a client-server model. Its security is based on a certificate chain anchored into the User-Agent (UA), usually a web browser, acting as the Trusted Computing Base (TCB). Websites are authenticated by their certificates and communications between the UA and the server are secured by protocols such as Transport Layer Security (TLS). The emerging web functionalities offer a lot more possible use-cases through JavaScript web Application Programming Interface (API) and HyperText Transfer Protocol (HTTP) [16] based protocols. These new relations are often tripartite between a user and two other servers, a service provider and a Relying Party (RP). This is, for instance, the case in authentication delegation scenarios. In these tripartite situations, each party cannot fully monitor interactions between the others parties. The security and trustworthiness of such communication may be more difficult to assess. As a result, it is up to each entity to take a decision based on subjective criteria and its own evaluation of the faced risk, *i. e.* to take a trust decision.

Real-time media and data communication is another complex use-case involving more than a single client-server relation. On one hand, legacy inter-operable communication systems, operated by telephony companies and Internet service providers (often referred as Telco) have been cluttered by issue regarding the trustworthiness of their incoming call. These legacy services suffer from a weak trust and identity model which is subject to abuse. The openness of trust-circle networks and the availability of computation power allow malicious actors to launch thousands of robocalls at once[1]. This factor could be one of the reasons behind the drop of value in telephony services [19], a major asset of Telco, benefiting to Over The Top (OTT)[2] services. On the other hand, OTT are rapidly gaining market share over Telco but are all set in silo model. The silo model implies that while these services benefit from a controlled trust model, they cannot communicate with each other and are the theatre of an intensive battle to become, or

1: In 2014, the United States' Federal Trade Commission has received over 22 million complaints about illegal and unwanted phone calls [17]. Even though the practice of phone spam is illegal in the European Union without an opt-in from the recipient, in France 1,6 million voice and SMS spams were reported in 2016 to the 33700, the national spam reporting number [18].

2: OTT services are provided on top of existing internet service providers networks

stay, the hegemonic service[3]. As it is difficult to change network without losing the ability to reach their contacts, users are de-facto captive of these services.

WebRTC is a set of web API and protocols, specified by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF), which supports Peer-to-Peer (P2P) audio-video calling and data sharing. As WebRTC brings useful new functionalities to the web ecosystem, its deployment could be an opportunity to open the future OTT services. For the WebRTC specification to succeed, it should learn from the issues encountered by previous technologies. As past experiences have proved [22], a weak identity model may have an important impact later on and is particularly hard to fix once the system is deployed.

A Motivating Scenario

In a conference call, several users connect to a single room to organise a meeting. Users share their audio and video, they may also share some documents or their screen [23]. Several communication services are already offering this kind of service, some using WebRTC. However, a common issue appears when participants connect from different company, for instance during a collaborative project. While the service is often integrated with one company's Identity Provider (IdP), participants from other companies may not authenticate with it and may have to fallback to a declarative or self-asserted identity. Because the subject of their conversation is sensible, participants want to ensure that no one is maliciously listening, including the communication service itself which may come from a third party company. Currently, implementation decisions made by the Communication Service (CS) website drastically limit the user's choice. And at the same time communication services are missing the capability to interface with any IdP.

Going further, supposing every participant authenticated to each other, one of them may want to raise the security level of the session. Each participant would then be required to authenticate with a higher strength, and the session security would be renegotiated with stronger parameters. For this to be possible, it is required to have a model of the trust and security level of the session and a mechanism to negotiate it. In the long-term, the future telephony will move to a fully IP-based network, interconnected with the Web. In this scenario, inter-operability, *i. e.* the ability for a user of one CS to call a user from another CS, could be imposed by regulations. Some projects are already writing work in progress inter-operable CS architectures [24, 25, 26]. Evaluating the security of a call session is necessary for users to make an informed trust decision on whether or not to accept or pass a call. However, in these inter-operable architectures, the communication stack is drastically more complex than the usual client-server model.

Open Challenges

Our intuition is that users should be given more information and control over the security and trust level of their communications. This is our global objective. For this to be possible, we need to build a model that could represent the communication setup, the different channels, protocols, and the actors in operation. This model would allow us to forge a single metric characterising the risk of using the communication system, *i. e.* the trust and security level. It would also be a useful tool to model an instance of a communication setup specification and discuss it with expert.

To inform the user of the security of its communication session, it is necessary to answer the following questions:

- How to model the security of a communication setup made with several distinct channels, protocols, and actors?

- Can the same model integrate trust and security information into a single meaningful metric?
- Which parameters of the communication setup can be used to forge this metric?
- Can we build this model at runtime?

Supposing we dispose of a trust and security model for the communication setup, we could leverage this model to raise the trust and security level of the session. This process could either be manually or automatically controlled by the user. To give this kind of control to the users, we would also have to answer the following questions:

- Can users control and eventually raise the trust and security level of their communication sessions?
- Is it possible to trust a communication setup in which some of the actors are not trusted?
- Can users choose trusted actors to participate in their communication setup?

As we mentionned, legacy communication services obey strict standards allowing interoperability between actors. There are also multiple OTT communication services, but since they are set in silo and often operate closed-source softwares, we consider them as blackboxes. WebRTC, however, is a disruptive technology for web real-time communications. It is envisioned, given the simplicity to deploy a WebRTC service, that the number of WebRTC enabled websites could skyrocket in the near future. Considering the proposed use cases, the number of WebRTC services, and the other actors: the complexity of communication setup, from a trust and security perspective, could be difficult to assess for non-expert users. At the moment, WebRTC's final version of the specification has not yet been published, and some functionalities are yet to be implemented in web Browsers[4]. It may be the right time to challenge its security architecture.

4: The state of WebRTC implementations on web browsers can be followed on Mozilla's developer website: `https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection`.

For these reasons, we want to address our research questions with respect to the WebRTC specifications. We want to understand to which extent users could trust a WebRTC session, even if some parties are untrusted. Our first research question is thus the following:

- **RQ1**: What are the risks for the user of a WebRTC session and which abstractions can we use to show these risks to the user?

One way to ensure that the session is secured is to interact with trusted parties, either before the session is established, or during the session to raise the trust level. This opens the two following questions:

- **RQ2**: Can we act on a WebRTC session to raise the trust and security level?
- **RQ3**: Can we let users chose actors they trust to participate in the communication setup?

Contributions

In this thesis, I propose three main contributions:

In our first contribution, we study the WebRTC identity architecture and more particularly its integration with existing authentication delegation protocols. This integration has not been studied yet. To fill this gap, we implement components of the WebRTC identity architecture and comment on the issues encountered in the process. In order to answer RQ1, we then study this specification from a privacy perspective

and identify new privacy considerations related to the central position of IdP. In the Web, the norm is the siloed architecture of which users are captive. That is all the more the case for authentication delegation systems where most of the time it is not possible to freely choose an IdP. In order to answer RQ3, we conduct a survey of the top 500 websites according to Alexa[5] to identify the reasons why users cannot choose their IdP.

5: https://alexa.com

In our second contribution, we aim at giving more control to users. To this end and in order to answer RQ2, we extend the WebRTC specification to allow identity parameters negotiation. We present a prototype implementation of our proposition to validate our proposition. We then propose a web API allowing users to choose their IdP in order to authenticate on a third-party website. Our API reuse components of the WebRTC identity architecture in a client-server authentication scenario. Again, we validate our proposition by presenting a prototype implementation of our API based on a Firefox extension.

Finally, in our third contribution, we look back on RQ1 and propose a trust and security model of a WebRTC session. Our proposed model integrates into a unique metric the security parameters used in the session establishment, the encryption parameters for the media streams, and trust in actors of the communication setup as defined by the user. Our model's objective is to help non-expert users to better understand the security of their WebRTC session. To validate our approach, we conduct a preliminary study on the comprehension of our model by non-expert users. This study is based on a web survey offering users to interact with a dynamic implementation of our model.

Figure 8 represents how our contributions articulate and how they relate to our research questions. In our first contribution we study the WebRTC identity architecture. We then experiment on how to control this identity architecture to increase trust and security. Finally and based on our knowledge, we close the loop and propose a model of the trust and security of the WebRTC security architecture.

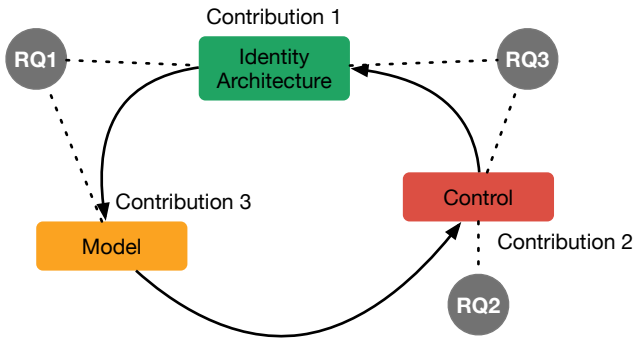


Figure 8: Overview of our Contributions

Organisation

The document is organised in three main parts. In the first part, we expose the context of our research. In Chapter 1, we detail the technical specifications used in our work such as OAuth 2 and WebRTC. We then present the state of the art to which our work contributes in Chapter 2. In the second part of this thesis, we present our contributions. In Chapter 3 we study WebRTC privacy issues based on actual implementations of the WebRTC identity architecture. In Chapter 4 we propose to give users more control over their WebRTC identity parameters. Chapter 5 presents a trust and security model of a WebRTC session intended to help advanced users in understanding the security of their WebRTC session. Finally, the third part of this thesis proposes a summary of our contributions in Chapter 6 and discusses our perspectives in Chapter 7.

Part I: Context

Chapter 1 *WebRTC Trust and Security Architecture* is an introduction to the techniques, algorithms, and protocols securing WebRTC and forming the context of our thesis. In this chapter, we introduce the WebRTC architecture and specifications, present an overview of the concepts accomplishing practical security on the Web, and focus on the protocols used to secure a WebRTC communication. We also introduce the concept of trust and we explain privacy in the context of the Web and in particular the privacy consideration of the WebRTC specification. Finally, we present some signalling architecture that could be deployed by WebRTC services.

Chapter 2 *State of the Art* presents the state of the art on Voice over IP (VoIP) security research. We review the results of a comprehensive survey of 245 articles on VoIP security and published in 2012. As the first draft of the WebRTC specification was published the same year, this survey is a solid starting point to study the field of WebRTC security. We then present our own survey of VoIP and WebRTC security research collecting papers published between 2012 and 2017. Based on our findings, we review collected papers dealing specifically with WebRTC.

Part II: Contributions

Chapter 3 *Privacy Implications of the WebRTC Identity Architecture* studies the WebRTC identity architecture from a privacy point-of-view. The specification lacks support and, to the best of our knowledge, there is no public implementation. We first describe our implementation of the WebRTC identity architecture and its integration with existing authentication delegation protocols. We then comment on encountered issues and detail additional privacy considerations. Finally, we report on our study on why users cannot choose their IdP on the Web.

Chapter 4 *Controlling the WebRTC Identity Parameters* proposes to give users more control over WebRTC identity parameters. Firstly, we define a Session Description Protocol (SDP) extension to negotiate the other peer's IdP and authentication strength during the call setup and present our implementation. Secondly, this chapter presents WebConnect, a prototype for an Identity Metasystem API based on the WebRTC identity specification which allows users to preserve their privacy by selecting trusted IdP. The WebRTC identity specification offers some interesting concepts limited to the scope of user-to-user authentication, WebConnect extends them to the use case of user-to-server authentication.

Chapter 5 *WebRTC Trust and Security Model* presents a trust and security model intended to help advanced users in understanding the security of their WebRTC session. We first present our methodology to build our model and then details our actual WebRTC trust and security model. The model evaluates security with regards to the risk presented on the confidentiality and integrity of the communication and shows which trust relations must be valid for the security level to be trusted too. In order to validate our approach, we present a preliminary study on the understanding of our model by advanced users based on a survey and a dynamic implementation of our model.

Part III: Conclusion and Perspectives

Chapter 6 *Conclusion* presents the conclusions of our work.

Chapter 7 *Perspectives* discusses possible future research directions and perspectives. More particularly, we look at continuing work on our WebRTC trust and security model, improving the WebRTC identity architecture and integrating it with other authentication protocols, and comparing the WebRTC identity architecture to the current work of the W3C WebPayment working group.

List of Publications

Scientific Publications

[1] Kevin Corre, Simon Bécot, Olivier Barais, and Gerson Sunyé. "A WebRTC Extension to Allow Identity Negotiation at Runtime". *Web Engineering - 17th International Conference, ICWE 2017, Rome, Italy, June 5-8, 2017, Proceedings.* Ed. by Jordi Cabot, Roberto De Virgilio, and Riccardo Torlone. Vol. 10360. Lecture Notes in Computer Science. Springer, 2017, pp. 412–419

[2] Kevin Corre, Olivier Barais, Gerson Sunyé, Vincent Frey, and Jean-Michel Crom. "Why can't users choose their identity providers on the web?" *PoPETs* 2017.3 (2017), pp. 72–86

[9] Ibrahim Tariq Javed, Rebecca Copeland, Noël Crespi, Marc Emmelmann, Ancuta Corici, Ahmed Bouabdallah, Tuo Zhang, Saad El Jaouhari, Felix Beierle, Sebastian Göndör, Axel Küpper, Kevin Corre, Jean-Michel Crom, Frank Oberle, Ingo Friese, Ana Caldeira, Gil Dias, Nuno Santos, Ricardo Chaves, and Ricardo Lopes Pereira. "Cross-domain identity and discovery framework for web calling services". *Annales des Télécommunications* 72.7-8 (2017), pp. 459–468

Internet Drafts

[3] Rebecca Copeland, Kevin Corre, Ingo Friese, and Saad El Jaouhari. *Requirements for Trust and Privacy in WebRTC Peer-to-peer Authentication.* Internet-Draft draft-copeland-rtcweb-p2p-idp-auth-00. IETF Secretariat, Sept. 2016

A note on Request For Comments (RFC) : RFC are documents published by the IETF and describing technical aspects of the internet. RFC are first proposed as *draft* with an expiration date of 6 months, which can be extended by submitting new version of the draft. If an interest emerges, a working group may form. This working group will develop the draft further until a request to publish it as a RFC is submitted. Not all RFC are however submitted on the standard track. Their status may alternatively be *informational*, *experimental*, *best current practice*, *historic*, or *unknown*.

Patents

[4] Kevin Corre and Vincent Frey. "Method of managing the authentication of a client in a computing system". WO2017006013 A1 Patent App. PCT/FR2016/051,601. 2016

Technical Reports

[5] Rebecca Copeland, Ahmed Bouabdallah, Ibrahim Javed, Eric Paillet, Simon Bécot, Ewa Janczukowicz, Kevin Corre, Jean-Michel Crom, Paulo Chainho, Felix Beierle, Sebastian Göndör, Frédéric Luart, Adel Al-Hezmi, Andreea Ancuta Corici, Marc Emmelmann, Ricardo Lopes Pereira, Ricardo Chaves, and Nuno Santos. *Framework Architecture Definition.* Deliverable D2.1. reThink Project, 2015

[6] Jean-Michel Crom, Kevin Corre, Simon Bécot, Ingo Friese, Felix Beierle, Sebastian Göndör, Ahmed Bouabdallah, Marc Emmelmann, Andrea Ancuta Corici, Ricardo Chaves, and Ricardo Pereira. *Management and Security features specifications.* Deliverable D4.1. reThink Project, 2015

[7] Jean-Michel Crom, Kevin Corre, Simon Bécot, Felix Beierle, Sebastian Göndör, Ahmed Bouabdallah, Saad El Jaouhari, Rebecca Copeland, Marc Emmelmann, Ricardo Chaves Andrea Ancuta-Corici Robert Ende, and Ricardo Pereira. *Implementation of Governance and identity management components for phase 1.* Deliverable D4.2. reThink Project, 2016

[8] Jean-Michel Crom, Kevin Corre, Ingo Friese, Felix Beierle, Sebastian Göndör, Ahmed Bouabdallah, Hao Jiang, Rebecca Copeland, Ibrahim Tariq Javed, Marc Emmelmann, Andrea Ancuta Corici, Robert Ende, Ricardo Chaves, Nuno Santos, and Ricardo Pereira. *Implementation of Governance and identity management components for phase 2.* Deliverable D4.3. reThink Project, 2017

This thesis presents the results of my three years research work. It is the result of a collaboration between the Identity and Trust Architecture research project from Orange Labs and the INRIA DiverSE Team. We also contributed to the H2020 reThink project for its identity and security architecture.

Part I

Context

Chapter 1

WebRTC Trust and Security Architecture

Information security aims at protecting the confidentiality, integrity, and availability of an information. These objectives are referred to as the CIA triad and are often associated with additional security objectives such as non-repudiation, authenticity, and privacy. Our research questions are centred on the idea of building a security and trust model for WebRTC. This chapter is an introduction to the techniques, algorithms, and protocols securing WebRTC and forming the context of our thesis. Firstly we introduce the WebRTC architecture and specifications in Section 1.1. Then we present an overview of the concepts accomplishing practical security on the Web in Section 1.2 and later focus on the protocols used to secure a WebRTC communication in Section 1.3. We also introduce the concept of trust and present the WebRTC trust model in Section 1.4. Even if the content of a communication is sufficiently secure, the attacker's knowledge that a communication happened may also be a risk for the user. In Section 1.5, we explain privacy in the context of the Web and in particular the privacy considerations of the WebRTC specification. Finally, we present some signalling architecture that could be deployed by WebRTC services in Section 1.6. WebRTC does not specify a signalling architecture, but these have an important impact on the underlying trust and security properties.

1.1 WebRTC Overview

WebRTC is a standardisation effort for interoperable real-time communication in the Web, in line with the specifications of HTML5 technologies. This set of specifications aims to provide for dynamic webpages[1], running in a compatible browser, and suitably authorized by the user, with the capability to set up audio, video, or data communications. The main use-cases are audio conferencing, e-commerce support, and personal or enterprise communications. But other use-cases are also envisioned such as gaming or file sharing. Due to the ease of deploying a WebRTC service: a simple web server is enough, it is expected to see the emergence of a larger number of WebRTC enabled websites.

1: Dynamic HTML is the generic name for the set of techniques used by a webpage in order to modify itself. It is now more often referred to as a JavaScript/HTML/CSS page.

The three main tracks for WebRTC are a set of dedicated protocol profiles defined by the Internet Engineering Task Force (IETF) [27], and two JavaScript Application Programming Interface (API) specified by the World Wide Web Consortium (W3C) for WebRTC session management [28] and for audio and video media capture [29].

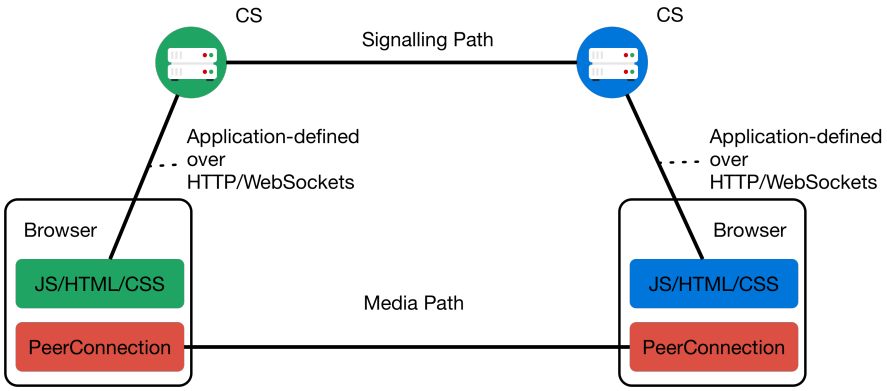


Figure 1.1: WebRTC deployment with two browser endpoints and two signalling servers [27].

These specifications work in conjunction so that "for all options and features of the protocol specification, it should be clear which API calls to make to exercise that option or feature; similarly, for any sequence of API calls, it should be clear which protocol options and features will be invoked" [27]. Although the endpoints targeted by the API are web browser implementing WebRTC, it is also possible to use non-browser WebRTC endpoints. These endpoints conform to the protocol specification but not to the WebRTC API. For instance, such endpoints may be a native library in C++ offering WebRTC call functionality to an application.

Figure 1.2 presents the components of the WebRTC browser model and their interactions. The design of WebRTC does not assume that the browser should provide every functionality required by telephone or conference services. Instead, the browser only offers functions required for a web application to implement such services. Hence why the only vital interfaces are the RTC API and the protocols for browser-to-browser communication. A typical WebRTC communication setup is presented in Figure 1.1. The Media path represents the browser-to-browser communication path over which media and data channels are established. Signalling is the communication between WebRTC endpoints establishing, managing, and controlling the media path. The signalling path generally involves one or more signalling server to serve as a rendezvous point. The mechanisms and architectures for client-server and eventually inter-server signalling are out of scope of the WebRTC specification. We present examples of such architectures in Section 1.6.

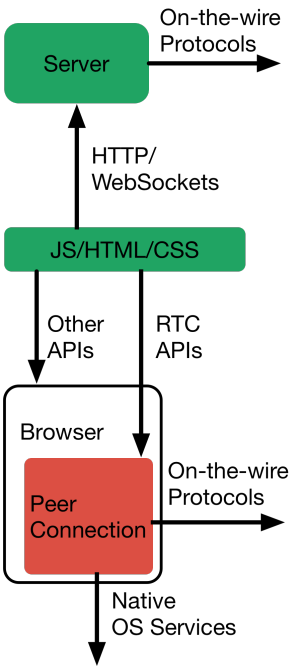


Figure 1.2: WebRTC browser endpoint model [27].

The main interface exposed by the WebRTC API is the RTCPeerConnection [28]. It exposes functions to the webpage's JavaScript allowing it to manage the WebRTC session and signalling. The JavaScript Session Establishment Protocol (JSEP) IETF draft [30] describes how the RTCPeerConnection is used. JSEP permits the establishment and control of a WebRTC session through the exchange of session description messages, in a simple offer/answer negotiation mechanism. When an offer/answer exchange is required, the initiating endpoint calls the `createOffer()` function. The resulting offer is installed as a local configuration, using `setLocalDescription()`, and transmitted on the signalling path. When received, the offer is used as a remote configuration, with the `setRemoteDescription()` interface. Finally, the process is reversed: the receiver generates an answer, applies it as a local configuration, and sends it. On receiving the answer, the initiator installs it as a remote description, completing the initial setup.

Session description messages use the Session Description Protocol (SDP) [31] syntax, as presented in Figure 1.3. The SDP syntax defines a list of parameters, identified by type letters and belonging to one of the description levels: the session description level, the time description level, and one or more media description level. The set of type letters is quite small and not extensible, however, extensions are possible through the

```
Session description
        v=  (protocol version)
        o=  (owner/creator and session identifier).
        s=  (session name)
        i=* (session information)
        u=* (URI of description)
        e=* (email address)
        p=* (phone number)
        c=* (connection information - not required if included in all media)
        b=* (bandwidth information)
        One or more time descriptions (see below)
        z=* (time zone adjustments)
        k=* (encryption key)
        a=* (zero or more session attribute lines)
        Zero or more media descriptions (see below)

Time description
        t=  (time the session is active)
        r=* (zero or more repeat times)

Media description
        m=  (media name and transport address)
        i=* (media title)
        c=* (connection information - optional if included at session-level)
        b=* (bandwidth information)
        k=* (encryption key)
        a=* (zero or more media attribute lines)
```

Figure 1.3: Session Description Protocol syntax

use of `a=` attribute lines.

Figure 1.4 shows an example of a (shortened) SDP message for a WebRTC session. It is an SDP answer for a session with audio and video channels, respectively identified as `sdparta_0` and `sdparta_1`. Declaration of these channels starts with a `m` line specifying the channel type, *e. g.* `m=audio`. In the SDP offer/answer negotiation, peers have to agree on various parameters, for example, audio and video codecs. In order to conduct the negotiation, an offer includes multiple supported parameters. If compatible, the answerer accepts these parameters by including them in his answer. Preceeding this example, the original offer proposed five codecs for the audio channel: `opus`, `telephone-event`, `G722`, `PCMU`, and `PCMA`. Only two, `opus` and `telephone-event`, are included in the answer and thus accepted by the answerer.

In order to efficiently communicate in a Peer-to-Peer (P2P) fashion, peers use the Interactive Connectivity Establishment (ICE) [32] to find which of their network interfaces are able to see each other. Indeed, a computer may be connected to different networks through different interfaces. In the ICE protocol, peers collect candidate addresses and exchange them during the SDP negotiation in `a=candidate` attributes. Each pair of local and remote candidates are then tested to find potential matching interfaces. In some situations, for example behind a Network Address Translator (NAT), a peer may not know his publicly visible address. Session Traversal Utilities for NAT (STUN) [33] is a client-server protocol used by a client to discover the binding between his source address and his reflexive transport address[2]. STUN is a tool that may not work in some configuration, in particular, if both peers are behind NAT routers with endpoint-dependant-mechanisms for address attribution. In that case, a relay can be setup using

2: NAT modify the source or destination of packets going through them. The reflexive transport address is the public Internet Protocol (IP) address and port created by the NAT closest to the STUN server.

```
v=0
o=mozilla...THIS_IS_SDPARTA-54.0.1 5897145307417630851 0 IN IP4 0.0.0.0
[...]
a=fingerprint:sha-256 33:B1:D7:4B:29:29:29:AA:87:01:47:B3:59:41:[...]5D
a=group:BUNDLE sdparta_0 sdparta_1
a=ice-options:trickle
a=msid-semantic:WMS *

m=audio 29188 UDP/TLS/RTP/SAVPF 109 101
c=IN IP4 74.125.39.43
a=mid:sdparta_0
a=candidate:0 1 UDP 2122252543 131.254.67.174 54163 typ host
[...]
a=candidate:20 1 UDP 8331263 74.125.39.43 29188 typ relay [...]
a=rtpmap:109 opus/48000/2
a=rtpmap:101 telephone-event/8000
[...]

m=video 29188 UDP/TLS/RTP/SAVPF 121
c=IN IP4 74.125.39.43
a=mid:sdparta_1
[...]
```

Figure 1.4: Example of a SDP message (answer)

the Traversal Using Relays around NAT (TURN) [34] extension to STUN. Once ICE checks have completed, both peers can set up their secure channels.

1.2 Security on the Web

The World Wide Web, or more commonly the Web, is one of the many applications of the Internet. It allows to retrieve and manipulate resources hosted by a server and referencing other resources by hypertext links. The Web is built on a client-server model, and the most common type of client is, of course, the web browser. Web clients are often referred to as user-agents, particularly in web specifications.

The set of protocols and specifications forming the Web are mainly defined by two standardisation bodies. Languages such as HTML and CSS, protocols such as the HyperText Transfer Protocol (HTTP), and web API such as WebRTC API are standardised by the W3C [35, 36]. As the Web is built on the Internet, the IETF contributes to protocol specifications which are then used in the context of the Web. The European association for standardizing information and communication systems (Ecma) is also an important actor as it standardises the ECMAScript programming language, better known under the name of Mozilla's implementation: JavaScript. Browser makers, as the implementors of the most used web clients, are also central to the definitions, evolutions, and adoptions of web specifications. The most important browser makers are Google (Chrome), Mozilla (Firefox), Microsoft (Edge/IE), and Apple (Safari). Figure 1.5 gives a rough indication of the balance of power between these main four browsers. If a feature is not implemented or removed from one of the popular browsers it will soon be abandoned by websites. Yet, implementing the full standard specifications is a daunting task and browsers, although regularly updated, often offer incomplete support for some features. Keeping up with particular features and implementations from each browser is a complex task for website developers. A common technique is to conditionally use *shims* or *polyfills*, adapter code that abstract browsers' implementations under a single

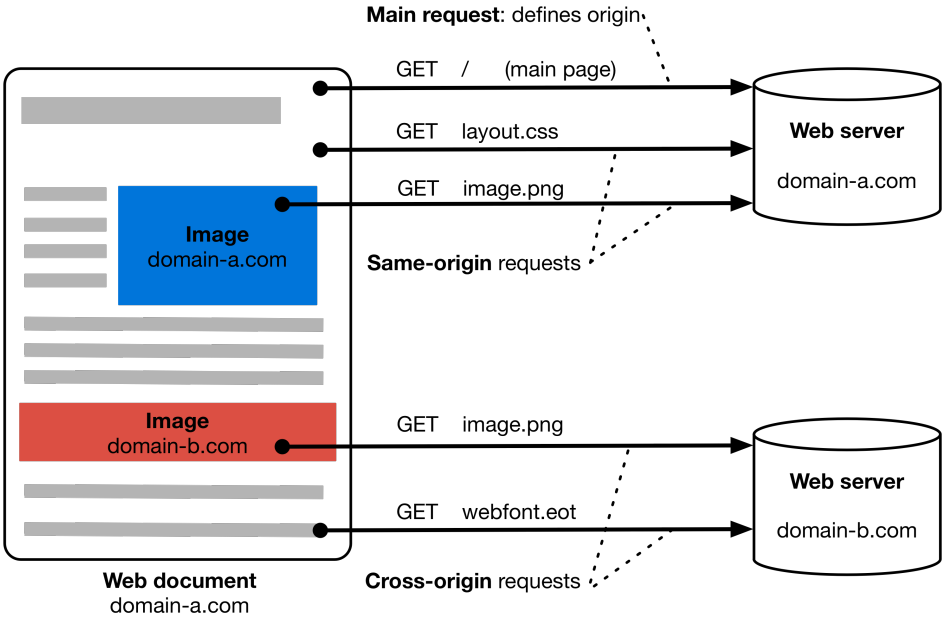


Figure 1.6: Cross-site request example. The HTML page from `domain-a.com` requests resources from `domain-a.com` and `domain-b.com`. Requests for `domain-a.com` are same-origin and always allowed. Requests for `domain-b.com` are cross-origin and requires valid CORS headers and CSP directives. Example taken from `developer.mozilla.org`.

interface.

1.2.1 The Trusted Computing Base

The Trusted Computing Base (TCB) is the set of hardware and software on which the security of a system relies upon. In a correct implementation of a TCB, if a part of the TCB is compromised, then the whole system's security may be at risk. However a compromised component not part of the TCB should not present a risk to the security of the system in general or to any other components in particular. While from an Operating System (OS) point of view the browser is just an application outside of the TCB, in the context of the Web the browser is a central part of the TCB.

In other words, the browser can actually be considered as the OS of the Web. Indeed, it is responsible for downloading and running client web application, while exposing resources and services through web API. As other OS, the browser is also responsible for maintaining applications isolation.

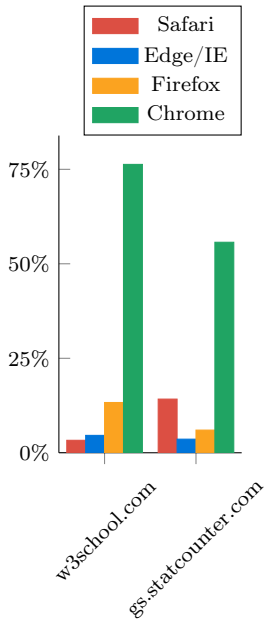


Figure 1.5: Percentage of recorded browsers visiting `w3schools.com` and `amiunique.org` in June 2017.

1.2.2 The Same Origin Policy

For web applications, isolation relies on the concept of origin and the same-origin policy [37]. An origin is a protection domain regrouping Uniform Resource Identifier (URI) [38] sharing the same triple of scheme, host, and port. Two pages from different origins are isolated and cannot share resources, such as variables or cookies. However, communication is still possible using the postMessage API. This interface allows pages from different origins to exchange scoped message and cloned variables.

A web application can also embed contents such as scripts or images from other origins. Figure 1.6 shows an example of a webpage with content loaded from two different domains. Cross-origin content may enable vulnerabilities in a webpage as it enables the coexistence of two origins in a single security-context. Cross-Site Scripting (XSS) and others code injection attacks exploit the user's trust in a webpage. This kind of privilege escalation attack allows a malicious script to gain access to sensitive session information from the security context of the main origin. One way for a webpage to

prevent XSS is to define whitelisted origins through the use of Content Security Policy (CSP) directives [39]. This ensures that only trusted content can be loaded and executed by the web application. Unlike XSS attacks, Cross-Site Request Forgery (CSRF) attacks exploit a web application's trust in a user's authentication to issue commands in place of the user. To issue a CSRF attack a malicious webpage visited by the user executes an illegitimate HTTP request on the vulnerable server. The request may, for instance, be triggered by loading a link inside an HTML image element or the execution of a JavaScript fetch request. Supposing that the user is authenticated on the vulnerable application the illegitimate request inherits from this authentication, for instance through a session cookie. Web applications may deter CSRF attack by enforcing the same-origin policy for some of their exposed HTTP interface. This can be achieved by setting unique and unpredictable csrf-token cookie on the client side, or by defining Cross-Origin Resource Sharing (CORS) [37] headers on HTTP responses. In both types of attacks, CSRF and XSS, the browser is responsible for correctly applying policies defined by web applications in order to protect the user.

Web-based protocols may require the discovery of policy or configuration information through metadata. In order to avoid collisions with existing resources but also prevent impersonation on self-hosting services, Request For Comments (RFC) 5785 [40] defines well-known URI. A well-known URI is a URI whose path components starts with `/.well-known/`. Specifications often define resources located under a well-known URI and assume that they are under control of the host.

1.2.3 The HyperText Transfer Protocol Secure

The HyperText Transfer Protocol (HTTP) [16] is an Internet application layer protocol [41] developed to access and modify web resources. It offers several methods to send resource requests from a client to a server. The two main ones are GET which is used to retrieve a resource, and POST which is used to send data to the server for creating or updating a resource. Responses to HTTP requests are made of a status code and an eventual body containing the actual response, *e. g.* a file or an object. For instance, the status code 200 means that the request is successful while the set of 300 codes indicates a redirection.

HyperText Transfer Protocol Secured (HTTPS) [42] is the combination of the HTTP and Transport Layer Security (TLS) [43] protocols[3]. TLS enables authentication and secure communication between a client and server[4]. To authenticate itself to a client a server must provide a signed and valid certificate containing, in particular, his public key, his domain name, and the certification authority which issued the certificate. After the client validated the certificates, it uses the public key to communicate a secret session key to the server initiating a secure communication between both parties.

Certification authorities are the entities responsible for issuing and signing certificates. There exist four types of HTTPS certificates ranging from self-signed certificates to Extended Validation certificates. These types differ from the verification conducted by certification authority, their costs, and the way they are displayed by browsers. Although there is debate whether Organisation Validation certificates offer more guarantees as Domain Validation certificates. Some Extended Validation certificates are integrated by browser makers within their browser installation packages. These are often referred to as root certificates as they do not depend on other certificates. It is the responsibility of the browser makers to select trustworthy certification authority for providing root certificates.

In addition to initialising trust chains, the responsibility of the browser is to provide security information to the user when they connect to a website. This is usually done by displaying a colour coded indications of the page security level in the address bar and in-context warning such as in Figure 1.7. Mixed contents such as an insecure login forms

3: Note that as HTTP is not considered secure, HTTP and HTTPS page from the same domain are not considered to be from the same origin.

4: Although client authentication is also possible with TLS, it is not much used and often limited to server-server interactions or other particular use cases.

are also taken into account to determine the security level. However, if the certificate is invalid the page is automatically blocked by the browser. While this effectively deters attacks, users usually only see false alarms [44] which could, in turn, have the effect of lowering their vigilance.

Browser makers are currently pushing HTTPS and advocating for a fully encrypted Web. For instance, Firefox indicated that they will advertise HTTP site similarly to bad certificate [13]. A full HTTPS web would lower the trust of users in HTTP websites, and force those site to finally adopt HTTPS. This objective is made possible thanks to automated Certification Authority (CA) such as Let's Encrypt which operates the Automatic Certificate Management Environment (ACME) protocol. Let's Encrypt aims at drastically reducing the complexity and cost of certificate deployment by avoiding manual operations, a major barrier for the multitude of small websites. Ultimately, the usage of HTTPS is not a proof of trustworthiness of the website itself but only that the control of its domain name was verified to some extent and that secure communication is in place.

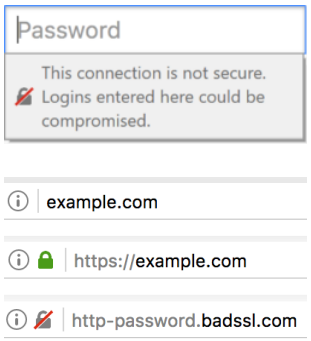


Figure 1.7: Security indications for HTTP, HTTPS and insecure password form on Firefox. The HTTP indicator will soon be replaced by a warning similar to the example shown here.

1.2.4 Public Key Cryptography

This section introduces the notion and usages of public key cryptography, a cornerstone of the security of the Web. This technique allows to sign, authenticate, and secure messages exchanged by two peers without the need to share a common secret key. Public key cryptography is used in the certificate infrastructure deployed by browsers and CA for HTTPS, but also in the WebRTC security protocols, and in authentication delegation protocols.

In 1976 Diffie and Hellman published the concept of public key cryptography, also called asymmetric cryptography. The race for such an algorithm was won by Ron Rivest, Adi Shamir, and Leonard Adleman with the publication of the famous RSA algorithm. In symmetric cryptography, the same key is used to encrypt and decrypt the message. This is, for instance, the case with Caesar's cypher where the key is the number of shifts to apply in order to encrypt a plaintext message, but also the number of opposite shift to decrypt the message. Conversely, public key cryptography protocols use two keys. The secret key is randomly generated while the public key is derived from the secret key, both forming a key pair. As knowledge of the public key does not allow to retrieve the private key, the public key can be shared freely. We do not go into details on the specific cryptography mechanisms and algorithms but focus on the use cases made possible by asymmetric cryptography.

In order to bind an identity to a key pair, a CA issues a digital certificate. These certificates contain information about the public key, the signing algorithm, the owner's identity, and the CA. Certificates may also refer to the CA's own certificate until a trusted root certificate is reached forming a chain of trust. X.509 certificates, one of the most used standards for digital certificates, contain several fields describing the certificate itself, the issuer, and the subject. An example is given in Figure 1.8. Another way to issue certificates is to participate in a web of trust where members respectively sign their keys during key exchange party. OpenPGP [45] is the most widely used web of trust standard, particularly for signing emails.

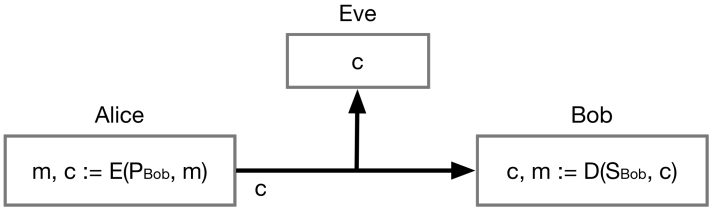


Figure 1.9: Public-key encryption of m from Alice to Bob [46].

Subject	
Common Name	energyq.idp.rethink.orange-labs.fr
Subject's Public Key	
Public Key Algorithm	PKCS #1 RSA Encryption
Public Key	c5 44 1c 33 79 bf e5 [...] c9 26 a4 e3 5e 4e 6d
Issuer	
Common Name	Let's Encrypt Authority X3
Period of Validity	
Begins On	31 mai 2017
Expires On	29 août 2017
Fingerprints	
Signature Algorithm	PKCS #1 SHA-256 With RSA Encryption
SHA-256 Fingerprint	77:21:3C:64:0A:ED:E3:AD:0F:82:03:6D:9B:45:66:B4: D0:28:D8:04:9E:F3:41:C1:C7:A5:EA:57:BB:A0:34:1D

Figure 1.8: Extract of a X.509 certificate issued by Let's Encrypt to energyq.idp.rethink.orange-labs.fr.

In the public key encryption scenario (Figure 1.9), Alice wants to send an encrypted message m so that it is only readable by Bob, the intended recipient. Bob possesses a key pair (P_{Bob}, S_{Bob}) and publishes its public key. Alice then uses this public key to encrypt the message and send the cyphertext to Bob. On receiving the encrypted cyphertext, Bob uses its secret key to decrypt the message. While the attacker Eve could not read the cyphertext, it could still intercept the message and modify it. Confidentiality is ensured, but encryption does not prove message integrity nor the sender's authenticity.

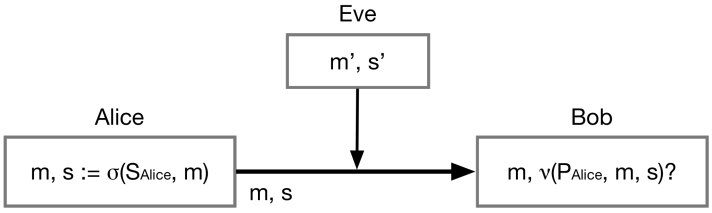


Figure 1.10: Public-key signature of a message m from Alice to Bob [46].

In the signature scenario (Figure 1.10), Alice possesses a key pair (P_{Alice}, S_{Alice}) and publishes its public key. In order to sign a message m, Alice uses a signing algorithm with the private key and the plain text as a parameter to produce a tag, *i. e.* the signature s. She then sends both plain text and signature to the recipient. The signature algorithm is often associated with a cryptographic hash function, a one-way function mapping the message to a fixed size string. In this case, the signature is applied on the resulting hash, also called a digest. By verifying the signature with the public key and comparing it to the message, or the message's digest, the receiver is then able to verify the integrity of the received message. Only the holder of the secret key could have produced the signature. If the key pair is bound to the sender's identity, for instance with a certificate containing the sender's name, the receiver also verifies the authenticity of the message.

Achieving both confidentiality and integrity is possible if both parties publish their respective public keys. To do so the sender would sign the message with its own private key and then encrypt both message and signature with the other party's public key. Using asymmetric encryption is, however, a computationally heavy operation compared to symmetric encryption. In order to solve this problem, public key cryptography is

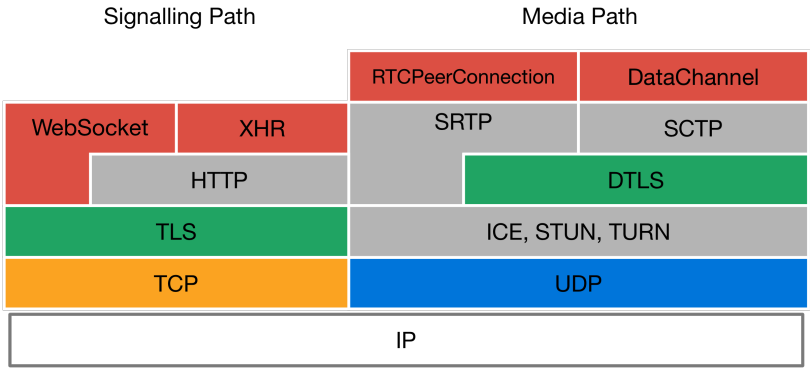


Figure 1.12: WebRTC Protocol Stack [48].

used to establish a symmetric session key. Such key exchange scheme is often referred to as a Diffie-Hellman exchange.

A cypher suite is a standardised configuration of cryptographic algorithms and options meant to serve as a toolbox for establishing a secure communication. In TLS, a cypher suite would define the key exchange algorithm, the bulk encryption algorithm, the signature algorithm and the relevant key size to be used. Prior to starting the negotiation, both peer would make a hand-shake to compare their respective available cypher suite and find a common ground. Figure 1.11 shows the cypher suit used on `idp.energyq.rethink.orange-labs.fr` with Firefox.

TLS_ECDHE_RSA_WITH_AES256_GCM_SHA384

Figure 1.11: This cypher suit defines ECDHE as the key exchange mechanism, RSA as the authentication mechanism, AES_256_GCM as the symmetric encryption algorithm, and SHA384 as the digest algorithm.

1.3 WebRTC Security

While the signalling path is secured as any other client-server connection on the Web. The media path serves for real-time media communications and as such uses different protocols. Figure 1.12 shows the protocol stack for the WebRTC signalling and media path. WebRTC mandates the use of secure media channels[5] [47]. In this section we present the protocols protecting confidentiality, integrity, and availability of the media channel. The WebRTC security architecture also introduces an optional identity path to let users verify their peer's authenticity. We give definitions related to user authenticity in Section 1.3.3 and present the WebRTC Identity Path in Section 1.3.4. We also give an overview of some alternative key management protocols in Section 1.3.6.

5: A consistent decision with the direction for HTTPS everywhere.

1.3.1 Confidentiality and Integrity of the Media Path

The media path is a P2P connection between two WebRTC endpoints. This path allows both the setup of media streams offered by the Media Capture and Streams API or data channels from the RTCDataChannel API.

Media streams are transported over the Secure Real-time Transport Protocol (SRTP) [49], a secure profile for the Real-time Transport Protocol (RTP) [50] and RTP Control Protocol (RTCP) "which can provide confidentiality, message authentication, and replay protection". Usually, the RTP protocol (and SRTP) is carried over the User Datagram Protocol (UDP) [51] as real-time media streams are time-sensitive[6]. As SRTP does not offer key management functionality, it must rely on another key management protocol. The preferred and default protocol for SRTP key management is the DTLS-SRTP (SRTP profile for DTLS) [53] which must be offered by any WebRTC implementation. Datagram Transport Layer Security (DTLS) [54] is an adaptation of TLS to datagram transport protocols. SRTP profile for DTLS [55] is "a SRTP extension for DTLS that combines the performance and encryption flexibility benefits of SRTP with the flexibility and convenience of DTLS-integrated key and association management". SRTP profile

6: Contrary to the Transmission Control Protocol (TCP) [52], UDP does not guarantees the delivery, ordering, or duplicating of its packets and avoids the overhead of such guarantees.

for DTLS can thus be seen as a DTLS optimisation for RTP. On the other hand, data channels use the Stream Control Transmission Protocol (SCTP)[7] [56] and are secured by DTLS only.

7: SCTP can be configured to provide similar features to UDP or TCP.

In order to exchange their public keys and setup a session key, peers proceed to a DTLS handshake over the media path by exchanging certificates. Contrary to server exposing certificates signed by trusted CA, peers' certificates are self-asserted and thus untrusted. In order to authenticate the received certificates, fingerprints of these certificates are previously exchanged over the signalling path in session level `a=fingerprint` SDP attributes (see Figure 1.4 for an example). As long as the signalling path is trusted, the confidentiality and integrity of the media path is ensured.

1.3.2 Availability of the Communication

The main threat to availability of a WebRTC communication consists in Denial of Service (DoS) attacks, and in particular Distributed Denial of Service (DDoS). As WebRTC calls are programmatically controlled, a malicious software could launch multiple calls from several users to a single destination, for instance a call-center. The consent mechanisms required by the Media Capture and Streams API mitigate these kinds of attack using media streams. However, data channels do not require consent before being initialised and may be used to mount DDoS attacks. Client applications and automated WebRTC implementations, *i. e.* call-center gateway, should implement filtering policies to detect and respond to suspicious call, *i. e.* call without audio or video. The WebRTC security architecture [47] details and reference additional availability attacks against WebRTC.

More generally, WebRTC clients may also be vulnerable to SPam over Internet Telephony (SPIT), *i. e.* the transmission of bulk unsolicited call offers. The possibility of WebRTC spam depends on the underlying signalling architecture. For instance an open signalling federation would be more vulnerable to SPIT than a one-way call service deployed on a website. Nonetheless, there is few doubts that in the former case a spam agent would be easy to write. RFC 5039 [22] references solutions to SPIT call for Session Initiation Protocol (SIP) that could probably be adapted to WebRTC scenarios. One solution proposed by the RFC is to implement white and black lists to automatically accept or reject incoming calls. However, obtaining a new identity may be quite cheap making black lists ineffective.

Finally, a WebRTC session may be impacted by the available bandwidth on the network. Janczukowicz *et al.* [57] demonstrate that concurrent TCP flow may have an impact on the perceived quality of a WebRTC flow under best-effort routing. Saturating a network path could thus be used to compromise the availability of the communication. Alternative routing mechanisms could be used to provide specialised and paid communication services with a managed quality of service. For instance using TURN servers for routing the media path, rather than using P2P best-effort [58]. However, specialised services on the internet are criticised by some as being against the philosophy of the Net Neutrality[8].

8: The french Regulation Authority for Postal and Electronic Communications (ARCEP) published a state of the art report on net neutrality regulations and operators practices in September 2015 [59].

1.3.3 User Authenticity

For users navigating the Web, authentication is a common action involving an identifier and a password. But it may be difficult for them to explain what are the authentication inherent concepts, and what precisely is their identity. In the following paragraphs, we define and explain the concepts of identity, identifier, credentials, and claims. We explain what constitutes an authentication process, and what makes it secure.

A subject's *identity* is a set of attributes, called claims, representing the subject in a specific scope [60]. As an identity is only defined for a specific scope, and a subject may exist in multiple scopes, a subject may have multiple identities. For instance, the claims

described by a subject's driver license constitutes one of his identity. The same subject may also have a passport and some social network accounts. Although these identities belong to the same individual and may overlap, for instance sharing the same name and age, they are distinct identities.

Identity is often used in the sense of an identifier [60, 61]. This confusion is due to the fact that a name, or a public identifier in digital identity, is often used as a synonym for an identity. In the rest of our thesis, we will clearly make the distinction to avoid confusion.

Claims are the attributes constituting an identity. Identity claims can describe features of a user such as his name, his age, or his address. But claims may also be used to grant access to resources. In that case, we talk about authorization claims. Though some of the claims on a driver license would be describing the driver itself, *e. g.* `name: Bob` or `age: 30`. Some other claims may describe the category of cars the driver is authorized to drive, *e. g.* `cat. A: authorized`. On the Web, an authorization claim may allow write and read access to some data or API. Claims are usually uniquely identified by a URI or reserved names defined by various standards. For instance the OpenID Connect [62] specification defines a set of standard identity claims. As claims can change over time, an identity may also be dynamic.

An *identifier* is a claim uniquely identifying the subject in a given scope. Identifiers may have different forms such as a pseudonym, an email address, or a random text. A website identifies the claimed identity by looking up its records and verifying if such identifier exists.

Authentication is the process by which an entity proves the validity of some claims to another party. In the most common scenario, a user authenticates by claiming his identifier. In this case, we say that the user authenticates himself (or just authenticates if unambiguous). In order to prove the ownership of an identifier, *i. e.* to authenticate, the user has to present valid *authentication factors*. Contrary to identifiers, authentication factors are not necessarily unique. Indeed, several users may, by coincidence, use the same password.

Although authentication commonly refers to proving the ownership of an identifier, it is also possible to authenticate non-identifying claims. For instance, to protect its privacy a user may want to claim only his age without revealing his identifier. In order to allow for this scenario, a trusted third-party is often necessary to assert the validity of claims. However, identification remains possible if enough claims are provided.

In the most simple case, authentication is a simple relation involving two participants. However, for privacy, usability, or security reasons, the process may be delegated to a trusted third party. An Identity Provider (IdP) is a trusted authority issuing and validating identity claims. The Relying Party (RP) is the consumer of identity claims. Technically, claims are requested, obtained, and validated using authentication delegation protocols. These protocols allow the exchange of claims through security tokens. Depending on the protocol, the RP requests some claims to an IdP or to the subject before granting access to a service[9].

9: An RP is also commonly referred to as a Service Provider. However, we reserve this term for the context of WebRTC. Note that due to the need of authenticating users, a WebRTC Service Provider is often an RP too.

Usually, the IdP is responsible for the lifecycle of an identity. This includes the enrolment, the management, and the eventual revocation of an identity. Before validating any claims, verifying authentication factors, and checking an identifier, the identity must first be created. *Enrolment* is the registration process used to create or update an identity. In this situation, there are two possible stances: to define a new identity ex nihilo or to create an identity from a previously existing one, *e. g.* by authenticating a national identity card. In some use cases, *e. g.* banking or communications, a server may require a user's identity to be anchored into a trusted identity. To cater for these different needs, the Authentication Assurance Levels (AAL) have been proposed by several national or regional frameworks and later standardised as ISO:29115:2013 [63]. These frameworks

categorise the AAL into four roughly equivalent categories, though details may differ for historical or cultural reasons. Hatin *et al.* [64] propose a mapping of these discrete levels into a continuous authentication score. Authentication strength may also be associated with a time to live or an eroding factor applied over time. Such authentication model shows similarity to trust models (see Section 1.4) and current work on pervasive authentication systems further contributes to closing the gap between authentication and trust levels [64].

Alternatively, some servers may be satisfied to know that a user is indeed just a returning customer. In these cases, the actual identity of the user is not that important and the server is more concerned with the authentication. Such identity can be qualified as a declarative identity and its claims are said to be self-asserted. Although confidence in the authenticity of such identity may be low, anchoring it into a source of reputation such as a social network may prove its trustworthiness. Conversely, verifying that identity documents are genuine and up to date is a complex task [65, 66].

1.3.4 WebRTC Identity Path

In some scenarios, users may not trust the website or the underlying communication provider on which they are making the call. Indeed, the signalling server may be lying about the connection authenticity and could be mounting a man-in-the-middle attack. This type of attack, presented in Figure 1.13, allows an invisible attacker Eve to be setup in the middle of the media path between Alice and Bob by a malicious signalling server S. When Alice sends her SDP offer to Bob through S, the offer is instead relayed to Eve. Eve then generates an offer to Bob and configured so that Alice's media stream is relayed to Bob. Bob believing to be receiving a legitimate call from Alice replies to Eve with his own media. Again, Eve relays Bob's media to Alice. While Alice and Bob talk to and see each others over encrypted media streams, the encryption is not end-to-end. Alice and Bob have actually negotiated their media path encryption with Eve, without knowing it, and Eve is thus able to decrypt their streams.

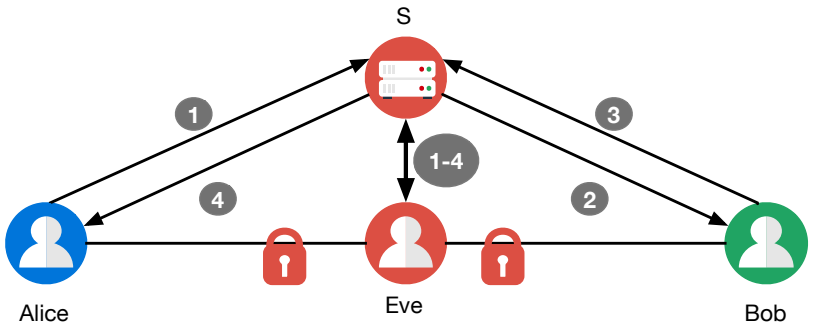
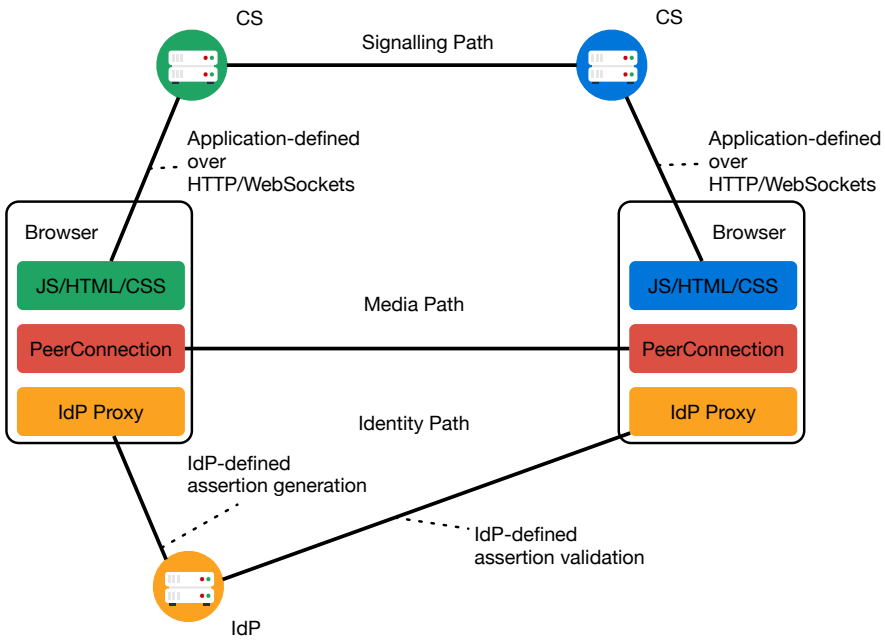


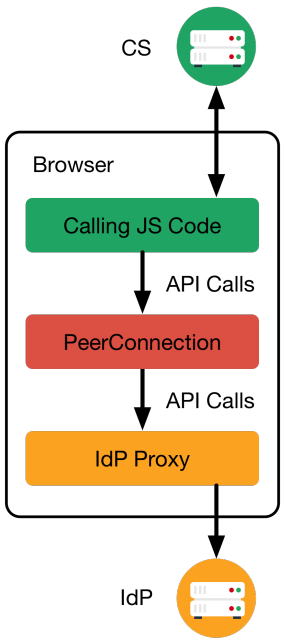
Figure 1.13: WebRTC Man-in-the-Middle attack. The SDP messages are: (1) Alice's offer relayed to Eve, (2) Eve's offer impersonating Alice, (3) Bob's anwser, (4) Eve's answer.

To solve this issue, WebRTC proposes a mechanism for peers to authenticate to one another using an IdP as a trusted third party. We refer to this mechanism as the WebRTC identity architecture, presented in Figure 1.14a. To authenticate, peers bind their session fingerprint to a human readable identity by providing an identity assertion. This identity assertion covers two main claims: the user identifier and the session fingerprint used. By verifying the assertion and comparing it to the received session fingerprint attribute, peers can authenticate the other peer and verify that no man-in-the-middle attack is being mounted.

In order to enable the generation and the verification of identity assertions from any authentication delegation protocol and providers, the WebRTC security architecture [47] specifies the IdP Proxy component. This component serves as an interface between the



(a) WebRTC deployment with two browser endpoints, two signalling servers, and a single identity provider authenticating the left-side peer.



(b) WebRTC Identity API Architecture

Figure 1.14: WebRTC Identity Architecture

WebRTC PeerConnection object and the IdP through the interface presented in Figure 1.15. The IdP Proxy is available at a well-known location on the IdP domain. Before making an SDP call offer or answer, the PeerConnection calls the IdP Proxy to generate an assertion covering the session fingerprint. After the IdP authenticated the user, the identity assertion is returned. This assertion is then included in the SDP message in an `a=identity` attribute, along with the IdP Proxy location on the IdP domain. This allows peers to discover IdP Proxy location without prior knowledge or relationship with the IdP. On receiving an SDP message containing an Identity Assertion, the PeerConnection object downloads the IdP Proxy from the specified location. It then calls the IdP Proxy's assertion verification function. If the verification is successful, the session fingerprint and the user identity are returned.

```
dictionary RTCIdentityProvider {
    generateAssertion(
        DOMString                   contents,
        DOMString                   origin,
        RTCIdentityProviderOptions  options,
    ),
    validateAssertion(
        DOMString                   assertion,
        DOMString                   origin,
    )
```

Figure 1.15: Interface Exposed by Identity Providers in WebIDL.

The WebRTC specification allows users to choose a default IdP in their browser preferences if none was specified by the Communication Service (CS). But implicitly, if the CS sets up an IdP to authenticate a WebRTC call, we would expect it to be the

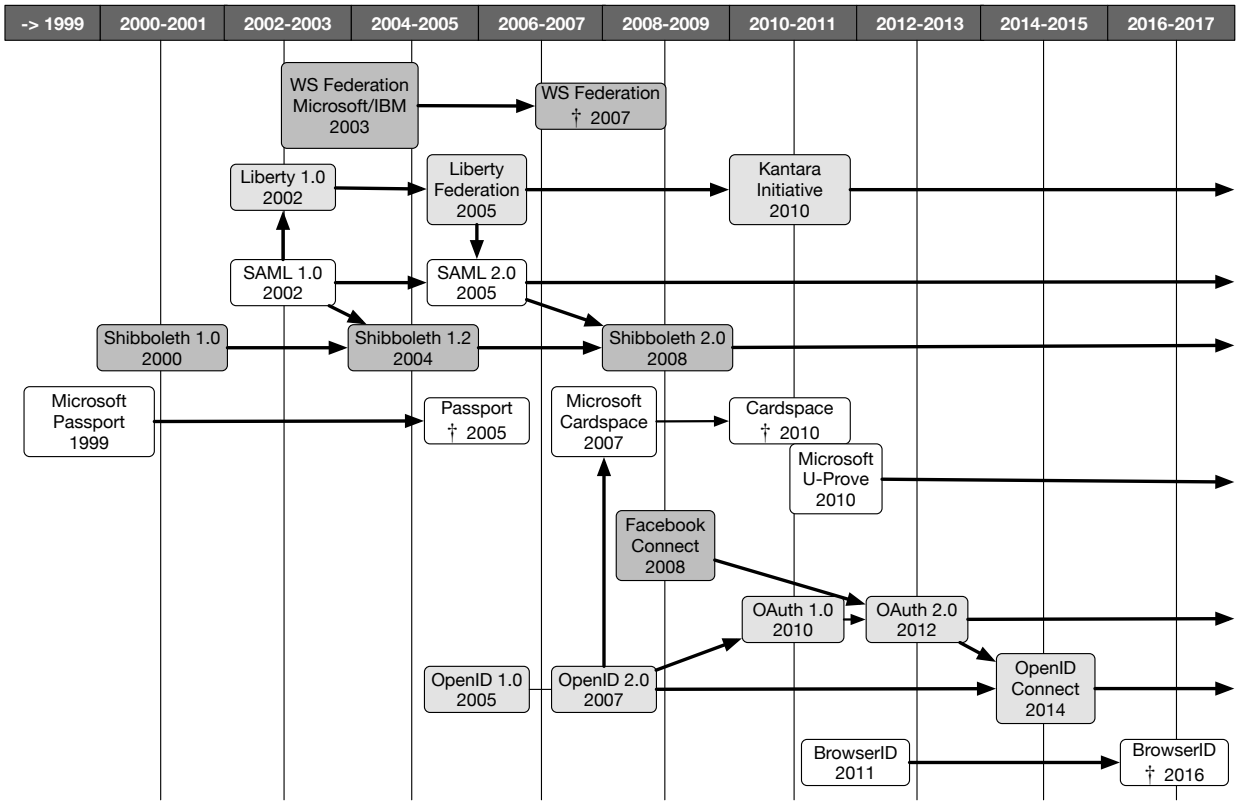


Figure 1.16: Evolution of standards and technologies for user identity management, updated from Jøsang's 2014 survey [67].

same IdP used to sign into the CS's website. Indeed, the CS must be able to handle the authentication procedure and needs to know that the user has an active account on this IdP. Besides, from a user experience perspective, it is important that the identity presented on the CS webpage and the one received in the identity assertion are consistent with each other. In practice, the choice of IdP would thus be defined by the CS and limited in the same way as for common authentication delegation services. We refer to this constraint as *identity continuity*.

1.3.5 Considered Protocols for WebRTC Peer Authentication

In this section, we present the authentication delegation protocols considered for integration with the WebRTC identity architecture [47]. These protocols are the BrowserID - Mozilla Persona protocol and the OAuth 2 protocol and its identity extension OIDC. Figure 1.16 replace these protocols in the history of standards and technologies for user identity management. As it is a technical prerequisite to both protocols, we first explain the JSON Web Token standard.

JSON Web Token

JSON Web Token (JWT), specified by RFC 7519 [68], is a way of representing claims to be exchanged by two parties. These claims are encoded in a JavaScript Object Notation (JSON) [69] object which can then be signed as a JSON Web Token Signature (JWS) [70] or encrypted as a JSON Web Token Encryption (JWE), [71]. JWS and JWE tokens contain several JSON members and in particular, a header and a payload or cyphertext,

i. e. the JWT claims set. These tokens can be compactly serialised by concatenating their base 64 encoded members. The acronym JWT is commonly used in place of JWS or JWE depending on the context.

For instance, the following set of claims describes the subject `248289761001` whose name is `Jane Doe`. The token was issued by `http://server.example.com` at `1509105796` seconds since the Portable Operating System Interface (POSIX) Epoch, *i. e.* January the 1st of 1970.

```
{
  "iss": "http://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1509192196,
  "iat": 1509105796,
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "gender": "female",
  "birthdate": "1987-01-01",
  "email": "janedoe@example.com",
  "picture": "http://example.com/janedoe/me.jpg"
}
```

The following JavaScript Object Signing and Encryption (JOSE) header describes the usage of the `RS256` algorithm and the key with id `1e9gdk7`. RS256 is the compact representation of the `RSASSA-PKCS-v1_5 using SHA-256` algorithm, *i. e.* a public key signature algorithm.

```
{"kid":"1e9gdk7","alg":"RS256"}
```

Serialisation of this token is done by encoding both components in base 64 and concatenating them separated by a dot. The signature is then computed by using this string and the specified algorithm and appended to the serialised token. The following text is the previous JWT and JOSE header examples signed into a JWS. The red part is the header, the violet is the JWT payload, and in blue is the signature [10].

[10]:

```
eyJraWQiOiIxZTlnZGs3IiwiYWxnIjoiUlMyNTYifQ.eyJpc3MiOiJodHRwOi
8vc2VydmVyLmV4YW1wbGUuY29tIiwic3ViIjoiMjQ4Mjg5NzYxMDAxIiwiYXV
kIjoiczZCaGRSa3FOMyIsIm5vbmNlIjoibiOwUzZfV3pBMk1qIiwiZXhwIjox
NTA5MTkyMTk2LCJpYXQiOjE1MDkxMDU3OTYsIm5hbWUiOiJKYW5lIERvZSIsI
mdpdmVuX25hbWUiOiJKYW5lIiwiZmFtaWx5X25hbWUiOiJEb2UiLCJnZW5kZX
IiOiJmZW1hbGUiLCJiaXJ0aGRhdGUiOiIxOTg3LTAxLTAxIiwiZW1haWwiOiJ
qYW5lZG9lQGV4YW1wbGUuY29tIiwicGljdHVyZSI6Imh0dHA6Ly9leGFtcGxl
LmNvbS9qYW5lZG9lL21lLmpwZyJ9.iYiil5mBuznlG5aGMnOeHkVALg41q1aK
BFCoXW2f5sth3mZtkCwaRWUbj18pMN7iXlJOOdp9YVthccXORflV1hHlMKnrI
aqlsoTo6aSnAHVgCJc64pKKMTOsIFOEVhcJ-obigxxOUOvme8AV1oQQkn8Bzc
2Dd9E5D0qJX5YbeDw
```

Figure 1.17: A JWT Example: OIDC ID Token. Visit `https://jwt.io` to decode the token. It can also be verified using the following public key:

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4
GNADCBiQKBgQCrpVE2fAdanHGf
HA1ORkmNPIFvCry5XMccRguIGR
zU9wgVBfJ+UeChN9GmcmGf67bE
GbtOY7mScWidKpm3u+XZUOXfl3
PQTF3kIPzKU2cOUwDeziHRmGKR
QXvtTy2esBH45GKzKjFHH6ti6o
Uy3QG7wSZ7kXGGS6pgXjkPBU6y
qwIDAQAB
-----END PUBLIC KEY-----
```

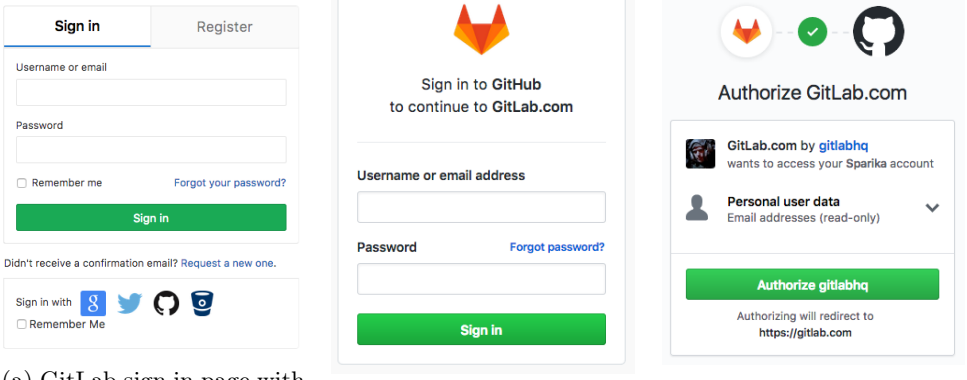


Figure 1.18: OAuth 2 process involving Github as AS and RS, and Gitlab as client.

(a) GitLab sign in page with a selection of IdP.

(b) Redirection to Github sign in page for Gitlab.

(c) Github consent page to share email with Github.

BrowserID - Mozilla Personna

BrowserID [72] was Mozilla's attempt at a decentralized Single Sign-On (SSO) protocol. Though it has been deprecated in 2016 due to a declining usage, the protocol was a central example of the WebRTC identity architecture [47] interoperability claim. Mozilla Persona was the BrowserID service hosted by Mozilla and using a simple email ownership verification mechanism to authenticate users. The reason for the failure of BrowserID is probably that it did not attract enough interest from potential IdP.

BrowserID adds the *id* property to the browser's *window.navigator* object. This property exposes several functions allowing a website client script to request an identity assertion from the browser. The browser responds to identity assertion requests with a signed JWT. This JWT contains the user identifier and a certificate for the user's public key corresponding to the JWT's signature. The IdP's public key verifying the certificate is publicly exposed on the IdP. Any website can verify the certificate's signature thus establishing a trust chain from the user to the IdP.

The BrowserID protocol allows any domain to be used as an IdP by signing a certificate for the user. The user identifier must be an email address whose domain corresponds to the IdP's domain name. The email's domain allows IdP discovery as the location of the IdP's public key and other endpoints are standardised on a `/.well-known` route. The browser stores user's certificates and is then responsible for signing identity assertion for requesting website. This separation of functionalities between the IdP and the browser allows for a privacy-friendly approach as the IdP is not aware of which site is the user login into.

Hiding the user's login actions behind a trusted component is also an approach proposed by Orange with the Trusted Identity Module [73]. Similarly, our technique described in the patent for a Method of managing the authentication of a client in a computing system [4] protects user privacy by putting the identity assertion in a decentralised network such as a blockchain. In these architectures, the IdP cannot monitor who is verifying the identity assertion.

OAuth 2 and OIDC

With more and more content being published by users, websites and in particular social networks need a way to securely expose their API. In order to respect users' privacy, it is, however, necessary to get users' consent to share private information. The original OAuth specification was published in 2006 as a solution to the problem of authorization

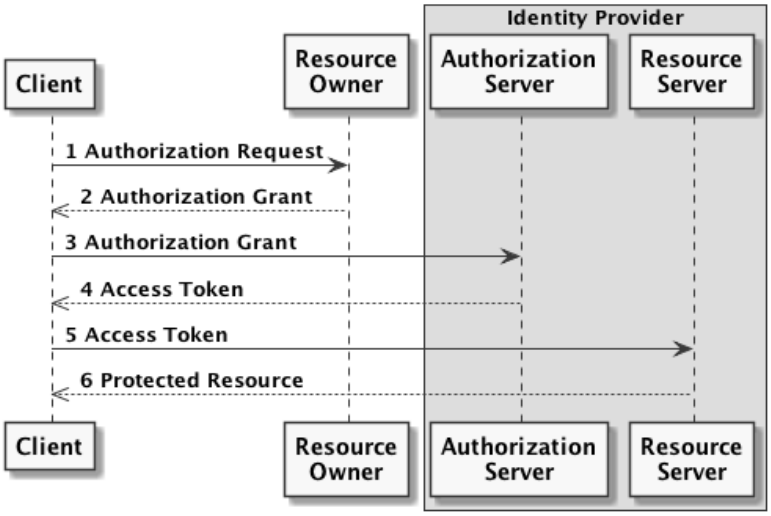


Figure 1.19: Abstract Oauth 2 code flow taken from RFC 6749.

delegation. OAuth was then standardised by RFC 5849 [74] and later deprecated by RFC 6749 [75]: the OAuth 2.0 protocol.

The OAuth 2 specification defines four roles. The Resource Owner (RO), in most case a user, is legally capable of granting access to the resource. The Resource Server (RS) is where the resource is hosted. The Client is the application requiring access to the resource on behalf of the resource owner and with its authorization. Finally, the Authorization Server (AS) can grant an access token to the client after getting the resource owner consent.

The main OAuth 2 protocol flow is called the code flow, represented in Figure 1.19. Figure 1.18 shows the user actions in such a flow. An OAuth 2 flow is based on HTTP and make use of redirection response (code 300) to let servers exchange information on behalf of the user. In this flow, the client makes an HTTP request for an authorization from the RO for a specific resource (1 in Figure 1.19). The AS authenticates the user (Figure 1.18b) and then asks for his consent (Figure 1.18c). The requested authorizations are transmitted to the AS in the *scope* parameter of the HTTP authorization request. After the AS got the user's consent, the request is redirected to the client (2). This redirection contains an authorization code, for instance in the URL query parameter, which can be retrieved by the client. The client then exchanges the authorization code with the AS for an access token (3, 4). This exchange happens out of the RO scope and allows the AS to authenticate the client. The access token is then used to access the resource on the RS (5,6). In order to be authenticated by the AS, the client must be registered. The process of registration most often involves the client's developer taking manual actions to register on the AS through HTML forms. However, automatic discovery and registration are also possible if the AS exposes some metadata and registration endpoints.

In the alternative implicit flow, the AS directly responds to the authorization request with the access token rather than with an authorization code. This simplified flow is adapted for client implemented in the browser rather than on a server. In such scenario, the client code is visible from the user and cannot protect a secret key. As a result, the client cannot be authenticated by the AS and the authenticated authorization grant is unnecessary. However, the resulting process is less secure as the access token may be exposed on the web browser.

While OAuth 2 is able to convey user authentication information, it does so in non-standard ways left for implementors to decide. Actually, authentication in OAuth 2 is

implicit as it is assumed that the user was authenticated by the AS in order to get the user's consent. For the client, getting the user's email means that the owner of the email has been authenticated, but it does not know much more than that.

OpenID Connect (OIDC), developed by the OpenID foundation, is "a simple identity layer on top of the OAuth 2.0 protocol" [62]. It allows clients to retrieves an ID Token in parallel from the access token. This ID Token is a JWT, containing information related to the user's identity, as the one in Figure 1.17. In addition to the ID Token and additions to the OAuth 2 protocol, OIDC standardises a set of identity claims. A client using OpenID Connect and requesting an ID Token must add the *openid* scope to his OAuth 2 authorization request. The received ID Token contains a list of attributes describing the user's authentication result. In particular, the triple of identifiers *iss*, *sub*, *aud* respectively identify the token ISSuer (*i. e.* the AS), the SUBject (*i. e.* the user), and the token AUDience (*i. e.* the client). The audience identifier certifies to the client that it is the intended audience of the token and is not subject to replay attacks. Note that to avoid correlation attacks (as described in Section 1.5), the subject identifier must be locally unique *i. e.* the same user must not have the same `sub` identifier for two different clients. The ID Token also contains expiration time and security parameters such as authentication time, nonce, and authentication level. The requested claims describing the user identity can either be contained in the ID Token or retrieved from the UserInfo endpoint. This endpoint is an OAuth 2 protected resource that can be accessed with a valid access token for the `openid` scope. While OIDC standardises a set of identity claims, additional claims can be added freely to both the ID Token and the UserInfo endpoint as long as they use collision-resistant names or that naming conflicts are unlikely.

From a user's point of view, the difference between OAuth 2 and OIDC flows should not be visible.

1.3.6 Alternative Key Management Protocols

SRTP profile for DTLS and DTLS in conjunction with the use of the identity path allow to bind negotiated keys to an identity authenticated by an IdP. Although these are the default and preferred key management protocols respectively for media and data channels, alternative key management protocols offer different security guarantees. In this section, we present the SDES and ZRTP protocols.

The Session Description Protocol Security Descriptions (SDES) [76] was the main competitor to SRTP profile for DTLS as the default security protocol for WebRTC. The main advantage of SDES consists in its existing deployment in existing SIP/RTP [77] based Voice over IP (VoIP) networks[11]. In particular, interconnectivity between WebRTC and SIP/RTP endpoints would require the deployment of media gateways. As media gateways are complex and costly to implement, leveraging the existing implementations of SDES gateways would reduce costs and favour WebRTC interconnectivity. Additionally, due to the existing SDES deployment, gateways may not have to decrypt/encrypt session negotiated with SDES thus achieving end-to-end encryption. However, contrary to DTLS, in the SDES protocol keys are exchanged in clear over the signalling path inside SDP messages. This implies that the CS can decrypt the streams invisibly. While this seems to be a major vulnerability, in the context of real-time communication it may be a legit requirement from CS as they may be subject to lawful interception requirements. In comparison, mounting lawful intercept against SRTP profile for DTLS is mainly possible through a Man-in-the-Middle (MitM) interception as described earlier[12]. DTLS was considered by the IETF as offering stronger security guarantees and qualified as mandatory to implement. Furthermore the implementation of both the DTLS and SDES protocols may allow for a downgrade attack. To implement such attack, a malicious service would have to modify the SDP messages to only advertise SDES

11: VoIP designates the techniques to communicate using voice or voice and video over any compatible IP networks.

12: Note that in interconnectivity scenarios, gateways can decrypt SRTP profile for DTLS streams unless both endpoints are using SRTP profile for DTLS in conjunction with verified identity assertions.

support. As a result, peers could negotiate a lower security level, allowing the service to then mounts an invisible interception attack. Not only was SRTP profile for DTLS preferred over SDES, but SDES was also not authorized for implementation in WebRTC endpoints.

The ZRTP protocol [78] is a media path keying protocol for SRTP. It uses a simple Diffie-Hellman exchange to agree on the symmetric session key. A Short Authentication String (SAS) is then derived from the session-key and vocally transmitted over the media path (see Figure 1.20). As both peers compare the exchanged SAS, *i. e.* $hash(K_A) = hash(K_B)$, they get a confirmation that they negotiated the same session-key and that no MitM is being set up, *i. e.* $K_A = K_B$. Although the SAS comparison is vulnerable to a real-time audio spoofing attack, in practice it is quite complicated. Indeed, to succeed and remain undetected, the attacker must inject his own SAS at the right moment. This requires detecting, how, when, and which one of the peers will say the SAS. Video session may further deter such attack. However, practical attacks and bugs on ZRTP implementations have been recently reported [79]. As a matter of fact, ZRTP is currently implemented by multiple VoIP clients offering end-to-end encryption. But their reliance on the SAS vocal comparison procedure and their implementations of different security indicators and procedures may be confusing for most users. ZRTP is also less flexible than the IdP Proxy solutions of WebRTC as it is only adapted for the VoIP use case. WebRTC data channels, uni-directional communications, or non-human peer use cases cannot be secured by ZRTP.

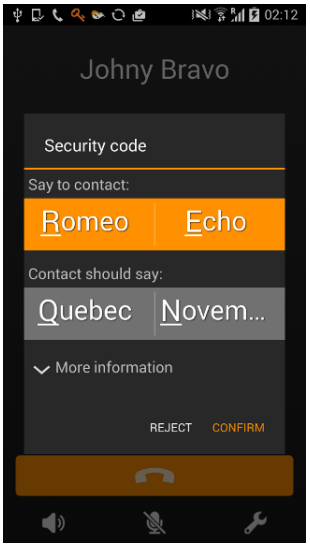


Figure 1.20: Phone-X application displaying the ZRTP SAS.

1.4 Trust

Evaluating the security of a system is only possible if it can be observed and monitored. In communication scenarios, only a fragment of the whole system is under the user control. The rest of the system is controlled by other peers, certificate authorities, communication service providers, identity providers, and public internet service providers or enterprise network administrators. If a complete monitoring is not possible, it may be necessary to make a trust decision. In this section, we define trust and the properties often associated with formal trust models. We then explain the actual WebRTC trust model.

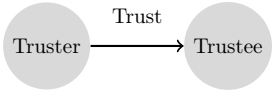
1.4.1 Introduction on Trust

Experienced every day, trust is nonetheless hard to define due to the broad range of situations and interactions it covers. Several definitions of trust have been proposed. Inspired by McKnight and Chervany [14], Jøsang and Presti define trust as: "**the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible**" [15]. In the sense of an action, we would say that trust is the decision of intelligent agents to cooperate in the face of risk and uncertainty about the behaviour (capability or intention) of other agents. This heuristic is often based on prior interaction history, reputation from a community, or recommendations from other trusted sources.

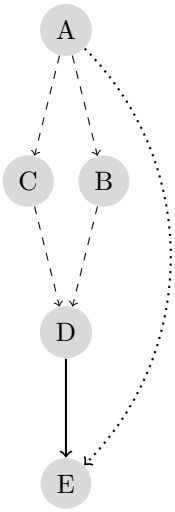
A truster is a person, an organisation, or a software agent while a trustee is an agent, possibly non-intelligent, that can provide a service to the truster. Figure 1.21a presents such a simple trust relationship. Trust between two agents is asymmetric and depends on a specific context, *i. e.* a specific scope of action for which trust is granted. Transitivity is a property commonly associated with trust relationship where an agent D would recommend its trust in E to B (see Figure 1.21b). As capability in a given context does not imply the capability to recommend other agents in the same field of expertise,

transitivity is not present in all models. However, a recommendation could be seen as a context allowing transitivity to other contexts. Indeed, Jøsang and Pope describe a transitive path as a chain of recommending trust terminated by a functional trust [80]. Trust recommendation can also be aggregated from multiple sources. Figure 1.21b shows trust transitivity and aggregation. Dashed arrows represent recommendations, while the dotted arrow represents the trust relation from A to E built from recommendations aggregation.

Figure 1.21: Trust representation



a: Trust relationship



b: Trust transitivity

Methods to establish trust are commonly categorised between policies and reputation based [81]:

- Policies describe the "hard evidence" requirements to obtain trust in a situation where trust in an agent itself is unknown but there is an existing trust in a third party authority.
- The concept of reputation describes the perception of a community on the reliability of an agent part of this community for a particular context. These recommender systems are either organised by a trusted third party or in a peer-to-peer fashion. Furthermore, recommender systems can only establish a reputation in a network where data about agents' behaviour is available.

Trust relationships are dynamic as new feedback and credentials creation or revocation will impact previously existing trust. In some models, recommendations are part of this dynamic as delay can occur in a trust information propagation. Time may also play an important role in trust dynamic, either as an ageing factor determining ponderation of new experience over previous ones or as an eroding factor decreasing trust or the precision of a trust value.

Various trust metrics have been proposed ranging from discrete or fuzzy to continuous scale [81]. We observe two categories of trust measure representations that differ from their representation of distrust. The first category represents trust on a scale of $[-x; +x]$ with the negative value representing distrust, and zero a neutral trust. The second category uses a scale of $[0; x]$ with 0 representing full distrust. In the first case, a bad recommendation by a distrusted source contributes positively to the final trust level, while on the second computational model, the recommendation would simply be discarded due to a low ponderation. Other representations can be considered as a special case of these two categories.

Authors in the literature [82, 83] define operators for trust relationships. In these models, trust transitivity is generally computed with a multiplicative operator while trust sources aggregation is modelled by an averaging operator. Some models may as well represent certainty and distrust. These models handle contradiction in trust recommendation and uncertainty of trust sources.

1.4.2 The WebRTC Trust Model

From a security perspective, trust is often synonym with "providing proofs of identity, authorization, and performance" [81]. Although multiple formal trust models have been proposed and refined in the literature [81], commercial softwares and services often make use of trust models in a simpler manner. The current WebRTC trust model is an informal model categorising entities into three categories: the TCB, authenticated entities, and unauthenticated entities.

At the root of the model is the WebRTC agent, in most cases a browser, which serves as the TCB. All security properties must be guaranteed by the TCB, as we explained in Section 1.2.1. In particular, the TCB is responsible for authenticating other entities acting on the communication.

In WebRTC, authenticated entities are either CS, IdP or other peers. Optimally, these are cryptographically authenticated by presenting certificates via HTTPS for servers, and DTLS or identity assertion for peers. We previously described in Section 1.2.4 how CA build a chain of trust from a root certificate to the final owner's certificate. The trust chain is then anchored in the browser as it is browser makers who configure the list of root certificates in browsers. Similarly, the signalling path forms a transitive recommendation path for media session security certificates. Note that while these entities are authenticated, these may not necessarily be trusted[13]. However, it is mandatory to authenticate entities before making a trust decision. This trust decision is ultimately left for the user to decide.

13: "Just because we can verify that https://www.evil.org/ is owned by Dr Evil does not mean that we can trust Dr Evil to access our camera and microphone" [47].

In particular, while the CS has a large control over the WebRTC session, in some scenarios it may be untrusted. For this reason, the specification introduces an alternative identity path to exchange identity assertions. These assertions bind media session security certificates to an identity and from a trust perspective, this forms an alternative transitive recommendation path. Paradoxically, this identity path is left for the CS to configure and control as it is the CS who sets the IdP to use. From our point of view, it is not clear if the identity path can be trusted independently of the CS.

Other network elements are generally not authenticated by the browser. The WebRTC specification thus assumes that they behave maliciously and the system is supposed to be secure against attacks from these entities.

1.5 Privacy of the Call-Session

Some JavaScript web API may expose critical data to malicious websites. To protect users these resources must be guarded against unwanted access. This is done by enforcing explicit consent policies asking users whenever a website requests access to a sensitive API. Permissions may be granted or denied based and websites origin. HTTP origins, considered insecure, are often limited to one time authorizations. Considerations for privacy in internet standards is steadily increasing[14].

14: The battery status API was even removed after its implementation in browsers as its benefits were estimated less important than the privacy risk faced by users. This API is now only available from privileged code in Firefox [84].

RFC 4949 [85] defines privacy as "the right of an entity (normally a person), acting on its own behalf, to determine the degree to which it will interact with its environment, including the degree to which the entity is willing to share his personal information with others". This definition emerged from the United States Privacy Act of 1974. which was concerned with the growing amount of personal data stored by the U.S. government. In the meantime, the protection of personal data emerged in Sweden, France, and Deutschland, mainly driven by the fear of mass surveillance by an authoritarian state [86]. Privacy should not be used as a synonym for data confidentiality: as a right, privacy is a reason for security rather than a mean. In this section, we describe types of privacy attack and possible mitigation techniques. We also give a brief overview of the legal regulation applying to personal data protection. Finally, we present existing privacy considerations of the WebRTC security architecture.

1.5.1 Attack and Threat Mitigation

The risks associated with privacy threats are multiple [87]. In the most trivial cases, individuals can suffer from the divulgation of their private life by financial or reputation loss. Individuals can also feel a sense of unease by knowing or fearing the monitoring of their actions. This feeling can even lead users to refrain from acting freely. In the worst case, if the attacker is able to physically harm the individual, the individual's life may be endangered. RFC 6973 [87] categorise privacy threats between combined security-privacy and privacy specific threats:

- **Combined security-privacy threats**

  - Surveillance
  - Stored Data Compromise
  - Intrusion
  - Misattribution or Spoofing

- **Privacy specific threats**
  - Correlation
  - Identification
  - Secondary Use
  - Disclosure
  - Exclusion

Particularly related to the context of web real-time communications are the threats of surveillance, misattribution, correlation, and identification. Surveillance is conducted by an attacker who observes or monitors the individual's communications. The threat is present even if the communication is encrypted as the possibility of surveillance can lead the individual to change his behaviour, while the fact that the communication is happening may be enough knowledge for the attacker. Misattribution of behaviour can happen as a result of inadequate or insecure authentication. Similarly, a privacy attacker may also draw wrong conclusions based on his observations of the individual behaviour.

Correlation threats are made possible by the combination of pieces of information regarding an individual. By correlating pieces of information from different sources, an attacker may learn more than what the individual believes to have shared. Finally, identification is the correlation of an individual's information in order to infer or establish the individual's identity. For instance, browser fingerprinting techniques can uniquely identify a browser, and its associated user, by executing a set of JavaScript scripts on a website [88]. Combined together these technical pieces of information can form a unique and portable identifier, without the user knowledge.

The main solution to mitigate privacy threat is data minimisation. As it is difficult to define what information can cause a privacy risk, reducing data disclosure globally reduce the privacy attack surface. Data minimisation is a general principle that should be applied by developers. The most straightforward approach is to limit data collection to the minimum required, however, a limit may also be set to the time of retention of an information. In some cases, individuals may themselves enforce data minimisation by reducing the information they expose to websites. Though this may come as a tradeoff between privacy and functionalities. If the original value of some data is legitimate, denying its usage can protect an individual's privacy but at a cost. For instance, most browser fingerprinting techniques rely on the execution of JavaScript. It is possible to deactivate JavaScript through browser's options but this would break most websites in the process.

An individual is said to be anonymous if it belongs to a set of individuals appearing to have the same attributes. This set is called an anonymity set and individuals within it must be indistinguishable from each other. Anonymity is relative to the point of view of the attacker as it depends on the type of information the attacker can observe. Additionally, the risk of correlation means that it may be difficult to assess if one belongs to an anonymity set or not. A slight difference in attributes collected may de-anonymise an individual or at least reduce the size of the anonymity set. Some privacy preserving approaches thus try to quantify the uniqueness of an attribute.

Respect for privacy recognises the right for an individual to determine to which degree it is willing to interact and share private information with its environment. For

this reason, data collection happening "in secret" can be detrimental to an individual's privacy. User participation approaches involve the individual in the collection process by asking his consent. Though modern privacy regulations enforce a strict policy for explicit user consent (see Section 1.5.2). Depending on the use case and underlying architecture it may be easier or harder to explicitly ask the user for his consent.

Finally, security in itself is an essential component of privacy protection. Confidentiality of exchanges and strong peer authentication ensure that data are protected against a malicious third party.

1.5.2 Regulations

Privacy-preserving regulations have been in development since the 1970's both in the USA and in European countries. Growing concerns by citizens for their privacy on the Internet pushed legislators to adopt more protective legislation. The new European General Data Protection Regulation (GDPR) [89] is a regulation adopted by the European Parliament in 2016 to reinforce and unify data protection for individuals within the European Union (EU).

On one hand, the GDPR simplify the regulatory environment by unifying it across the EU. On the other hand, the requirements for companies or public entities collecting personal data are more significant. At the same time, possible sanctions are also more severe going up to 20 000 000 euros or 4% of the global company revenue. New principles are also introduced such as privacy by design and by default. Privacy by design is a concept developed in 1995 by a joint-team from Canada and the Netherlands. As defined by the GDPR in its article 25, privacy by design means that appropriate technical and organisational measures should be taken both at design time and processing time. The definition of Privacy by default states that appropriate measures should be taken to ensure that only necessary personal data are collected, processed, and stored for each specific purpose of processing. This principle also comes with an obligation of conformity as soon as the design time of a personal data processing system. The responsibility to implement and prove conformity regarding the regulation belongs to a Data Protection Officer appointed by each company or public entity.

Major differences oppose the United States and the EU regarding how their laws protect the privacy of their citizens [86]. In the United States, privacy is a right recognised by the Constitution in order to protect citizens freedom from the government. In addition, privacy-preserving laws are decided at the state level rather than at the federal level. Oppositely, though the GDPR also concerns public entities, the regulation is primarily aimed at the Over The Top (OTT) companies often located in the United States. For users, the GDPR introduces important new rights such as explicit consent, right to be forgotten, right of access, and right of rectification. Furthermore, the GDPR protects EU citizens whether the company processing data is located inside or outside the EU. Such protective regulations from the EU affect companies across the world forcing them to be compliant. For these reasons, both United States and EU laws are led to interact with each other. In a similar fashion, several national legislation across the world have been influenced by European laws.

In the meantime, terrorist attacks are pushing governments to adopt new disposition to facilitate the collection and the exchange of data between security services. While governments reinforce privacy preserving regulations and may invest in privacy-preserving research, communication surveillance is also an ever more important objective for their security services. As an example of these contradictory efforts, the United States government is both funding research on the Tor network [90] and trying to break it through efforts of the National Security Agency [91].

1.5.3 Privacy Considerations for WebRTC

The WebRTC security architecture intends to prevent several types of privacy attacks. Persistent identifiers such as DTLS certificates allow for correlation attacks against anonymous call. To prevent this, WebRTC implementations should allow resetting such identifiers, for instance with a similar lifetime as cookies or through a website-controlled mechanism.

ICE candidates addresses may also form a unique identifier, either by themselves or in conjunction with other web API during browser fingerprinting attacks. Alternatively, ICE candidates may be used to discover the peer's location. Several mechanisms implemented by browsers allow controlling exposed candidates addresses. For instance, users may deactivate the sharing of local candidates or force the use of a TURN relay. The sharing of local candidates may also be bound to a consent mechanism, for instance when requesting `getUserMedia` [92]. This trade-off between privacy and availability means that an optimal network path may not be available for the communication.

1.5.4 Tor: Onion Routing

Anonymity networks protect user's privacy from network surveillance and traffic analysis by anonymising their communications. The most used anonymity network on the Internet is Tor[15] [**INTA_DarkWeb**]. In this section we give an overview of Tor functioning, and explain how it is not practical for WebRTC.

15: `torproject.org`

Tor is an implementation of onion routing whose principle is to build an anonymous routing protocol on top of the Internet. The goal of such protocol is to defeat traffic analysis attacks. The Tor network is constituted of thousands of public routing nodes publishing their public key, and clients. Clients often connect to Tor using Tor Browser, a privacy-preserving browser based on Firefox.

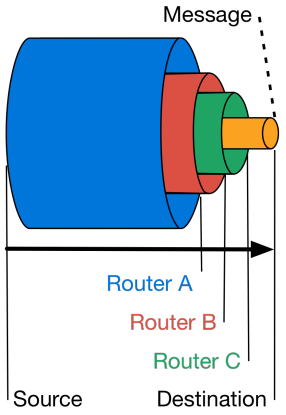


Figure 1.22: In this example, the message is sequentially encrypted with keys from router C, B, and A. It is then routed through these routers starting from A. Each router removes a layer of encryption and routes the message to the next router. Finally, router C transmits the message to the destination. This example is taken from `wikipedia.org`.

In order to establish a communication through Tor, a client randomly chooses nodes to build a routing circuit. In a circuit, each node knows only the previous and next nodes. The entry node (resp. the exit node) knows only the IP of the source client (resp. the destination client). Before sending a packet to the circuit the client encrypts it sequentially with the public key of each nodes part of the circuit, starting from the exit node. This creates layers of encryption around the original packet. To route a Tor packet, each node decrypts the received packet using its own private key, peeling an encryption layer each time. The resulting packet is then sent to the next node in the circuit. When the exit node finally decrypts the packet the original TCP packet is recovered and sent in clear to its destination. An example of onion routing is presented in Figure 1.22.

Tor is particularly useful for avoiding traffic analysis and circumventing internet censorship. This is particularly of interest for journalists, human-rights activists, lawyer, and whistleblowers. However, due to the time consuming layered encryption of each packet and the additional routing overhead, the latency over the Tor network is quite high compared to the standard Internet. Additionally, some JavaScript functionalities or other libraries can reveal privacy-sensitive data. As a result, they are blocked by Tor Browser. For these reasons Tor is not adapted to protect web real-time media communications and WebRTC is actually not implemented in the Tor Browser.

1.6 Signalling Architectures

WebRTC does not mandate any signalling architectures. Silo architectures are the most frequent for web services but do not offer interoperability between CS. In this configuration users are captive of their CS domain. This architecture can be represented in a

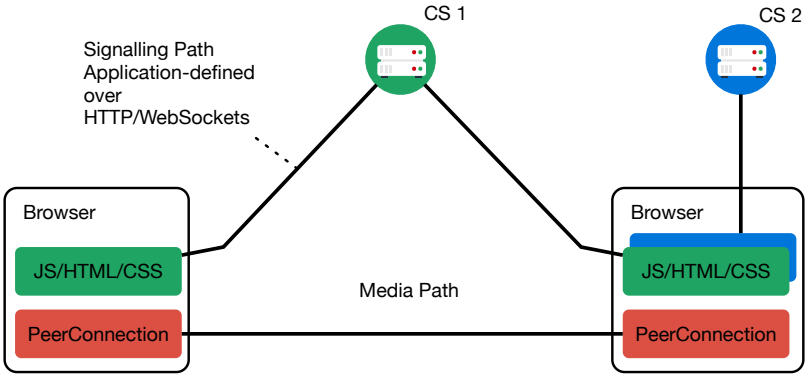


Figure 1.23: In silo architecture users have to connect though the same CS to establish a session.

triangle involving only one CS server, as in Figure 1.23, On the contrary, interoperable architectures allow users of some CS to call users from other CS. They are often represented as a trapezoid architecture with the signalling path involving two CS servers as in Figure 1.1.

Many existing companies offer VoIP communication services, some of which are already switching to WebRTC[16]. The only constraint for signalling is that it uses the SDP offer/answer model. For the moment VoIP communication services on the Web are heavily relying on the silo model. More complex architectures are being developed to allow inter-domain interoperability. These architectures differ by the way they handle user discovery, routing, and identity management and formats. Although the main use case for WebRTC is VoIP, other services such as video streaming [93] and data sharing [94] can be supported. In this section, we give a quick overview of some interoperable and distributed signalling architectures.

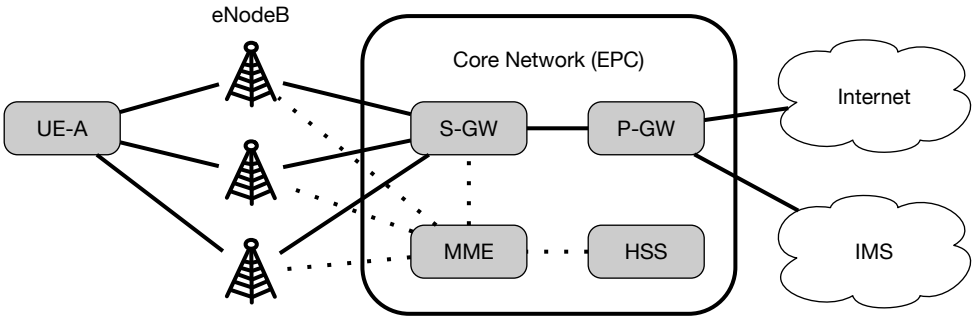
16: Interestingly, browsers makers porting their own communication services to WebRTC aim first at compatibility with their existing solutions and tend to implement non-standard features in their WebRTC implementation. For instance, WebRTC in Chrome supports SDES to cater for hangouts while Edge's first video codec support was H.264 Skype's proprietary codec.

1.6.1 Voice over LTE and WebRTC Interconnectivity

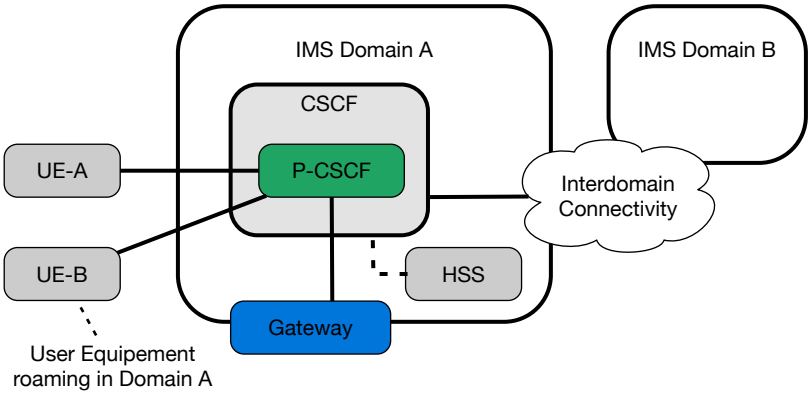
Voice over LTE (VoLTE) is an architecture for VoIP over 4G mobile networks. Figure 1.24a presents a simplified view of the Long Term Evolution (LTE) mobile network, a standard for high-speed mobile networks designed by the 3rd Generation Partnership Project (3GPP). This architecture is separated in an access network, *i. e.* a network of eNodeB responsible for all radio-related functions including encryption of data sent over radio interfaces, and a core network responsible for the control of the User Equipement (UE) and the establishment of IP connectivity. In the core network, the Serving Gateway (S-GW) and Mobility Management Entity (MME) handle the UE mobility, while the PDN Gateway (P-GW) handles IP addresses allocation and Quality of Service (QoS) enforcement. Finally, the Home Subscriber Server (HSS) is a database containing user's profile, location, and IP information. It is responsible for the user authentication, subscription, and discovery. VoLTE interconnects the LTE network with the IP Multimedia Subsystem (IMS) network. The main goal of the IMS is to achieve the concept of Fixed Mobile Convergence[17]. A simplified view of the IMS architecture is presented in Figure 1.24b. Call Session Control Function (CSCF) form the core of an IMS domain. The Proxy-CSCF (P-CSCF) is a SIP proxy server which serves as the point of contact for each user equipment. When a UE connects to a network, it looks up for a reachable P-CSCF. In IMS roaming scenario, the UE connects directly to the visited network's P-CSCF rather than to his home network.

17: The convergence of fixed and mobile telephony networks under an All-IP environment.

SIP is a signalling protocol supporting the determination of user location, availability, as well as session negotiation, setup, and management. As it is only concerned with the signalling part of the communication session, SIP is used in conjunction with other protocols such as SDP and SRTP (presented in Section 1.1). A network of proxy servers

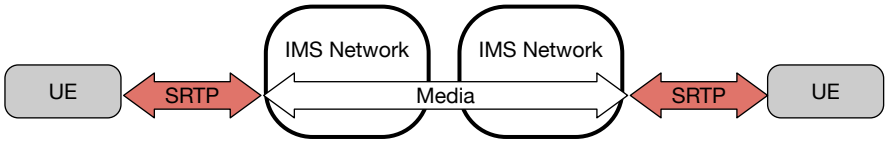


(a) Simplified LTE architecture.

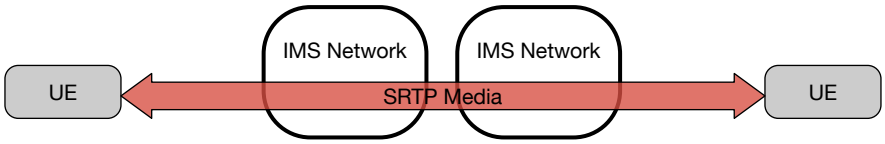


(b) Simplified IMS architecture.

Figure 1.24: A simplified view of the VoLTE architecture. The LTE offers connectivity to the Internet and to the IMS while the IMS offers VoIP service.



(a) End-to-Access-Edge Encryption



(b) End-to-End Encryption.

Figure 1.25: End-to-End and End-to-Access-Edge encryption scenario over IMS networks.

allows SIP endpoints to register and routes requests to a user's location. In VoLTE, the UE contains a SIP user-agent and an Universal Integrated Circuit Card (UICC) often refered to as a "SIM Card". The UICC contains several identity modules including the IP Multimedia Services Identity Module (ISIM) which in particular provides the IMS home domain name, a private and public identity[18], and a secret key used for authentication and registration.

18: The public identity is a telephone URI of the form `tel:phonenumber`.

VoLTE offers roaming, handover, and interconnectivity with other networks. Interconnectivity with WebRTC can be handled by an IMS domain offering a compatible gateway (see Figure 1.24b). In this case, the IMS domain has to deploy a WebRTC web server, a WebRTC JavaScript client, and enhanced functionalities for WebRTC in the P-CSCF. The gateway performs adaptation between WebRTC and IMS protocols such as transcoding and encryption. While encryption using SRTP profile for DTLS is mandatory in WebRTC (see Section 1.3), encryption of both media and signalling is optional in the IMS. Furthermore, as VoLTE security relies on the radio encryption of the access network, its RTP profile does not support encryption. This constraint implies that WebRTC-VoLTE interconnectivity cannot use End-to-End encryption with SRTP and is instead limited to End-to-Access-Edge encryption (see Figure 1.25).

1.6.2 Matrix

Matrix [25] is a specification for a messaging and data synchronisation federation. One of its use cases is the exchange of signalling message for WebRTC communications. Figure 1.26 shows three users, each connected to a Matrix home server, and exchanging synchronised message in a room.

Matrix specifies several API, amongst which a client-server and a server-server API. The server-server API defines the HTTP interfaces and data format allowing Matrix home servers to synchronise messages with each other in real-time. Servers can either push a message to another server, broadcast it to all servers in a room, or make a query to a server to get a state snapshot. This API also defines how servers are discovered and authenticated, and how user's presence information is exchanged between home servers. The client-server API defines the HTTP interfaces for a client application to connect and communicate with a Matrix home server. It allows the application to manage rooms and send messages. The client-server API also defines how users authenticate to their home server through any Matrix application. This particularity of the Matrix architecture means that users can choose any application to connect to any home server.

Matrix user identifiers are of the form `@UserID:HomeServerDomain`. These identifiers are used internally by home servers, and in particular for home server discovery. However, Matrix also defines third-party identifiers (3PID) as a more user-friendly way

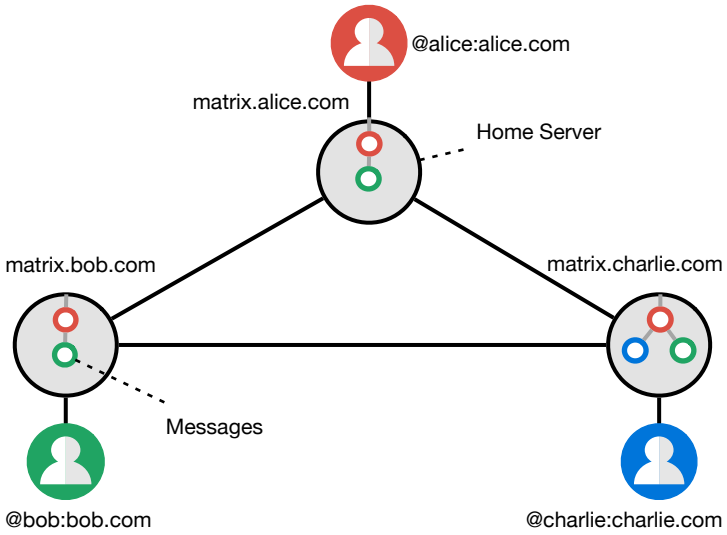


Figure 1.26: Federated messaging and data synchronisation between three Matrix home servers.

to refer to users. The Identity Service API defines how 3PID are associated to Matrix identifiers, stored, and queried. This service relies on a trusted federation of Matrix identity servers.

Despite its interesting concepts, the Matrix ecosystem is still in early development. Most implementations are in the alpha stage and the specification itself is rated as unstable. The Matrix identity specification is the less advanced specification, and no identity server implementation is referenced on the Matrix webpage.

1.6.3 reThink

reThink, an H2020 European project, specified and experimented with a real-time communication framework. Its global objective is to allow Telcos to compete with large OTT companies by offering open and interoperable communication services and applications.

A simplified view of the reThink architecture is presented in Figure 1.27. It builds on the earlier signalling on-the-fly concept from the Wonder European project[19]. Signalling on-the-fly avoids the standardisation of messaging protocols by providing a messaging API. When two parties want to communicate, they agree on a protocol and download the corresponding protocol-stub implementing the messaging API. While this architecture allows interoperability without standards protocols, it still requires standardisation of the messaging API and message format used for communication.

19: `http://hypercomm.github.io/wonder/`

On the user's side, reThink applications are executed inside the runtime. The runtime can be a modified web browser or a native program[20]. reThink applications make use of modular software components, called hyperties. These hyperties are downloaded from service provider catalogues and executed in sandboxed environments. Hyperties can provide many services to an application, including P2P communications. In the interoperable reThink architecture, users can connect to each other through different service providers. Signalling between two hyperties is thus established through one users' service provider. More specifically through a messaging node exposed by the selected service provider. In order to establish the connection, both runtimes agree on available protocols and download the corresponding proto-stub from the chosen service provider's catalogue. This process allows hyperties to set up their session and eventually open a P2P communication.

20: The runtime was initially envisioned as a web browser modification. However, it was implemented as a JavaScript library due to the complexity and costs involved.

Identity management in the reThink architecture is handled by the Identity Module

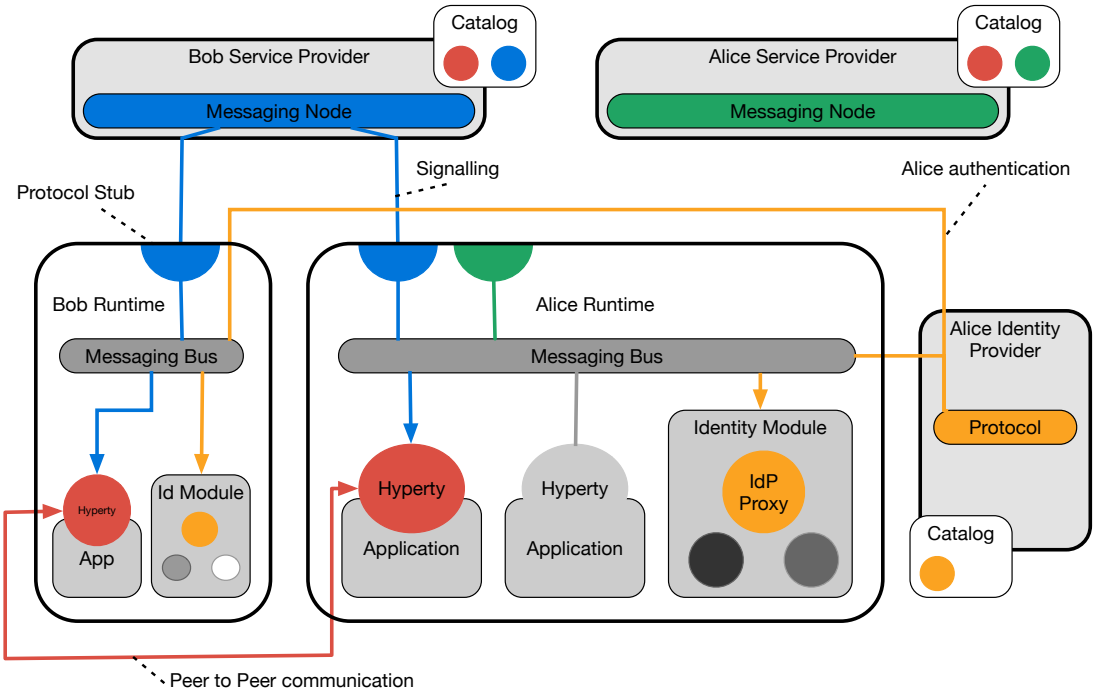


Figure 1.27: Simplified reThink architecture

and reuses the concept of the WebRTC IdP Proxy. Some of its design objectives in addition to the WebRTC compatibility are the ability for users to choose their IdP and the availability of discovery and presence services. In particular, reThink discovery, presented in Figure 1.28, is a layered architecture relying on a cryptographic Globally Unique IDentifier (GUID). The global registry, a trusted distributed hash table run by a consortium of service providers, maps GUID to user profile information. Users can manage and control their public profile information stored in the Distributed Hash Table (DHT) by proving ownership of a private key. These user profiles also point to user identities on service providers by exposing pairs of service provider domain and service bound user identifiers. Services providers, in turn, expose a domain registry containing the user's reachability information. GUID can either be exchanged manually or discovered through the use of a discovery service.

1.6.4 Distributed Signalling Architectures

The rapid growth of web traffic and video streaming, in particular, has an important impact on infrastructure costs both for content providers and network operators. Content Delivery Network (CDN) are constituted of proxy servers geographically distributed to be as close to clients as possible. Besides tidying core network traffic, CDN optimise the availability and performance of content delivery through various techniques and in particular caching, load-balancing, and routing techniques. Distributed CDN merge the advantages of CDN and P2P networks. These networks leverage their clients' upload bandwidth to redistribute content to other clients and further reduce costs. However, distributed CDN require the installation of client-side software or browser plugin [95].

WebRTC allows browser-to-browser P2P communication for video streaming and data channels without third-party plugins. It is thus possible to build distributed CDN around a network of WebRTC enabled browser. To create such network, clients first

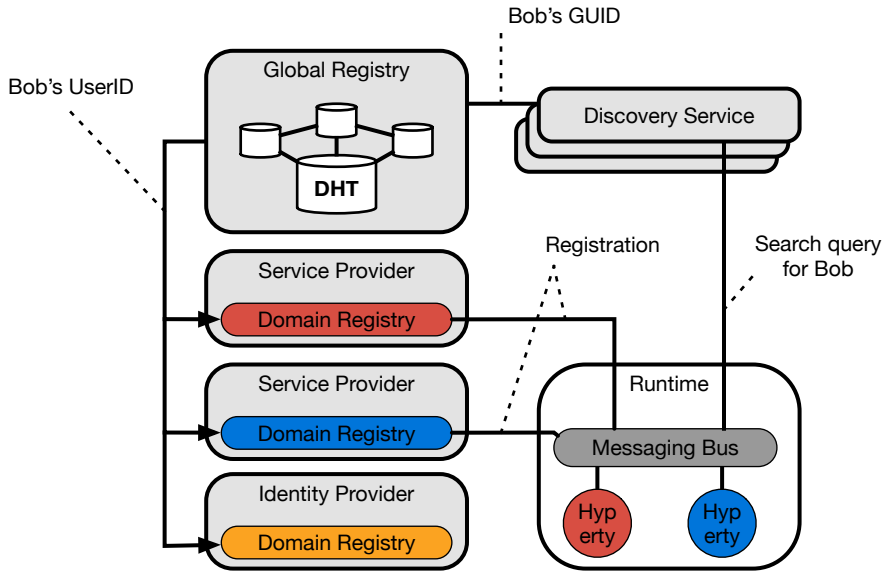


Figure 1.28: reThink Identity Discovery

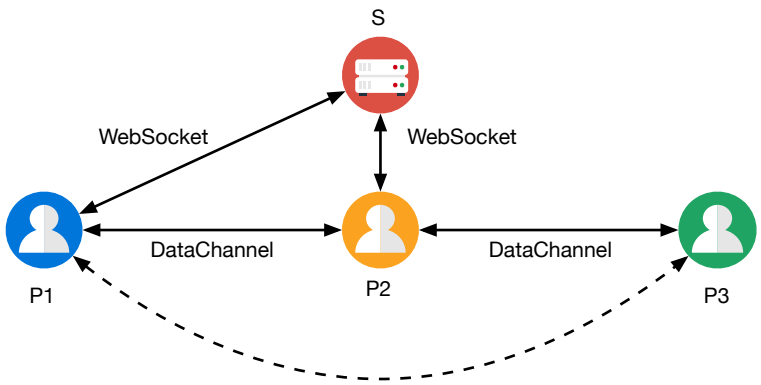


Figure 1.29: In this example P1 and P2 are already part of the network. To join the network P3 first reaches the bootstrapping server S. S signals P2 and P3 so that they open a P2P data channel. P3 then requests a video stream from P1. SDP messages between P3 and P1 are signalled through P2 using existing data channels. Finally, the video stream is established from P1 to P3 [97].

reach a bootstrapping server. This server allows new clients to open data channels with other clients already part of the network. These data channels form an overlay signalling network. Once a client joined the network, it can discover other distant clients and establish P2P connections with them, for instance, to retrieve a file or a video stream. Figure 1.29 shows an example of such network architecture where P3 joins the network and finally retrieves a video stream from P1. As the signalling message from P3 to P1 goes through P2, the signalling architecture is similar to MitM attack as shown in Figure 1.13. Such signalling path may not be trusted and depending on the use case it may be an issue. Zhang *et al.* [96] also report some security and privacy considerations for their WebRTC CDN framework.

1.7 Summary

The WebRTC peer-to-peer media path is negotiated by peers exchanging SDP offer/answer messages over the signalling path. Media path encryption is mandatory and is mainly built around the DTLS and SRTP protocols. It ensures confidentiality and integrity of the P2P sessions between peers. The signalling path generally involves one or more signalling server and its security relies on standard web security protocols. At the root of these protocols is the web browser which acts as the trusted computing base. WebRTC applications and their associated signalling servers are also responsible to protect the availability of the session and network resources by filtering incoming call.

Depending on the signalling architecture or the signalling servers' origin the signalling path may be untrusted by users. WebRTC remedies to this issue by allowing peers to bind DTLS certificates to identity assertions. These assertions are generated and verified by identity providers forming two symmetric identity paths. Paradoxically, these identity paths are configured by the communication services.

Privacy is a growing concern for users and contributors of web specifications alike. The main privacy threats against WebRTC users are related to identification of anonymous callers and correlation of web browsing and call history. Browsers enforce consent policies to protect users against malicious sites launching non-legitimate calls. However, legitimate communication services also gain an important knowledge of their users' call history.

Chapter 2

State of the Art

In this chapter, we present the state of the art on VoIP security research. In Section 2.1 we present the results of a survey reviewing 245 articles on VoIP security and published in 2012 by Keromytis [98]. As the first drafts of WebRTC specifications were published the same year, this survey is a solid starting point to understand the field of VoIP security. In Section 2.2, we present our own survey of VoIP and WebRTC security research papers published between 2012 and 2017. We follow a similar methodology as used by Keromytis and collect and classify 208 papers. We then review the papers dealing specifically with WebRTC.

2.1 VoIP Security Research - 2012

WebRTC is in 2018 still a young technology. The World Wide Web Consortium (W3C) WebRTC Working Group was created in May 2011 and the first version of the WebRTC Security Architecture draft was published in January 2012. The same year, Keromytis published "A Comprehensive Survey of Voice over IP Security Research" [98] which reviews 245 articles related to fields of Voice over IP (VoIP) security. This survey is a starting point for understanding the field of VoIP security research just before the introduction of WebRTC. Indeed, WebRTC contributors built on the same accumulated experience when making design and implementation decisions.

In this section, we summarise Keromytis' survey and categorisation of VoIP research topics. We also give additional references from our personal knowledge when it seems relevant. We particularly focus on research that could be applied in the context of WebRTC communication. Note that a large part of the security research on VoIP is focused on the network side such as the Session Initiation Protocol (SIP) signalling protocol and SIP-based architectures. While WebRTC does not mandate a particular signalling architecture, SIP-based researches are still relevant to WebRTC. Actually, SIP could be used on the signalling path, but more importantly, most vulnerabilities and defence mechanisms related to SIP are still relevant for ad-hoc signalling protocols.

2.1.1 Threats Classification and Methodology

To build the survey, Keromytis initially collected papers from personal knowledge, searches on online library[1], and browsing of proceedings of top security conferences, journals, and specific workshops[2]. In particular, searches were conducted with the following keywords: "VoIP security", "VoIP vulnerabilities", "VoIP attacks", "SIP security", "SIP vulnerabilities", and "SIP attacks". The collection was then expanded by browsing

1: CiteSeer, IEEE Xplore, ACM Digital Library, and Google Scholar.

2: IEEE Security & Privacy Symposium, ISOC Symposium on Network and Distributed Systems Security, ACM Computer and Communications Security, USENIX Security, RAID, ACM Transactions on Information and Systems Security, and IEEE Transactions on Dependable and Secure Computing.

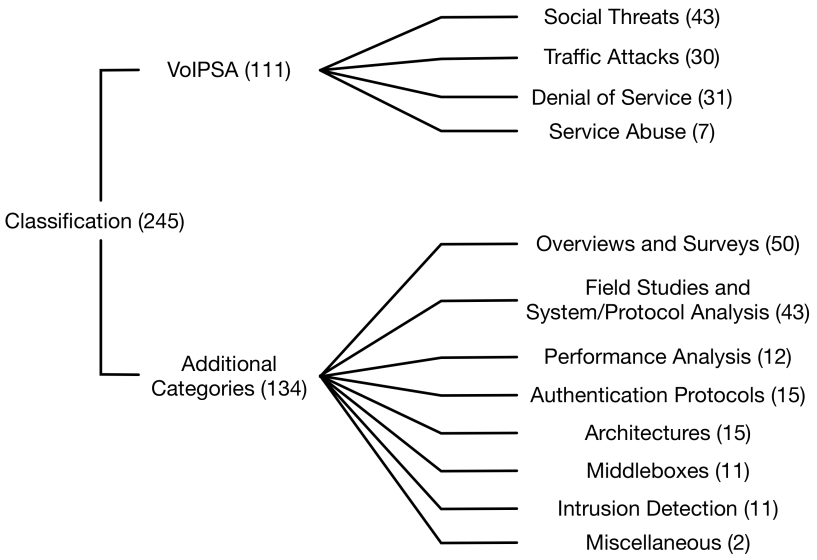


Figure 2.1: Classification tree [98]

the proceedings of conferences in which these papers appeared and searching for other VoIP security papers by the same authors. The process was iterated over until no new papers were added to the collection.

These papers were then manually classified according to an extended version of the VoIP Security Alliance (VoIPSA) threat taxonomy [99]. The considered VoIPSA threat classes are the following: social threats, traffic attacks, denial of service, and service abuse. In addition, eight additional classes were considered: overviews and surveys, field studies and system/protocol analysis, performance analysis, authentication protocols, architectures, middleboxes, intrusion detection, and miscellaneous. Figure 2.1 presents the repartition of surveyed papers in these classes.

2.1.2 Keromytis Survey Summary

Social Threats

Social Threats represent the attacks aimed at the human users rather than at the software systems. Such attacks are for instance unwanted contacts misrepresenting the identity of a malicious calling party or bypassing opt-out consent. In practice, research on social threats is mostly focused on defence against SPam over Internet Telephony (SPIT) call which we presented in Section 1.3.2.

According to the VoIPSA, the classification of call as spam is subjective. Indeed, SPIT call may be lawful solicitations and become spam only after bypassing refused consent. The bulk of defence against SPIT thus focus on getting user's input. Caller classification may follow a simple binary approach into whitelists or blacklists. However, more complex approaches propose to use reputation-based models and social relations between users to assign a trust value to an incoming call. In these systems, the intelligence of SPIT detection algorithms is located at the endpoints and allow users to manage their own policies for SPIT rejection.

Other approaches place SPIT detection at the network level, *i. e.* on the signalling path. These systems measure various criteria such as the number of incoming and outgoing calls, call duration, and call history. Deviation from standard long-term expected average may reveal a spam call. These analyses may further be refined by user input or filtered with a Turing test[3] presented to the caller. Some authors also propose to apply

3: Turing tests tries to distinguish computers from humans.

fingerprinting techniques, either targeting SIP messages format or the audio data of incoming VoIP calls. SPIT calls would be detected by the presence of unique fingerprint over a large number of different calls. Sorge *et al.* [100] propose to evaluate the reputation of Communication Service (CS) and their SPIT detection algorithms, providing incentives to honest CS to correctly tag outgoing calls.

Trust and reputation models may easily be bypassed under weak caller authentication. Some authors thus propose to enforce strong caller identity verification. In particular, Srivasta *et al.* [101] propose to consider the caller origin domain and the confidence level in the authentication performed while Croft *et al.* [102] propose to include a Verifying Authority into the call setup. This Verifying Authority would be responsible for applying policies in particular based on the caller identity before transmitting the call to the user.

Traffic Attacks

The traffic attacks and defences class is concerned with the risks of eavesdropping, interception, and modification of signalling data and media sessions. These attacks may either target unencrypted sessions or bypass cryptographically protected sessions. This field of research, however, does not consider generic cryptography research but only focus on specific VoIP cases.

Indeed, the specific nature of VoIP traffic, transmitting voice in stream sessions, may allow specific attacks. Some researchers thus propose to use packet delay variation (also referred to as jitter) in order to de-anonymise VoIP streams or introduce covert channel. Other researchers apply machine learning techniques to determine the language spoken or identify blank, phrases, or even words in encrypted voice streams. As summarised in Figure 2.2, these attacks use packet size variation of Variable Bit-Rate (VBR) human-speech codecs. VBR codecs, as opposed to Constant Bit Rate codecs, use variable packet size depending on factors such as encoded phonemes, blank in speech, or encoding quality. As VoIP is a time-sensitive application, delay or jitter may have a dramatic impact on the availability of the communication. Thus reducing bandwidth usage and adapting to network conditions is a critical feature for VoIP usability. However, these researches reveal a trade-off between security and availability/quality of VoIP communications. Request For Comments (RFC) 6562 [103] gives guidelines on the usage of VBR codecs in conjunctions with Secure Real-time Transport Protocol (SRTP) sessions.

Traffic analysis and signalling data can be used to compromise users' privacy. Some researchers study approaches based on anonymisation networks such as Tor (see Section 1.5.4) and traffic padding techniques [105, 106]. Srivasta *et al.* [107] study the issue of Quality of Service (QoS)-sensitive routes in anonymising networks. To this day, QoS is still an issue for VoIP over anonymisation networks and researchers are still exploring possible solutions [108].

A number of researches suppose unencrypted signalling paths or media session or focus on alternative key agreement protocols such as ZRTP. As WebRTC mandates the use of Datagram Transport Layer Security (DTLS) and SRTP for media and data sessions, and browsers enforce the use of Transport Layer Security (TLS) for signalling session, we do not consider these results as relevant to our research.

Denial of Service

We presented Denial of Service (DoS) attacks in Section 1.3.2. While DoS attacks may target either endpoints or networks elements, researches in this field mostly focus on attack and detection at the network level. DoS flooding attacks, schematised in Figure 2.3, have similarities to SPIT calls meaning that these can be detected by monitoring deviation from standard call frequencies or error rate. Legitimate situations such

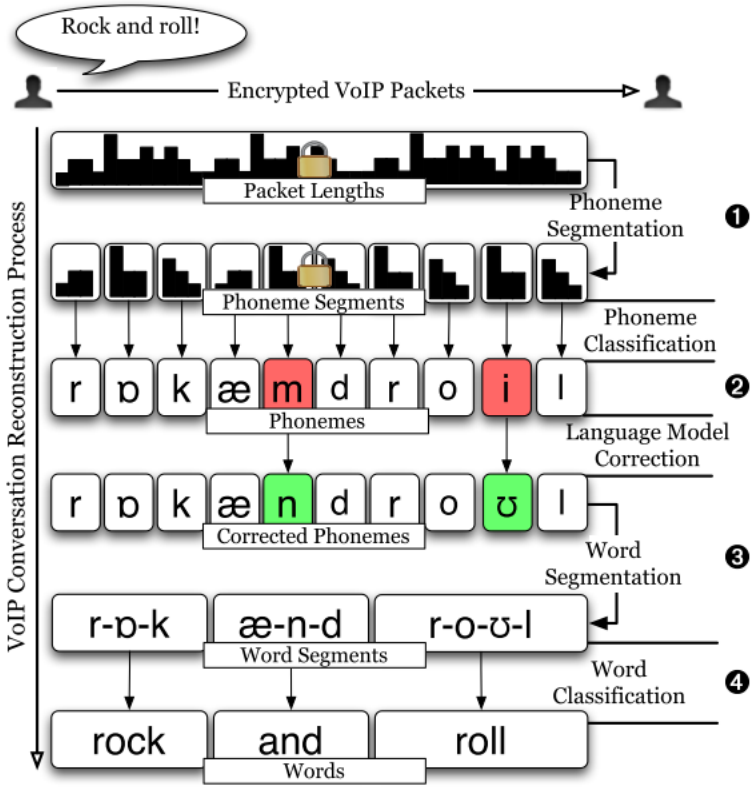


Figure 2.2: Overall architecture of an approach for reconstructing transcripts of VoIP conversations from sequences of encrypted packet sizes [104].

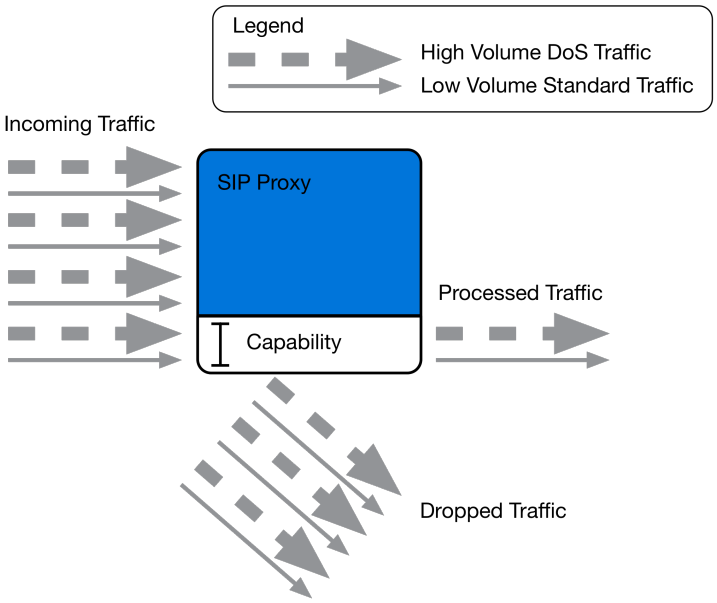


Figure 2.3: Schematic overview of a DoS flooding attack. Due to the server's limited processing capabilities a lot of regular requests cannot be processed if a high load of malicious messages are targeted towards the server [112].

as emergencies may also conduct to flooding scenarios. Some researchers thus try to differentiate DoS attacks from flash crowds situations [109]. However, the main difference with SPIT is that DoS attacks aim at shutting down the VoIP network. It is crucial for detection algorithms to introduce as few overheads as possible during call setup.

Indeed, other DoS attacks directly aim at flooding the memory or CPU of network elements and in particular SIP servers. While strong message authentication is proposed as a solution against DoS attacks, some researchers show that authentication may have a negative impact in some DoS scenarios [110]. Whitelisting or lightweight authentication based on call history from regular caller's identity or Internet Protocol (IP) adress, actually similar to some trust models, are proposed to mitigate attacks. Other mitigation techniques also propose to adapt server procedures during DoS scenarios, for instance, dropping ringing call earlier when being overloaded [111].

Payload and flow tampering attacks target specific signalling protocols, respectively to crash servers or abort sessions. Most research on these attacks focus on SIP and protection mechanisms are well-known [112] including tools to check SIP implementations. Additionally, encryption on the signalling path deters flow tampering attacks.

Service Abuse

Service abuse threats are related to the improper use of VoIP services especially in commercial services, for instance, to increase or avoid billing. The research in this area is quite limited compared to other threat classes as it features only 7 references. This can be explained by the specificity of architectures concerned by service abuse threats. Billing is closely associated with authentication and authorization of users. Some described attacks use a SIP protocol vulnerabilities revealed by formal verification to forge SIP messages impersonating users. A solution to fraud and proposed by Geneiatakis *et al.* [113] is to let an Authentication, Authorization, and Accounting (AAA) server sign SIP messages after the user authenticated to the server, hence providing authenticity and non-repudiation for signalling messages. Although the solution is based on Telco rather than web protocols, it shows similarity to the introduction of a third-party Identity Provider (IdP) in WebRTC communication setup.

Field Studies and System/Protocol Analysis

Researches in this category focus on analysing protocols, implementations, and deployed systems using various techniques in order to find security vulnerabilities and flaws. Techniques used include formal verification [114], fuzzy testing [115], as well as black-box [116] or source code analysis [117]. Similar analyses are also conducted in paper classified in other categories.

Formal verification techniques ensure that supposing a defined attacker model, the attacker cannot learn compromising information. Such an attacker model is the Dolev-Yao [118] model in which the attacker can listen to any message on the network, build arbitrary messages from known information, and send them over to the network. The AVISPA project[4], for Automated Validation of Internet Security Protocols and Applications, is a suite of tools for formal modelling and verification of security protocols. In particular, the project offers a library of protocol models in the High-Level Protocol Specification Language, some with known and demonstrated attacks.

4: avispa-project.org

Formal verification should not hide the fact that actual implementations may present faults and weaknesses. These faults may be due to weak specification, error in the implementation, or use of default configurations. For instance the Heartbleed bug[5] allowed an attacker to reveal the memory of an OpenSSL protected system, the most popular TLS implementation, by using a missing bound-check in the handling of the TLS heartbeat[6].

5: heartbleed.com

6: The actual fix shows how a small implementation error can have dramatic consequences: https://git.openssl.org/gitweb/?p=openssl.git;a=commitdiff;h=96db902

A number of surveyed work resulted in vulnerability disclosure publications in databases such as the Common Vulnerabilities and Exposure (CVE) database. Such database allows the rapid dissemination of vulnerability disclosures and fixes to organisations using security software.

Performance Analysis

Performance analysis works focus on evaluating the impact of security protocols on both call setup and media sessions. We already mentioned the trade-offs between security and availability/quality of the communications. Papers in this category try to precisely measure this trade-off.

Results globally show that the main overhead is due to the asymmetric encryption of signalling messages while the symmetric encryption of the media session only produces a small overhead. The exact figures vary depending on the actual protocol and configurations being compared. For instance, a prototype demonstrates an improvement from a factor between 2 and 8 in handled call setup requests per second by using the Elliptic Curve DSA algorithm instead of RSA.

Other researches, not referenced in the survey, evaluate the strength of cryptographic protocols. In practice, perfect security is highly impracticable and only imperfect security can be achieved [119]. Modern cryptography thus aims for a high enough security given reasonable computation power. Guessing attacks (or brute-force) form an upper-bound to the amount of computation required to break an encryption. Against some encryption schemes, faster techniques than exhaustive search can be used. To estimate the difficulty of an attack against these encryptions it is necessary to factor computation time and cost in the equation. For instance, Kleinjung et al. [120] use Amazon cloud public prices to compare the security level of current algorithms with the level of the DES in 1980, as proposed by Lenstra [121]. As the computing power increases and computing cost decreases over time these estimations must be updated regularly. National security agencies [122, 123] also give recommendations on security algorithm implementations and usages to achieve reasonable security properties in a given timeframe.

Alia *et al.* [124] propose a component-based adaptation model to manage the trade-offs between QoS and Security. They model the adaptable VoIP system as a composition

of components each providing different QoS and security properties. A utility function aggregating QoS and security dimensions, shown in Figure 2.4, allows discriminating between different configurations. Considered dimensions are the latency and video scheme quality for QoS and the confidentiality, anonymity, and authentication for security. Their model also uses user's preferences as weight in the utility function and risk context as minimal required value for each security dimensions.

$$U = W^{lat}.F(lat) + W^{qua}.F(qua) + W^{conf}.F(conf) + W^{anon}.F(anon) + W^{auth}.F(auth)$$

Figure 2.4: Overall utility function [124] with $F(k)$ the utility functions and W^k the user preference weights for dimensions k as latency, video scheme quality, confidentiality, anonymity, and authentication.

Authentication Protocols

The SIP authentication mechanism is based on HyperText Transfer Protocol (HTTP) digest authentication [125] and allows any SIP proxy or SIP user-agent to issue an authentication challenge when receiving a request. The response to the challenge is basically a hash of some information including a nonce, *i. e.* a unique random number associated with the request, and a password. The response also includes a username. Upon receiving the response, the entity which issued the challenge looks up the password corresponding to the submitted username. It can then perform the same digest operation and compare the result to the given digest response to validate it.

Researchers working on authentication protocols mainly propose extensions or variants to SIP authentication and are mostly focused on VoIP as a use case for cryptography. Interestingly, an article published by Cao and Jennings [126] in 2006 deals with the issue of end-to-end user identity in VoIP call establishment. One of their assumptions is that using TLS over each signalling hop is unrealistic[7], thus breaking the necessary chain of trust. In 2017 this assumption cannot be considered valid anymore and the WebRTC security architecture recommends for the signalling path to be secured by TLS[8].

7: The authors explains this as to be "because of some difficulties and other reasons for deploying TLS" [126].

8: "[The signalling] message is sent to the signalling server, e.g., by XMLHttpRequest or by WebSockets preferably over TLS" [47].

Other categories

Other research surveyed by Keromytis are categorised as middleboxes, architectures, and intrusion detection aiming at various threats but "not easily classified in any of the previous categories". Middleboxes are network devices manipulating traffic for other purposes than packet forwarding. Researches on VoIP middleboxes thus focus on the traversal and operation of firewall and gateways to other networks. Architecture and intrusion detection researches mainly focus on the detection of anomalies in VoIP networks, either from malicious or non-adversarial causes and related defences.

Observations

According to Keromytis observations, almost 20% of the surveyed publications offer an overview of VoIP security problems and solutions. He also observes that over 15% of the work is coming from the cryptographic community, either to increase security or performance and that roughly 20% of researches are dedicated to addressing SPIT. While he remarks that SPIT is not currently an issue for VoIP, he adds that prior and current experiences in email and telemarketing spams should be sufficient motivations to continue researches against SPIT.

Comparatively, he observes that the problem of DoS is less studied and that researches on the subject focus on the network side of things. In previous surveys on the CVE database, Keromytis reported a majority of SIP-specific DoS vulnerabilities with half of the DoS vulnerabilities present at the endpoint. He thus argues for more research targeting this problem, especially looking at strengthening implementations and not addressing the problem from a black-box approach. Finally, Keromytis also argue for more work addressing cross-protocol and cross-implementation problems.

2.2 VoIP and WebRTC Security Research - 2012+

To complete our state of the art we survey and categorise VoIP research and published since 2012. We first present our methodology for collecting papers. We then give a rough overview of the repartition of VoIP research since 2012 by classifying collected papers. Finally, we review collected papers dealing specifically with WebRTC.

2.2.1 Methodology

To build our survey we first look at collecting papers related to VoIP security research and published between 2012 and 2017. We use the same keywords as Keromytis [98], that is: "VoIP security", "VoIP vulnerabilities", "VoIP attacks", "SIP security", "SIP vulnerabilities", and "SIP attacks". As WebRTC was introduced in 2012 and is the focus of our research, we also look for papers specifically targeting WebRTC. To this end we use WebRTC as an additional keywords, *i. e.* "WebRTC security", "WebRTC vulnerabilities", "WebRTC attacks". The search is finally conducted on Google Scholars search engine and using the search strings presented in Figure 2.5.

Figure 2.5: Paper collection search strings used on Scholar.

"VoIP OR SIP OR WebRTC Security OR Vulnerabilities OR Attacks "
"WebRTC Security OR Vulnerabilities OR Attacks "

Google Scholar indicates 27 800 results for the first search string and 2 890 results for the second string. We crawl these results, ordered by relevance until we estimate that proposed papers are not relevant anymore. Paper selection is done based on title and abstract, and ultimately our paper collection on VoIP research returns 208 results. We do not consider non-peer reviewed papers. Relevant RFC are presented in the background Chapter 1 on WebRTC trust and security architecture.

2.2.2 Observations on VoIP Security Research since 2012

We roughly classify our collection of 208 papers, based on title and abstract, into the same categories as presented in Section 2.1.1. This allows to compare the proportion of results for both periods and get a picture of the repartition of researches. Table 2.1 shows the classification of collected papers and compare them with the repartition of paper collected by Keromytis.

According to our classification, we observe some significant changes ($> +/- 5\%$) in the repartition of research. Firstly, we observe a drop in the proportion of research focusing on social threats by 11% for the period since 2012. Keromytis remarked that most of the social threat researches were focused on SPIT mitigation, although it was not an issue in VoIP yet. This fact may explain the decrease in research for this threat category. Researches focused on traffic attack also decreased by more than 7% on the same period. In particular, we do not observe any research related to traffic attack since 2016 and we classify only one 2015 paper as traffic attack related. Conversely, the

Category	-/2012	2012/2017	diff	WebRTC
Denial of Service	12.6% (31)	16.4% (33)	+3.8%	0
Service Abuse	2.9% (7)	5% (10)	+2.1%	0
Social Threats	17.5% (43)	6.5% (13)	−11%	1
Traffic Attacks	12.2% (30)	5% (10)	−7.3%	2
Overviews and Surveys	20.4% (50)	15.9% (30)	−4.5%	7
Field Studies and System/Protocol Analysis	4.9% (12)	8.9% (18)	+4%	2
Performance Analysis	5.7% (14)	7.5% (15)	+1.8%	0
Authentication Protocols	6.1% (15)	16.4% (33)	+10.3%	1
Architectures	7.7% (19)	10.4% (21)	+2.7%	8
Middleboxes	4.5% (11)	1% (2)	−3.5%	2
Intrusion Detection	4.5% (11)	4.5% (9)	−	0
Miscellaneous	0.8% (2)	2.5% (5)	+1.7%	0
Total	100% (245)	100% (201)		23

Table 2.1: Classification of VoIP security papers returned by our search.

authentication protocol category of research sees an increase of more than 10%. In particular, we collected multiple papers applying elliptic-curve cryptography to VoIP while in Keromytis's survey only two references are given. Note that some of these differences may be due to the way we collected and classified papers compared to Keromytis process.

We then looked for references to WebRTC in surveyed papers. We extracted a list of authors of these WebRTC security papers and looked for any missing publications using DBLP[9], revealing two additional overview papers. Unsurprisingly, the categories of Denial of Service, Service Abuse, and Intrusion Detection do not contain WebRTC related research. Such attacks are generally targeted against the service architecture which is not the focus of the WebRTC specification. Similarly, the social threats and traffic attacks categories only contain one and two WebRTC related paper respectively. While WebRTC mandates or recommends the use of some security mechanisms on the signalling and media paths security, researches in these areas are not specific to WebRTC and may target out-of-scope protocols such as SIP. Surprisingly, although WebRTC does not specify any signalling architecture we observe several WebRTC papers in the architecture and middleboxes categories. A large proportion of these papers are dealing with issue of integrating WebRTC services inside enterprise environment and existing VoIP infrastructures.

9: `dblp.uni-trier.de`

2.2.3 Survey of WebRTC Security Research

Traffic Attacks (2)

In their 2015 articles, Mauro and Longo [127, 128] use machine learning techniques to identify encrypted WebRTC traffic. Used classifier algorithms are configured to consider the inter-arrival times, packet lengths, and the number of packets received and sent. They implemented a detection system and tested it with three then four classification algorithms. Their results are however limited in significance due to the small and artificial test sample.

Overviews and Surveys (6)

In 2013, one year after the first WebRTC drafts, Jennings *et al.* [129] published an overview of the WebRTC architecture and its design principle. They present the security and identity architecture, in particular, mentioning that their approach aims at allowing users to "use their preferred identity provider and logs on to the provider in whatever way that provider uses". A similar overview paper is published in 2014 by Barnes and Thomson [130] this time focusing on the security and identity architecture exclusively.

Loreto and Romano published in 2012 [131] an overview of the ongoing efforts for WebRTC specifications in which they discuss some security considerations. In July 2017 they published an overview of the remaining efforts towards WebRTC 1.0 [132].

Rahaman published an overview on WebRTC security in 2015 [133]. After presenting the WebRTC security architecture, examples of trusted third-parties IdP are provided: Google, Facebook, LinkedIn, as well as the Browser Id and WebFinger protocols. Rahaman also lists concerns for WebRTC security including the inheritance of VoIP attacks through gateways and the security of third-party IdP. No details are however given on particular attacks. Issues of gateway implementation to integrate WebRTC service with SIP-based systems are also the subject of a 2013 paper by Amirante *et al.* [134].

10: The STREWS project was a European research project running between 2012 and 2015.

The Strategic Research Roadmap for European Web Security project's (STREWS)[10] major contribution is a technical state of practice document for web security [135]. The project's studied methodology targets new aspects added to the web ecosystem in parallel with the standardisation and deployment of the technology bringing these aspects. Following this methodology, the project published a security case study report on WebRTC [136] as it was deemed a "security sensitive extension to the Web". This document identifies six assets related to WebRTC and describes new threats. These assets are the browser, the client machine, the server machine, the client-side application code, the identity provider's infrastructure, and Session Traversal Utilities for NAT (STUN)/Traversal Using Relays around NAT (TURN) servers. The threats described cover a large scope including some DoS attacks, service abuse attacks, Man-in-the-Middle (MitM) attacks, and privacy attacks. In the second part of the document, a few areas are studied in-depth and new attacks and vulnerabilities are described including:

- In additions to MitM attacks, malicious web applications can also redirect both streams to an attacker either by accessing streams directly from JavaScript or by taking screenshots using the HTML canvas elements containing the video streams.
- The central position of IdP means that they can be used as a meta-data capture service, for instance, to allow legal pervasive monitoring or user profiling.
- The WebRTC identity architecture allows the disclosure of precise user identity information to malicious web applications.
- The web certificate infrastructure does not have an effective scoping of Certification Authority (CA)'s authority. This issue extends to IdP handling caller authentication in many WebRTC scenarios.

Field Studies and System/Protocol Analysis (2)

Reiter and Marsalek published in 2017 [137] an article describing new attacks to WebRTC. They identify multiple unprotected assets exposed by WebRTC that can be leveraged by attackers. These assets are the peers' public and private IP addresses, the local network, bandwidth, and peer identity. Based on these assets they present four attacks and possible mitigation techniques. They first consider an untrusted signalling path and show that a MitM attack can be mounted against the media path.

This attack is already considered in the WebRTC security architecture Internet Engineering Task Force (IETF) draft [47] which proposes the use of an identity path (see Section 1.3). Observing that this solution introduce dependencies to third-party IdP, Reiter and Marsalek propose manual verification of DTLS certificates as a "lightweight alternative". Two others presented attacks use Interactive Connectivity Establishment (ICE) IP address leaks against a peer's privacy, in particular, to allow device fingerprinting. Finally, they show that a flooding attack can be mounted from malicious JavaScript, *i. e.* without relying on an infected host. The JavaScript setups a WebRTC connection and then sends multiple ICE candidate offers to flood the target.

Also considering ICE IP address leaks, Al-Fannah *et al.* [138] test combinations of Operating System (OS), browser, Virtual Private Network (VPN), and VPN configurations. Based on the results from their 116 test cases, they report differences in the type of address leaked. They recommend that users concerned by this vulnerability carefully choose their browser and VPN.

Authentication Protocols (1)

De Groef *et al.* [139] try to determine whether "WebRTC provides endpoint authenticity guarantees for the peer-to-peer connection"? They consider the integrity and binding of DTLS certificate to the identity assertion as a prerequisite to ensure endpoint authenticity. Three possible attacks are described, relying either on a malicious signalling server or a malicious third-party JavaScript provider. Firstly, the DTLS fingerprint may be compromised by a malicious JavaScript provider, supposing no binding with an identity assertion. Their second attack assumes the IdP does not correctly check the origin of request from IdP Proxy. A malicious JS provider tricks an IdP to sign a certificate for a certificate controlled by an attacker, allowing a MitM attack. Finally, they argue that the lack of user interface controls to select a preferred identity or IdP undermine the integrity of identity assertion in WebRTC. They discuss mitigation strategies for each actor and in particular that "the browser should provide the necessary User Inteface (UI) chrome to enable users to select an appropriate identity from their favourite Identity Providers, and, even more important, enables them to only grant access to remote identities of their choice to set up a peer connection". They also recommend that "website owner needs to ensure that in all cases an Identity Provider is used [and if no] external Identity Provider is needed, the website owner can deploy his own IdP Proxy, that [could] for instance piggybacks on the session mechanism for the website". While code snippets examples are provided for each attack, no implementation of an IdP Proxy is referenced. In particular, the second attack refers to a "rtcweb://" origin and communication to IdP Proxy through the postMessage Application Programming Interface (API) although we were not able to find references to such features in the specifications.

Architectures (8)

Murányi and Kotuliak [140] simulate the interconnection between a WebRTC based streaming service and the IP Multimedia Subsystem (IMS) using OpenID. OpenID is implemented as an Single Sign-On (SSO) solution on the web service and used to perform AAA. It is however not used as peer to peer authentication as proposed by the WebRTC identity architecture.

Li *et al.* [141] consider a WebRTC architecture with multiple communication services. They observe a mismatch between the WebRTC identity architecture and traditional SSO authentication[11]. To solve this issue they propose three alternative identity architectures. The first architecture relies on an Identity Adaptor Provider (IdAP) providing IdP Proxy and interfacing between the browser and an identity provider. In the second architecture, a communication service from a domain (site B) can request a

11: In their scenario communication services may use IdP Proxy to authenticate users from other domains. This possibility is not considered by the WebRTC identity architecture which only considers IdP Proxy in peer to peer authentication.

Identity model	Identification	Anonymity to peers	Anonymity to CS	Unlinkability	Identity conf.	CS unlink.
Nontrust (BrowserID)	✓	–	–	–	–	✓
Nontrust (RP-Centric)	✓	✓	✓	✓	–	–
Partial trust (RP-Centric)	✓	✓	✓	✓	–	–
Full trust (no SSO)	✓	✓	–	–	–	n/a

Table 2.2: User privacy properties in identity provision model [142].

user (Alice) from another domain (Site A) to authenticate by issuing a challenge. Alice then authenticates using her own IdP which suppose the existence of a trust relationship between Alice's IdP and Site B. Finally, the last architecture proposes to set up a web-of-trust between identity providers and based on PGP. This web-of-trust allows creating an authentication chain between two browsers. While the paper discusses several architectures and authentication protocols, no implementation is mentioned.

Beltran *et al.* [142] works on the trust relationships between actors of the call setup implied by the WebRTC identity model in a single CS scenario. They identify differences between browser-centric SSO protocols, *e. g.* BrowserID, on which the WebRTC identity model is based and Relying Party (RP)-centric protocols such as OAuth 2 or OpenID Connect (OIDC). They discuss adaptation of RP-centric protocols to the WebRTC model, however without discussing implementation. Finally, they evaluate whether user's privacy is protected depending on the underlying trust model and SSO protocol used as presented in Table 2.2. Beltran *et al.* also discuss the question of trust relationships between actors in enterprise communication scenarios in two other articles [143, 144]. In particular, they observe that as identity providers may not know which are the targeted communication services, applying enterprise-specific policies may prove to be difficult.

In follow up papers, Javed *et al.* [145, 146] continue working on a WebRTC trust model. Their proposed trust architecture is presented in Figure 2.6. For each trust relations, they propose attributes that should be considered to evaluate a trust relation. In their model, trust relations represent previous experiences, identification, and reputation each as a vector representing trust, distrust, and mistrust. For instance the vector $< 1, 0, 0 >$ represents an absolute trust, while $< 0, .5, .5 >$ represent a doubtful distrust. Trust scores are computed as the proportion of previous good, bad, or unknown previous interactions and weighted by an ageing factor. They also refine the privacy comparison of the SSO protocol used in WebRTC identity architecture proposed by Beltran *et al.* [142]. While they propose an extremely detailed trust model for WebRTC, Javed *et al.* do not clearly explain how they extract input for their trust score from real WebRTC services and user input. For instance, it is not clear what defines a bad experience with a communication service and whether it can be observed at all. We also note that their proposed trust model do not consider authentication of IdP and CS server, *i. e.* TLS channels.

Copeland and Copeland proposed in 2016 [147] a "better than best effort" architecture allowing communication services to select the appropriate network depending on selected profiles. These profiles are built on balance between four considered criteria: QoS, Urgency, Security, and Affordability (QUSA). Profiles and the associated network are selected based on the context of the communication. The context is derived from input from multiple sources, *e. g.* calendar, social network, or location, with each source being attributed precision, accuracy, and confidence scores. The overall architecture is summarised in Figure 2.7. The paper claims to have simulated 200 cases but point at

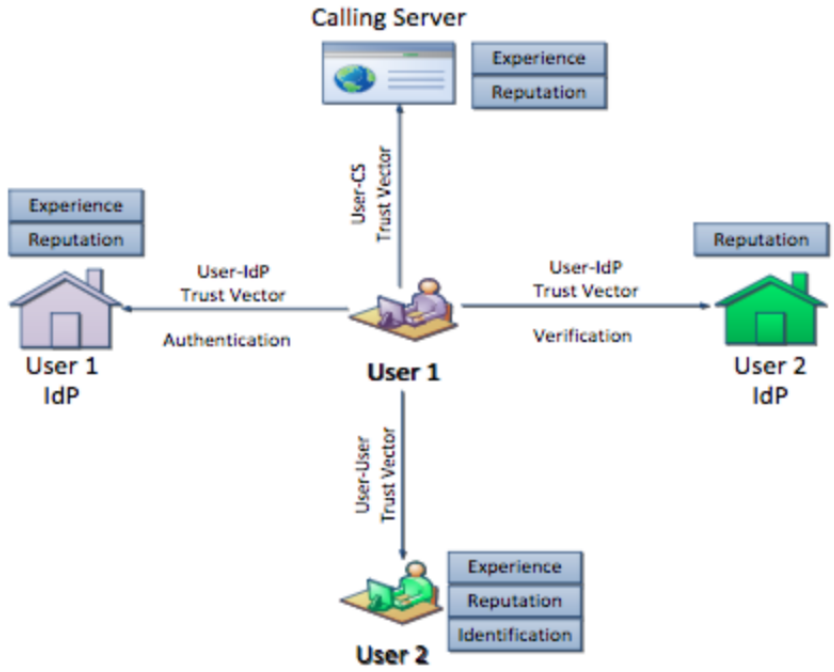


Figure 2.6: Javed *et al.* [146] WebRTC trust model.

the absence of real-world data.

Social Threats (1)

Following on their WebRTC trust model, Javed *et al.* [148] propose a trust model evaluating trust in peers of a WebRTC communication in real-time. In their model, a trust value is centrally computed by the communication service for each peer. This trust value is the weighted sum of an authenticity score, in fact a reputation, and a behavioural score based on talk time, incoming call numbers, and outgoing call numbers. They simulate their approach and compare it to other trust models from the literature against some VoIP social threats and trust model attacks. Security considerations in this work are quite limited and only consider the authenticity of the other peer. Furthermore, this authenticity of a peer is actually a reputation score rather than a measure of actual authentication.

In 2015, Vapen et al. [149] studied the identity management landscape on the web[12]. In their study, they classified the type of information shared by IdP to RP in five classes: basic information, personal information, created content, friend's data, and a transversal action class. They also defined semi-ordered risk types classes, build as conjunction of information shared classes. These classes range from R- to RA++ risk levels, A denoting action authorization. In addition, their observations show that in practice RP offer few choices of IdP to their users, with 47 % offering only one IdP, and 19 % offering four or more IdP. This situation profits to a few IdP trusting the top ranks, with Facebook as the number one, followed by Google and Twitter.

12: This paper is not part of our VoIP security research collection.

Middleboxes (2)

Johnston *et al.* [150] look at the issues of WebRTC communication services in enterprise networks, *i. e.* the traversal of enterprise firewalls for WebRTC session negotiated over

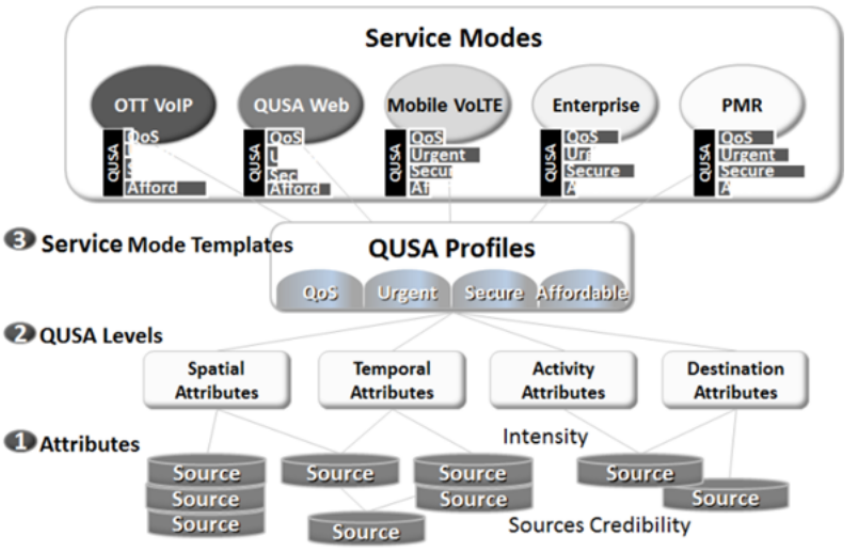


Figure 2.7: Service Mode decision process based on context and QUSA profiles.

HyperText Transfer Protocol Secured (HTTPS). They first present an overview of the issue and identify that current approaches for session border control and enterprise policies are not applicable to WebRTC. The reason is that contrary to traditional VoIP application using SIP, WebRTC applications may not expose sessions information to firewalls. Several possible solutions are then described and discussed. Other issues, not related to security, are also discussed such as the interoperation of WebRTC technologies with existing VoIP infrastructures.

Singh *et al.* [151] implemented a Google Chrome extension to apply enterprise policies to WebRTC calls without help from underlying web application. The extension overloads the WebRTC API to intercept calls, it can then inserts user's enterprise identity in signalling messages. Rather than implementing an IdP offering an IdP Proxy, the extension relies on an enterprise public key infrastructure to sign and verify signalling message. The extension also forces the use of an enterprise media relay, *i. e.* a modified TURN server responsible for applying enterprise's policies.

2.2.4 Observations

We now compare the surveyed state of the art on WebRTC security to our research questions. These observations allow us to narrow the focus of our contributions presented in later chapters.

Threats against user's security in the context of real-time multimedia communications and mechanisms to protect against these have been well-studied. WebRTC has been built on this foundation, and the state of the art on VoIP security research continued to develop since then. As WebRTC is not a full-stack solution, we observe that researches on WebRTC and on WebRTC security mainly focus on security at the endpoint, including privacy risks for the users, and the issues of WebRTC deployment in enterprises. One novelty introduced by the WebRTC security architecture is the integration of third-party IdP into the communication setup through the IdP Proxy mechanism. This specification attracted a lot of interest from the community: as presented in Table 2.3 we observe that out of 22 WebRTC security papers, a total of 13 papers reference the WebRTC identity architecture. However, the security and privacy of this

	Category / Title	Year	Security	Trust	Privacy	Negotiation	IdP Proxy
	Social Threats						
[148]	*TrustCall: A Trust Computation Model for Web Conversational Services*	2017		✓			✓
[149]	*Information Sharing and User Privacy in the Third-Party Identity Management Landscape*	2015			✓		
	Traffic Attacks						
[127]	*A Decision Theory Based Tool for Detection of Encrypted WebRTC Traffic*	2015	✓				
[128]	*Revealing Encrypted WebRTC Traffic via Machine Learning Tools*	2015	✓				
	Overviews and Surveys						
[129]	*Real-time Communications for the Web*	2013	✓				✓
[130]	*Browser-to-Browser Security Assurances for WebRTC*	2014	✓				✓
[131]	*Real-Time Communications in the Web: Issue, Achievements, and Ongoing Standardization Efforts*	2012	✓				
[132]	*How Far Are We from WebRTC-1.0?*	2017	✓				✓
[133]	*A Survey on Real-Time Communication for Web*	2015	✓				✓
[134]	*On the Seamless Interaction Between WebRTC Browsers and SIP-based Conferencing Systems*	2013	✓				
[135]	*D.1.1 Web-platform security guide*	2013	✓				
[136]	*D1.2 Case Study: Security Assessment of WebRTC*	2014	✓				
	Field Studies and System/Protocol Analysis						
[137]	*WebRTC: Your Privacy Is at Risk*	2017	✓		✓		✓
[138]	*One Leak Will Sink a Ship: WebRTC IP Address Leaks*	2017			✓		
	Authentication Protocols						
[139]	*Ensuring Endpoint Authenticity in WebRTC Peer-to-Peer Communication*	2016	✓			R	✓
	Architectures						
[140]	*Identity Management in WebRTC Domains*	2013		✓			
[141]	*Who Is Calling Which Page on the Web?*	2014		✓			✓
[142]	*User Identity for WebRTC Services: A matter of trust*	2014		✓	✓		
[143]	*Unified Communications as a Service and WebRTC: an Identity-Centric Perspective*	2015		✓			✓
[144]	*Identity Management for Web Business Communications*	2015		✓			✓
[145]	*Browser-to-Browser authentication and trust relationships for WebRTC*	2016		✓	✓		✓
[146]	*Br2Br: a Vector-Based Trust Framework for WebRTC Calling Services*	2016	L	✓			✓
[147]	*A Question of Quality - VoIP, WebRTC or VoLTE?*	2016				✓	✓
	Middleboxes						
[150]	*Taking on WebRTC in an Enterprise*	2013		P			
[151]	*Enterprise WebRTC Powered by Browser Extensions*	2015		P			L

Table 2.3: List of reviewed WebRTC papers. Checkmarks indicate whether the papers look or address the issues of security, trust model, privacy, security parameters negotiation, and the WebRTC identity architecture. The letters stand for L: *limited*, R: *recommends*, and P: *policy trust*.

specification are only studied from a theoretical point of view [139]. In particular, while WebRTC is intended to be interoperable with any SSO protocol, cross-implementation issues between the SSO protocol and WebRTC are rarely considered [141, 139]. The specification itself only sketches a short example using OAuth in annexe A [47]. Additionally, other works study the privacy implications of the identity architecture but only consider the privacy threats posed by a malicious signalling server against user identity. Consideration for this type of threats is already present in the WebRTC security architecture draft [47]. However, we do not observe research considering the privacy of the communication against the identity provider itself.

Besides understanding the threats faced by WebRTC users, we also want to act on a WebRTC session to raise the trust and security level. Some surveyed work propose to negotiate the configuration of a WebRTC or VoIP communication setup in order to achieve a given security level. The solution of Copeland and Copeland [147] focus on selecting an appropriate underlying network, *i. e.* at the network access layer, which does not allow to manage upper layer parameters. Alia *et al.* [124] propose an interesting model for balancing security and QoS. However, their utility function is not convincing. While an additive approach is coherent for modelling performance overheads, it does not achieve to model the dependent nature of security parameters. For instance, a weak integrity on the signal path may have an important impact on confidentiality of the media path.

Increasing privacy, for instance to mitigate ICE IP address leaks [138], mostly relies on permanent configuration options such as the selected OS or browser parameters. Choosing proper actors to participate in the communication setup may also be a way to increase privacy, either because they implement privacy-preserving protocols or because they are trusted to not compromise user's privacy. Regarding this last point, we note that De Groef *et al.* [139] recommend that browsers allow users to select an IdP of their choice, *i. e.* trusted, to participate in WebRTC sessions set up.

We want a model representing both security and trust to help users in the configuration and negotiation of WebRTC security parameters. A lot of work has been conducted on modelling reputational-trust in WebRTC [142, 143, 144, 145, 148]. However, these papers generally do not consider the security of the session and the strength of security parameters in their models. At most, only the user authentication strength is taken into account [146]. Alternatively, some researchers [150, 151] consider trust policies with respect to WebRTC security. However, they focus on applying enterprise policies and do not propose a complete model of WebRTC security.

2.3 Summary

We conducted a research survey on VoIP and WebRTC security research. In total, we classified 208 research papers of which 23 were actually addressing WebRTC security. We then reported on the researches conducted in these 23 WebRTC security paper.

Firstly, we observe that the WebRTC identity architecture attracted a lot of interest from the community. However, we note that the security and privacy of the specification are only studied from a theoretical point of view [139]. In particular, the cross-implementation issues between SSO protocol and WebRTC are rarely considered [141, 139]. The specification itself only sketches an implementation with the OAuth protocol in its annexe [47]. We neither observe research considering the IdP as a possible attacker of the user's privacy. In order to remedy to this issue, we intend to base our analysis of the WebRTC security and identity architecture on an actual implementation of SSO protocols in the context of a WebRTC service.

Secondly, one of our research objectives is to build a model representing the trust and security of a WebRTC session and capable of returning a single metric. In our survey, however, we do not observe research combining elements of trust and security models. At best, one of the trust model uses limited security parameters [146] which confirms that combining trust and security for VoIP is a novel approach. Existing researchs work on the dynamic configuration of VoIP services, but also uses a simplistic security model [124]. Nevertheless, other researchers recommend that users be given more control over which IdP they use in VoIP [139] and more broadly on the Web [152]. As we want to give them more control over the trust and security of their WebRTC session, we will focus on allowing users to negotiate WebRTC identity parameters.

Part II

Contributions

Foreword on Methodology

As we explained in Section 1.1, WebRTC is a set of standard specifications aiming to provide to webpages the capability to setup VoIP communications. These specifications also rely on other standards and techniques: authentication delegation protocols, VoIP protocols and archirectures, JavaScript API, cryptography, ... At the heart of the WebRTC identity architecture is the proposal of an abstract authentication interface for peer-to-peer authentication and the claim that trust in an IdP can replace trust in the signalling path. We aim to study these propositions and at the same time answer our research questions. However, we believe that formal paper-based study of standards must be conducted in conjunction with study of running implementations, in a complementary approach. Indeed, standards are ultimately implemented in actual running code. During this continuous process, implementors may take liberties with the specification, add non-standard functionalities, or decide to not implement some of them. Furthermore, cross-implementation issues may arise, in particular when the integration of two specifications has not been thoroughly considered. This is precisely the case of the WebRTC identity architecture which only sketches the OAuth 2 protocol integration. It appears that, from our state of the art survey in Chapter 2, most of the researches on WebRTC security have been conducted without considering actual implementations.

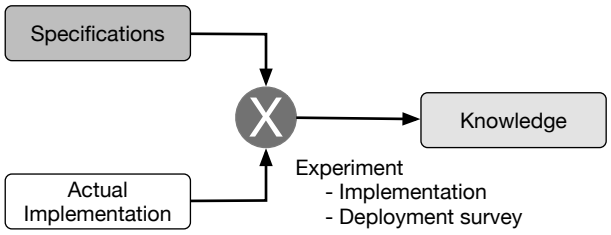


Figure 2.8: Overview of our Scientific Methodology

Our approach, which we schematise in Figure 2.8, differs. The first step of our work is to conduct a study of these standards and techniques. We presented the results of this step in Chapter 1 and 2. However, we then use actual implementations to answer our research questions. To do so, we conduct two types of experiments:

- In implementation experiments we develop software to put an already specified or a new functionality into action. This allows us to discover actual issues encountered during the implementation and integration processes. We describe our implementations so that our experiments can be reproduced [13].
- In deployment survey experiments, we observe whether a functionality is implemented or used in existing and deployed softwares. Observing an exposed functionality demonstrates if and how it can be used, while observing its actual usage reveals its importance for other services.

13: The description of our implementations may also be helpful for developers facing similar needs.

These experiments form the basis of our scientific methodology which we apply in the following contribution chapters.

Chapter 3

Privacy Implications of the WebRTC Identity Architecture

A claim of the WebRTC security architecture specification [47] is that trust in the signalling layer can be replaced by trust in the IdP. This has implications regarding potential privacy issues. As signalling and identity functions are decoupled, a new actor (the IdP) is introduced in the communication setup. Even-though IdP already occupy a central role on the Web, their role in WebRTC has the potential to reinforce their position. In this chapter, we study the privacy implications of the WebRTC Identity Architecture. This part of the WebRTC specification lacks support on web browsers. To the best of our knowledge, there is no publicly implementation or deployment of a WebRTC identity enabled WebRTC service or of an IdP supporting the WebRTC identity architecture. To better understand the WebRTC Identity Architecture, it is thus necessary to first implement it. We describe our implementations in Section 3.1. We then detail additional privacy considerations in Section 3.2. One issue we observe is that the IdP choice is limited by the CS which may appear contradictory with the initial objective of the specification. In Section 3.3 we study why users cannot choose their IdP on the Web.

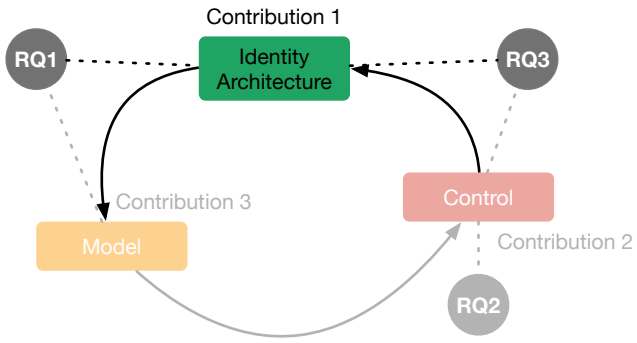


Figure 3.1: Overview of our Contributions: Study of the WebRTC Identity Architecture.

3.1 WebRTC Identity Architecture Implementation

The explicit peer authentication proposed by the WebRTC specification plays a central part of the WebRTC identity architecture. But it is also, for the moment, a feature whose implementation in browser lags behind others WebRTC features. The

`RTCPeerConnection` identity interface is only implemented in Firefox. As a result and to the best of our knowledge, no Identity Provider (IdP) or WebRTC service are supporting it[1]. Our first research question (RQ1) consists in understanding the risk for the user of a WebRTC session. More particularly, in this section we address the following question:

1: `https://bugs.chromium.org/p/chromium/issues/detail?id=493640`

- **RQ1.1**: Are there any security vulnerabilities in the identity path of the WebRTC security architecture?

We want to validate our work on a running implementation of the WebRTC identity architecture. To do so, we develop a simple WebRTC service[2] offering communication for two users in a single room. The communication server is built with the NodeJS framework. Session signalling is done through the Communication Service (CS) server over web sockets and the JavaScripts client code manages the call session. We conduct our tests on Firefox version 50.1.0.

2: `https://github.com/Sparika/ACOR_SDP`

As the RTCPeerConnection identity interface is provided by Firefox, the missing part of the WebRTC identity architecture is an IdP exposing IdP Proxy. In addition to a WebRTC service, we also implement IdP Proxies for three different IdP, following the WebRTC specification. The first IdP is actually our WebRTC communication service itself in what we call a local authentication scenario. We present this implementation in Section 3.1.1. Our two other IdP Proxies are developed for two OIDC servers. The first server is one of the reference implementations by Nat Sakimura[3], while the second one is an implementation in NodeJS[4]. We also conducted these implementations within the context of the reThink project, in order to reuse WebRTC identity architecture for the reThink Identity Module (see Section 1.6.3). Section 3.1.2 details how we implement OpenID Connect (OIDC) IdP Proxies and how we adapt their respective IdP servers. Finally, in Section 3.1.3 we discuss our implementations.

3: `https://github.com/reTHINK-project/dev-IdPServer-phpOIDC`

4: `https://github.com/reTHINK-project/dev-IdPServer`

3.1.1 Local Authentication Implementation

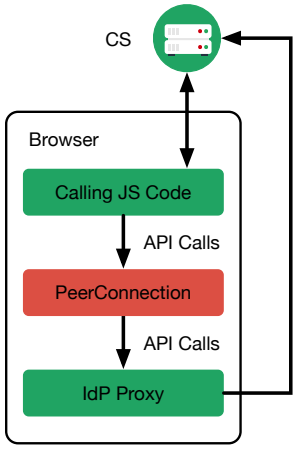


Figure 3.2: Local Authentication Architecture

Local authentication refers to a scenario where the CS also plays the role of the IdP in the WebRTC identity architecture, as in Figure 3.2. It is not the architecture initially envisioned by the specification. As we explained in Section 1.3, the decoupling of identity and signalling functions are meant to prevent the CS from setting up a man-in-the-middle during the call session establishment. This architecture may nonetheless be useful. Firstly, it provides a simple test use case for an IdP Proxy with a login and password authentication as used by most websites not relying on IdP. Secondly, in an interoperable signalling architecture, multiple CS are used to establish the session. A user may not trust the signalling path, except for his own CS. In this case using his CS as an IdP would offer a protection against man-in-the-middle attack from other CS. Finally, in compatibility scenarios call transit through a legacy interface, for instance, a Session Initiation Protocol (SIP) Gateway. In this case, the legacy interface plays the role of both the CS and the IdP.

Functionally, our implementation of the IdP Proxy maps the identity assertion to the `contents` parameter, *i. e.* the session fingerprint. The identity assertion thus serves as the key to retrieve claims covered by the assertion, and verify in the process that these claims were effectively registered on the IdP. Figure 3.5 presents the identity assertion generation interface exposed by the server to the IdP Proxy and its sequence diagram. To store a new pair, the `generateAssertion` function of the IdP Proxy POST a content to the `/assertion` REST interface. After the server checked that the user is logged in, the user's identity and content parameter are stored in a map and the key is returned with an HyperText Transfer Protocol (HTTP) 200 success response. The IdP Proxy then uses the key to instantiate an `RTCIdentityAssertionResult` (see Figure 3.3) and resolve the

`generateAssertion` promise with it. On the promise's resolution, the browser adds the assertion dictionary to the Session Description Protocol (SDP) message.

```
dictionary RTCIdentityAssertionResult {
    required RTCIdentityProviderDetails       idp;
    required DOMString            assertion;
};

dictionary RTCIdentityProviderDetails {
    required DOMString                    domain;
             DOMString                        protocol = "default";
};
```

Figure 3.3: RTCIdentityAssertionResult specification in WebIDL. The assertion is an "opaque string that MUST contain all information necessary to assert identity". It is consumed by the validating IdP.

Alternatively, if the user does not have an active session, the `generateAssertion` promise is rejected with an `IdPLoginError` JavaScript Object Notation (JSON) object. This object may contain an `idpLoginUrl` element, which can be used by the client service to open a login page on the IdP. A successful login following this Uniform Resource Locator (URL) is signalled by a `LOGINDONE` message sent using the postMessage Application Programming Interface (API) to the login page's opener window, *i. e.* the communication service client page.

```
<script>window.opener.postMessage('LOGINDONE', '*')</script>
```

Figure 3.6 shows the *local* identity assertion validation REST interface and its associated sequence diagram. To validate a received identity assertion, the peer's browser downloads the IdP Proxy from the IdP which produced the identity assertion. The `RTCIdentityProviderDetails` dictionary (see Figure 3.3), included in the assertion, describes the IdP Proxy location. Once the browser instantiated the verifying IdP Proxy, it calls the IdP Proxy's `validateAssertion` function. Our implementation of this function sends a GET request to the `/assertion` interface, using the provided assertion (see Figure 3.6a). The result of the request is a JSON object containing the stored identity and content. This object is used to instantiate an `RTCIdentityValidationResult` dictionary (see Figure 3.4) which is then returned to the browser. The browser verifies that the provided `contents` matches the SDP fingerprint attribute, binding the fingerprint to the assertion's `identity`.

```
dictionary RTCIdentityValidationResult {
    required DOMString                    identity;
    required DOMString                    contents;
};
```

Figure 3.4: RTCIdentityValidationResult specification in WebIDL.

3.1.2 IdP Proxy with OpenID Connect

In order to integrate our OIDC servers into the WebRTC identity architecture, the IdP Proxy acts as the OIDC client. In this role, the IdP Proxy requests an ID Token to the IdP. The IdP authenticates the user and verifies that the user has authorized the IdP Proxy to obtain an ID Token. As the IdP Proxy is a JavaScript code running inside the user's browser, *i. e.* client side rather than server side, we use the OIDC *implicit flow*. This flow allows the client to directly get the requested token, as explained in 1.3.5. The

POST **/assertion**

Parameters

content *(query)*	The fingerprint string to use as a claim in the identity assertion.

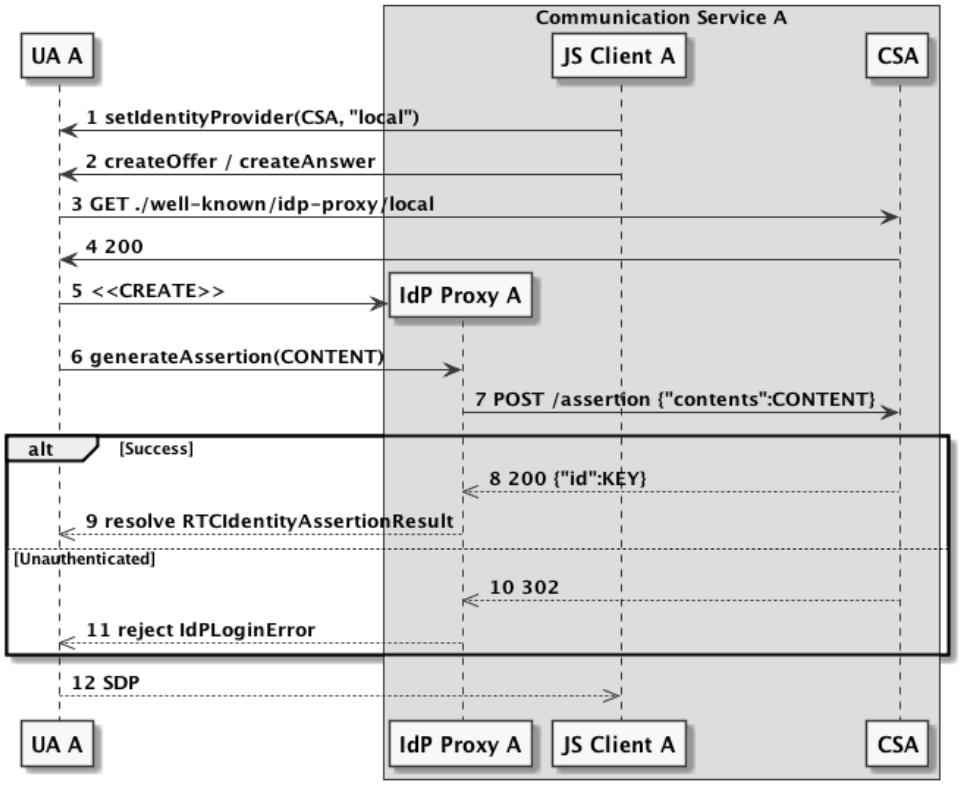
Response

200	Success

```
{
  id: 0,
  "user": "bob@idp.com",
  "contents": "01234"
}
```

302	User not authenticated

(a) /assertion POST API

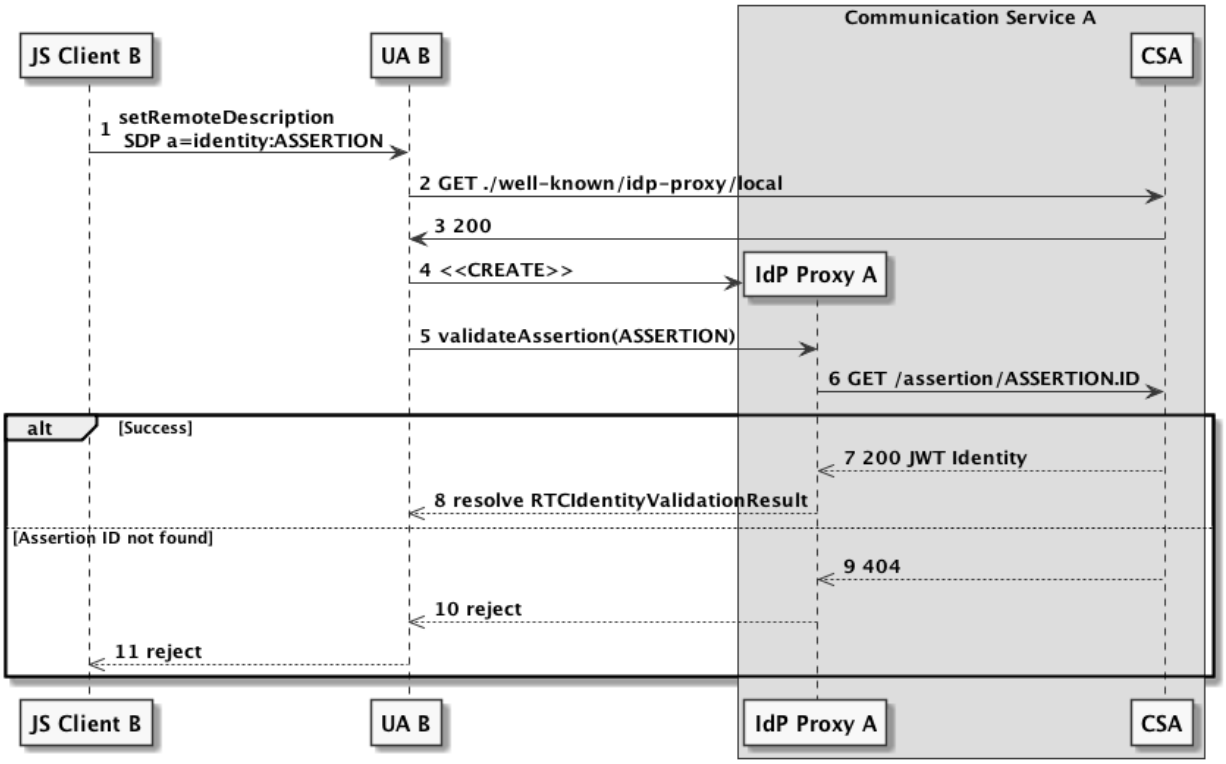


(b) Sequence Diagram

Figure 3.5: Local Identity Assertion Generation

GET	/assertion/{assertionId}	
Parameters		
assertionId *(path)*		The assertion id to verify.
Response		
200		Success
		`{ "identity": "bob@idp.com", "contents": "01234" }`
404		Identity assertion not found

(a) /assertion GET API



(b) Sequence Diagram

Figure 3.6: Local Identity Assertion Verification

resulting ID Token covers the user's identity and the session fingerprint as claims. It thus serves as the WebRTC identity assertion. Our implementation, however, requires additional modifications to OIDC requests.

Firstly, we include the session fingerprint as a claim of the ID Token payload so that it is covered by the server's signature. To send it to the server, we define a new request parameter, `rtcsdp`, to convey this value in the request. In practice, this parameter function is quite similar to the `nonce` request parameter. Both convey random opaque numbers making the request unique. However, in some cases, a WebRTC session may reuse a previously established key and thus the same session fingerprint twice.

Secondly, OIDC interactions with the user normally happen through a new tab or popup opened by the browser. This graphical user interface allows the IdP to authenticate the user or request user's consent before authorizing the requesting client. However, in our case, the IdP Proxy is running in a sandboxed invisible iframe. The OIDC `/authorize` GET request (see Figure 3.7) is thus executed through the Fetch API[5]. Such requests are invisible to the users, no new tab or window are opened. Hence why the IdP Proxy has to throw an `IdPLoginError` in order to interact with the user. As we described in Section 3.1.1, this happens if the user is unauthenticated. In the OIDC case, it also happens if the IdP Proxy client has not been authorized by the user. The login or authorization URL is returned in an `IdPLoginError` to the CS, which opens it so that the user can login or authorize the IdP Proxy. In either case, the process followed by the user lands on a page messaging the `LOGINDONE` signal to the CS. The url of this page is defined using the `redirect_uri` parameter of the `/authorize` request. When receiving this message, the CS restarts the `generateAssertion` procedure.

5: Fetch is a standard web API for network requests similar to the XMLHttpRequest (XHR) interface. An advantage of using the Fetch API is that it easily integrates with the promise based WebRTC interfaces. See `https://developer.mozilla.org/en/docs/Web/API/Fetch_API` for more details.

Finally, we also implement a new `response_mode` value. In the OIDC implicit flow, a successful authorization redirects to the client web page. The ID token would be returned with the redirection either in the redirected URL's query or fragment. However, both query or fragment are inaccessible from a Fetch (or XHR) response after following a redirection. Instead, the IdP returns the ID Token in the response body. We thus define a new `response_mode` value: `body` and modify the IdP server implementation accordingly.

Note that the `response_mode` and `redirect_uri` parameters are conflicting as they correspond to different HTTP response code, 200 and 302 respectively. In our implementation, the `redirect_uri` is only followed if `response_mode` is not set to `body`. Figure 3.7 shows our modified `/authorize` request specification.

To verify the ID Token validity, its signature must be verified by the IdP Proxy when it is executed from the other peer's browser. It is either possible to add the IdP's public key in the IdP Proxy code or retrieve it from a secured location on the IdP. Optionally a reference to the key URL could be included in the ID Token header `jku` parameter. This solution is discouraged by the OIDC specification which states that "ID Tokens SHOULD NOT use the JWS or JWE x5u, x5c, jku, or jwk Header Parameter fields" [62].

3.1.3 Observations

We note that the Firefox implementation is more restrictive than the specification regarding identity format. Firefox requires that the identity, the human-readable identifier, be in an email format with the domain of the email equals to the IdP Proxy domain. This format prevents domains from asserting identity from other domains, *e. g.* identifier ending in `@orange.com` could not be claimed by an IdP Proxy from the `dr.evil.net` domain. However, using the email's domain as an indication of the IdP's domain name is often an over-simplification. As an example, Figure 3.8 shows the Google identity selection popup. Although the popup's domain name, *i. e.* Google's IdP domain, is `accounts.google.com`, user's identifiers may end in `@gmail.com`. Some valid identifiers

GET /authorize

Parameters

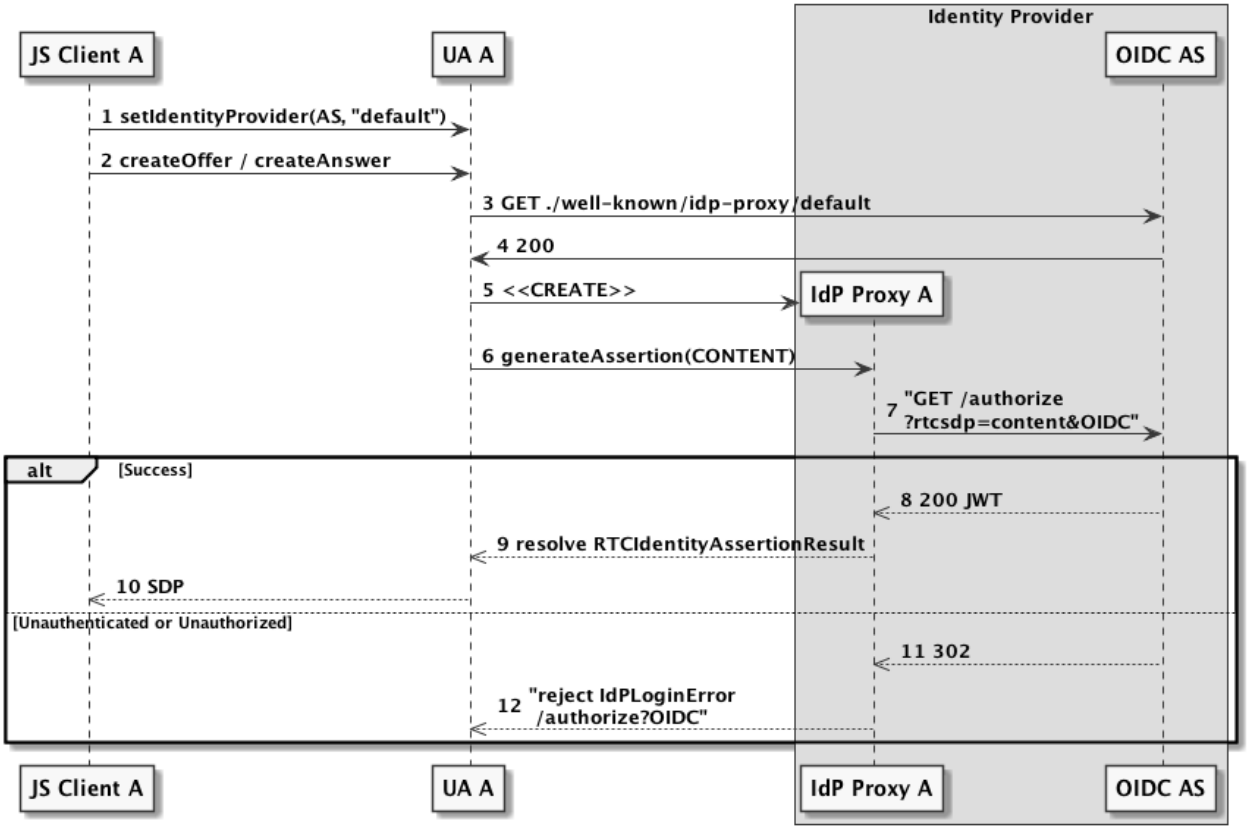
scope	**openid**
client_id	The client id registered for the IdP Proxy
redirect_uri	Redirection to the page notifying the IdP Proxy with 'LOGINDONE'.
response_type	**id_token**
nonce	Used to mitigate replay attacks.
response_mode	**body**
rtcsdp *(query)*	The fingerprint string to use as a claim in the identity assertion, in base 64 encoding.

Response

200	Success - Base 64 encoded JWT with the **rtcsdp** parameters as a claim.

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwiaXMiOiJpZHAuY29tIiwicnRjc2RwIjoiMDAwMDAifQ.II5BL60gLyS-PEwEZ6E7U_fXzayI2YDsmhKtX3uhC80

(a) /authorize GET API



(b) Sequence Diagram

Figure 3.7: OIDC Assertion Generation

Table 3.1: JavaScript code lines for the local authentication IdP Proxy implementation.

Module	New code lines	Total code lines
Proxy	66	66
Routes	38	290
Model	16	89
Project	120	6641

may even end in other domain names if the user did not activate his Gmail mailbox, for example, `@orange.com`.

As shown by Table 3.1 the implementation of the local IdP Proxy is quite simple. Proxy, Routes, and Model give the new and total number of JavaScript code lines respectively for the IdP Proxy, the HTTP interface implementation, and the database model. On the project as a whole, our implementation required only 120 JavaScript code lines. A number to be compared to the 6641 code lines, excluding NodeJS dependencies, of the full WebRTC service project.

Compared to the local IdP Proxy, implementation of an OIDC IdP Proxy is a complex task. Firstly, while following the same structure, the IdP Proxy is larger due to its requirement of ID Token signature verification, client key retrieval, and the handling of JSON objects. The IdP Proxy JavaScript file contains 207 JavaScript code lines. The more complex modifications are however made on the OIDC server itself. We add a few utility functions to the HTTP interface, but we also modify the core OIDC modules to support the parameters we introduced: `response_mode=body` and the `rtcsdp` ID Token claim. While these modifications are not that heavy in terms of code lines, they require the understanding and modification of a large code base. As an example, the NodeJS dependency implementing the OIDC specification contains 1274 code lines in a single file.

Another issue of the OIDC implementation is its reliance on non-standard modification of the specification. Industrial deployment of standards protocols may follow strict policies regarding the implementation of non-standard features. In our case and due to the policies of our company, integrating our proposed change to the OIDC server in production would have required to first get them published by the OpenID foundation. The complexity of the OIDC implementation does not seem to brings benefits compared to the simple solution of the local authentication IdP Proxy.

Our implementations shows that OIDC facilitates the creation and signature of WebRTC identity assertion in an open standard format: OIDC ID Token. However, it requires the modification of a complex code base not initially designed for this use case. While using the ID Token format may have practical use cases to exchange WebRTC Identity assertion. For some use cases, it may be as simple to use a map interface secured through HyperText Transfer Protocol Secured (HTTPS).

3.2 RQ1.1 Additional Privacy Considerations

We presented the WebRTC security architecture in Section 1.2 and reviewed previous research works on WebRTC security in Section 2.2. From this work, we observed that the identity path proposed by WebRTC security architecture has only been studied from a theoretical point of view and that in these works IdP had not been considered as possible attackers against the users' privacy. In this section, we answer RQ1.1 by presenting new risks for the user's privacy introduced by the IdP Proxy component. These privacy considerations were revealed by our implementation process.

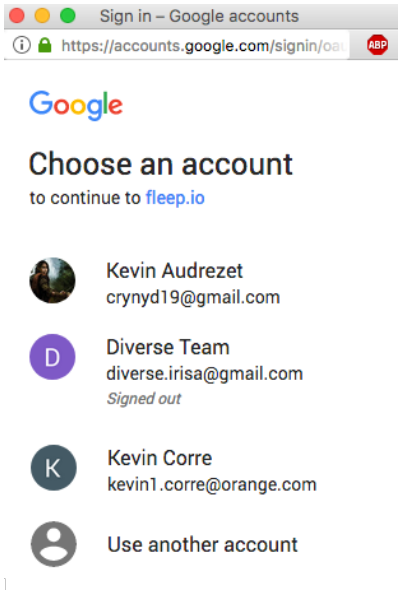


Figure 3.8: Google identity selection webpage.

3.2.1 Audience Issue

In addition to the `rtcsdp` claim, *i. e.* the session fingerprint, the WebRTC identity assertion carries two important information: the peer's identity in human readable form, and the peer's identity provider fully qualified domain name. Both pieces of information are necessary so that the assertion can be verified and associated with a peer's identity. But from a privacy point of view, they are sensible identifying information. In the SDP message, the identity assertion is an opaque string encoded in base 64, which means that it has no actual definition. However, the IdP Proxy can be downloaded by any party and instantiated to decode the identity assertion.

Although we explain this privacy issue with the OIDC scenario, it is present whether or not the IdP Proxy is based on OIDC. As we explained in Section 1.3.5, OpenID Connect is based on OAuth2, an authorization protocol. As such the concept of authorization and user consent is central to the protocol. In theory, the client, identified by the `client_id` in OIDC requests and the `audience` claim, is authenticated by the IdP. In the implicit flow, the client authentication is not performed explicitly and instead relies on the `redirect_uri` redirection to the client. In the WebRTC use case of user-to-user authentication, the `redirect_uri` is not followed as the ID Token is transmitted to the other peer over SDP signalling. There is no clear definition of the intended audience[6] and in our implementation we define and use a unique IdP Proxy client and `client_id`. We explored the alternatives of one `client_id` for each users, each CS, or for each pair of user/CS without finding clear advantage to any of these solutions.

6: The audience may not even be known a priori, for instance in the case of a call to an anonymous peer.

As represented in Figure 1.15, both `generateAssertion` and `validateAssertion` functions use an origin parameter. This parameter is the origin of the CS which required the instantiation of the IdP Proxy However, the `validateAssertion` is executed after the identity assertion is transmitted over the signalling path. As a result, the IdP does not know the signalling endpoint's origin during the generation of the assertion. Some policy checks could still be applied during the execution of the `validateAssertion` function, eventually preventing the decoding and decryption of the identity assertion. But the specification does not define the error that should be returned in this scenario and does not plan for transmitting a consent request back to the user. Implementing such consent mechanism could be allowed using a push notification to the user's device.

On the other hand the `generateAssertion` can interact with the user through the

`IdPLoginError`. It can thus be subject to user authorization and IdP policies, as long as the IdP implements such mechanism. Otherwise, without appropriate protection on the IdP Proxy, a web page could look for the user's identity. This could reveal user identities, or at least existing user accounts on IdPs. For instance, given a list of WebRTC compatible IdPs, running the script described in Figure 3.9 would reveal identity assertion from unprotected IdPs with active sessions. This is a major issue, and implementors should take appropriate measures to protect their interfaces against such vulnerabilities.

Figure 3.9: This JavaScript code starts multiple invisible WebRTC sessions. It then associates an IdP for each of them and extract their generated identity assertion.

```
IdPArray.forEach(function(idp){
    var pc = new RTCPeerConnection()
    pc.setIdentityProvider(idp.domain, idp.proxy)
    pc.getIdentityAssertion()
    .then(res => alert("Got your ID token: "+res))
})
```

The only visible hint to the user that a WebRTC session is starting happens when the browser request consent to share video and audio input. But it is also possible to start a WebRTC session without any attached media, as in our example in Figure 3.9. In this case, the session is invisible for the user without looking into the browser WebRTC session statistics. The Interactive Connectivity Establishment (ICE) IP leak issue (see Section 2.2) is due to the same weakness where the user may not see and control the start of a WebRTC session. A solution implemented to fix this issue is to tie the implicit sharing of private Internet Protocol (IP) addresses with the explicit authorization granted to the `getUserMedia` interface [92]. The same solution could work to implicitly authorize the sharing of identity information for legitimate WebRTC session. However, it may prove too restrictive for some use cases such as authenticated data sessions. An alternative solution would be to have the browser explicitly manage consent for authentication.

3.2.2 IdP in a Central Position

The WebRTC Identity Architecture replaces the users' trust in CS by trust in IdP to ensure secure communication through untrusted signalling. Although IdP already occupy a central role on the Web, they gain the ability to track any user call covered by an identity assertion.

Indeed, the IdP Proxy code is deployed on both sides of the call and running in the context of its IdP's origin. It is also subject to the same sandbox restrictions whether on the identity assertion generation or validation side. This implies that an IdP Proxy can place and read cookies on a user's User-Agent (UA) verifying an assertion. Note that the user verifying the identity assertion does not actively access the IdP, it only receives a call offer or answer containing an identity automatically verified by the browser. The IdP Proxy reading cookies could allow the IdP to track a user's call history. Eventually, even an identity unknown to the IdP could be linked with a known identity, and the call history versed into an existing profile. The IdP can also track user calls across multiple CS, as long as they use, or are called with the same IdP.

Reusing the classification proposed by Vapen et al., presented in Section 2.2, this ability to track user call history could at least be classified as $R+$. It could even be classified as $R++$, the highest privacy risk level, with regards to the verifying user's data as it would fall under the *Friend's data* class of privacy risk. However, users may not have authorized or even be aware of such tracking capacity. It is interesting to note that the information sharing relation is here reversed between the IdP and the CS. In a

classical authentication delegation, the user authorizes the IdP to share resources with the Relying Party (RP). However, in this case, the CS is in possession of call information and shares them with the IdP by adding it to the call setup.

3.3 Why Can't Users Choose their Identity Providers on the Web?

In Section 3.2, we presented how the WebRTC identity architecture may be used by IdP to track user calls without explicit user's authorization or awareness. As shown by Vapen *et al.* [149], on the web, users are presented with a very limited choice of IdP when signing in with authentication delegation. They reported that 47% of observed RP offer only one IdP, and only 19% offer four or more IdP. Due to the identity continuity constraint[7] the same limitation on offered IdP choices applies to WebRTC IdP. Users may have more trust in IdP whose business model does not rely on selling personal data or which they would host themselves [153]. However, current Single Sign-On (SSO) implementations do not permit users to make this choice.

Several reasons could explain that users choice is limited by the decision of the CS. Verifying the validity of these reasons would reveal if "we can let users chose actors they trust to participate in the communication setup" (RQ3) and eventual solutions to achieve this. In this study we thus address the following questions:

- **RQ3.1**: Do RP require specialised API?
- **RQ3.2**: Is dynamic discovery and registration commonly available for RP?
- **RQ3.3**: Do RP requires a trust relationship with the supported IdP?

7: We presented the identity continuity constraint in Section 1.2 which implies that the IdP set by the CS for the communication session would be the same as the one the user would have previously chosen to authenticate to the CS.

3.3.1 The Study: OAuth Request Collection

To answer the first of our research questions we implement a browser extension to parse the browser navigation history and look for OAuth 2 and OIDC authorization request URL. These requests can be identified as OAuth 2 requests by observing the presence of keyword parameters. They contain information identifying the RP making the request, the IdP to which the request is destined and the scopes requested by the RP.

```
https://accounts.google.com/o/oauth2/auth?
     client_id=74[...].googleusercontent.com&
     response_type=code&
     redirect_uri=http://www.dailymail.co.uk/
     registration/signin/google.html&
     scope=email+https://www.googleapis.com/
     auth/plus.login&
     [...]
```

Figure 3.10: Example of an OAuth 2 request collected by our extension.

The URL in Figure 3.10 is an OAuth 2 request for an `accounts.google.com` (URL Domain Name) authorization following the OAuth 2 code flow (`response_type=code`). The request comes from the OAuth 2 client `74658[...].apps.googleusercontent.com`, also identifiable through the `redirect_uri` parameter as `dailymail.co.uk`. Comparison of redirection Uniform Resource Identifier (URI) domains allows associating client from several IdP to a single actual RP. For instance, this client and the `facebook.com` client registered as `146[...]95` both share the same `redirect_uri` domain name. The requested scopes are `email` and `https://www.googleapis.com/auth/plus.login`.

Note that these URL do not contain private information regarding the user as they are accessed before the user is identified or authenticated.

To collect data, we manually visit each of the top 500 websites from the Alexa ranking[8] and try to use each one of the SSO solutions offered by these websites[9]. The visited URL are recorded into the browser history and then parsed by our extension. In Section 3.3.2 we present the results of our study and use them to answer RQ3.1. Finally, we use data on collected RP and IdP to answer RQ3.2 and RQ3.3 in Sections 3.3.3 and 3.3.4 respectively.

8: `http://www.alexa.com/topsites`

9: Our initial objective was to share the extension to a large panel. We later switched to the Alexa ranking as our panel was too small. However, a large-scale study could deliver interesting results, particularly regarding smaller websites.

As our data-collection search is focused on OAuth 2, we do not collect requests for IdP using other protocols, *e. g.* Twitter and Amazon. However, we scarcely encounter RP offering only these IdP, as a result, the large majority of visited RP is captured by our extension. In total, we observe 103 unique RP and 23 OAuth 2 provider's domain names. The two biggest observed IdPs are undoubtedly Facebook and Google, respectively serving 63 and 52 RP. The third most observed IdP is Twitter with 30 request URL, but it uses OAuth 1 and as such is not included in our data.

While our results confirm the claim that users are offered a limited choice of IdP, interestingly we also observe some variations of the number and domain of supported IdP based on RP geographical origins. Firstly, Chinese websites only offer to log in through Chinese IdP such as QQ.com[10]. Occidental websites, *i. e.* North-American and European, mostly offer to log in through one or both of the top two IdP, sometimes with a third solution. Finally, Russian websites offer the largest number of solutions as they include Russian IdP, *e. g.* VK.com, and occidental IdP often not limited to the top three providers. Other regions were not represented in sufficient numbers to draw any conclusion.

10: Chinese IdP also vary by the type of authentication mechanism as they often use phone authentication through a QR code.

3.3.2 RQ3.1: Do RP require specialised API?

RP not only require user authentication but also authorization to access protected resources. These resources may vary in nature, and may not share a standardised data format when of the same type. One reason for RP not to allow signing in with any IdP could be that they require specific resources, which are not available on any IdP.

We classify RP in three categories: *Authentication*, *Profile*, and *Specialised*, in function of the scope observed in OAuth 2 authorization requests. Authentication classed RP only require an assertion that the user got authenticated by the IdP. Any IdP could serve such RP, given a standardised assertion (*e. g.* a signed JSON Web Token (JWT)). This authentication assertion may also contain the user identifier for the RP's use. RP classified as Profile require basic user profile information in addition to an authentication assertion. These pieces of information are often used to complete their own database and provide a personalised user experience. IdP would need to give access to resources in a standardised data format in order to avoid a specific implementation on RP side. Finally, Specialised RP require specific resources, for instance, write access on a user shared repository. By definition, services provided by RP of this class are specialised to use resources from a few IdP, each one requiring its own implementation. They cannot be generalised to cover a broad range of IdP.

Figure 3.11 presents these classes, ordered by specificity. Indeed, access to a particular API is more specific than accessing generic profile information. Similarly, getting access to a username and phone number is more specific than authentication, which could be abstracted to a boolean, *i. e.* authenticated or not. Although special cases may exist, *e. g.* access to authorized resources without authentication of the user, this classification allows us to define which RP could, in theory, accept any IdP, and which one would be bound to a particular API.

We classify each RP-IdP relationships, *i. e.* each collected `client_id`, into one of our three classes. We evaluate from the available documentation if the requested scopes give

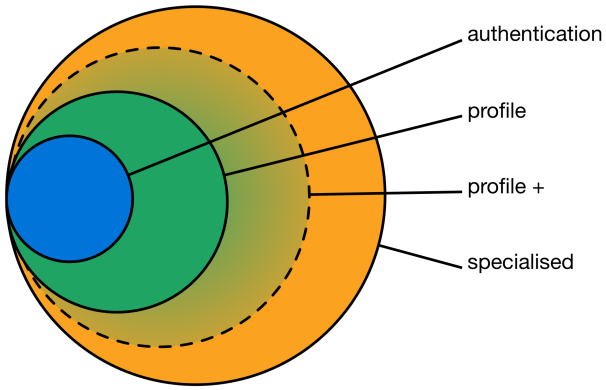


Figure 3.11: Classification of RP-IdP relationships.

access to a specialised API, a profile information API, or an authentication assertion. Differences between classes may be blurry, for instance, email and friends list can be considered as user profile information. However, email alone is often used as a unique user identifier in conjunction with a proof of authentication. OIDC defines a list of user-information claims and their types, we consider this as the standard for user information. Other types of information and API, *e. g.* friends lists or cloud access, are non-standard and not available from every IdP. We classify these requests outside of the Profile and Authentication classes as Specialised. Ultimately we define our classes as follows:

- **Authentication:** requests only subscriber identifier scopes, *e. g.* the `openid` or `email` scopes.
- **Profile:** requests scopes for user informations equivalent to OIDC's standard claims.
- **Specialised:** requests any scope not classified as either Authentication or Profile.

Some RP offer to sign in with multiple IdP and may require different types of information for each IdP. As a result, they may be classified differently for each of their implemented IdP. Based on common `redirect_uri` domain names, we regroup clients into unique RP. For each RP we attribute a minimum (MIN) and a maximum (MAX) classes, noted MIN/MAX. As an example, a RP classified as Authentication/Profile requires to access profile information on some of the supported IdP options, but only requires an authentication proof and a unique identifier on at least one offered IdP. RP offering only a single IdP are classified as MIN/-.

Result analysis

Our observations, summarised in Table 3.2, reveals that a majority of RP, 58 % of 103, are classified as Authentication or Profile. MAX-classed RP, 40 % in total, are the one providing support for multiple IdP. In our observations, 24 out of 56 RP with a Specialised class are classified with a MIN class of Authentication or Profile. This double classification demonstrates that while some websites require Specialised services, they also adapt to support other IdP offering fewer resources. Note that this also leads to different privacy risk level for the user. On the other hand, 34 % of observed RP are rated as MIN-Specialized.

The large majority of Specialised RP use resources from Online Social Network (OSN) such as friends list, user likes, and extended profile information. While data from different social networks can be similar from a conceptual point-of-view, the format and API used to access these data depend on the functionalities and concepts offered

Table 3.2: Observed relying parties' classes
Some RP offer to sign in with a single IdP, we classify them with a single class, *e. g.* Profile/-. Other RP offer multiples IdP, their relations with these IdP may belong to a single class, *e. g.* Profile/Profile, or different classes, *e. g.* Auth/Special. For these RP we show the minimum and maximum classes they offer. Risk class refer to privacy risk classes as defined by Vapen et al. (see Section 2.2). When a RP has different risk class due to multiples RP-IdP relationships, we give an interval for the Risk classification.

Min/Max Classes	Observed	Risk class
Authentication/-	10% (10)	$R-$
Authentication/Auth	1% (1)	$R-$
Authentication/Profile	9% (9)	$R-$
Authentication/Special	6% (6)	$[R-; RA+]$
Profile/-	13% (13)	$R-$
Profile/Profile	2% (2)	$R-$
Profile/Special	17% (18)	$[R-; RA+]$
Specialised/-	26% (27)	$[R; RA+]$
Specialised/Special	5% (5)	$[R; RA+]$
No Scope	11% (11)	
Total	100% (102)	

by social networks. In these cases, the user fully depends on implementation choices by RP. Although being a standard protocol, OAuth 2 lets IdP define their scopes and data formats. As a result, IdP may provide similar results in different scopes and data formats. For instance, we observed six different scopes for email access and seven different scopes for basic profile information. This gives an indication on the lower bound of the implementation work that RP must complete to support these IdPs.

OIDC solves this problem by standardising basic profile claims (*e. g.* profile, email, address, ...) as well as the scopes, endpoint, and data format to retrieve these pieces of information. From our observation, only four IdP out of twenty where implementing OIDC, as reported on Table 3.3. Notably, Google is one of the OIDC providers and serves 50 % of observed RP, while Facebook does not implement it and serves 63 % of observed RP. Other IdP serve less than 6 % of RP each. However, out of the 52 RP using Google's SSO, only 22 request standard OIDC scopes, and from these 22 only 10 request OIDC ID Token. Surprisingly, we also observe that out of the 34 RP using the Google API scopes, 19 were using deprecated scopes.

RP of MIN-Authentication and MIN-Profile classes represent 58 % of all observed RP. We defined these classes as being equivalent to claims covered by OIDC requests. It appears that from our observations, the quantity and type of information shared under these scopes would be sufficient to login into a majority of websites. We thus conclude that current standards for API and data format do not appear to be a hindrance to SSO interoperability. However, there is a clear lack of implementation of these standards from big IdP, *e. g.* Facebook not implementing OIDC or Twitter not implementing OAuth 2. But the lack of implementation effort also comes from RP. A non-negligible proportion of them did not update their SSO implementations to the latest versions. Implementation cost may be a reason, but we also note that studied websites come from the most 500 visited websites. They should have the resources to update their implementations.

Regarding scopes for sharing OSN data such as friends list, we observe that 43% of RP using OSN data also implemented other IdP without requesting OSN data. This is even the case when such data would be available from the IdP, *e. g.* with the `accounts.google.com` API. On one hand, a standard format for OSN data could allow more interoperability between RP and OSN based IdP. But on the other hand, since

the RP can accept non-OSN IdP, sharing of OSN data appears to be non-mandatory. The possibility to opt-out of consent for data sharing is however often not offered or not clearly advertised.

3.3.3 RQ3.2: Is dynamic discovery and registration commonly available for RP?

In order to support sign-in on an RP with any IdP, the identity protocol must offer discovery mechanism. This mechanism allows the RP to find the IdP's protocol parameters from a URI provided by the user. Depending on the protocol this URI may be for instance the user identifier or a resource location on the IdP. Without discovery, the RP may not know which endpoint to use on the IdP, or which public keys and algorithms to use to verify information provided by the user or the IdP. Additionally, the protocol should also allow interactions between IdP and RP without prior manual configuration. Some protocols require the RP to possess credentials to be authenticated by the IdP. In this case, a dynamic registration process must take place before any further interactions.

For instance, OAuth 2 recommends that the RP possesses a client identifier and secret for authentication in order to retrieve an access token. The use of an unregistered client is not excluded by the specification, but our investigation did not reveal any use of this. Similarly, OIDC requires the RP to be authenticated to get access and identity tokens. The current OAuth 2 version does not specify dynamic registration mechanism, but OIDC optionally offers discovery [154] and dynamic registration [155]. OIDC discovery uses Web Finger [156] to find user's IdP from the user identifier and standardises IdP's metadata location. Metadata are in turn used to find, if available, the dynamic registration endpoint. However, both discovery and dynamic registration extensions are optional. Request For Comments (RFC) 7591 [157] proposes to generalise the specifications of OIDC discovery and dynamic registration to the broader OAuth 2 specification.

Table 3.3, summarizes the observed OAuth 2 IdP. In total, we collect 23 unique IdP domain names. Out of those, 15 are not requested with a scope parameter and are not included in our OIDC features investigation. For instance, the `rambler.ru` request to `instagram.com` does not contain a scope parameter, but Instagram still asks some authorization to the user. On the other 18 unique IdP domain names, only 5 are used with a `openid` scope. This indicates implementation of OIDC.

The standard end-point for OIDC configuration metadata is accessed on the path `/.well-known/openid-configuration`. We access metadata for each observed OIDC provider. In total, only two out of five offer `openid-configuration` metadata. None of these metadata defined dynamic registration and we neither found support for Web Finger. We are not able to test Web Finger for every IdP as some IdP do not offer a look-up compatible user identifier[11].

11: As explained in Section 3.1.3, an email identifier may not have the same domain name as the IdP. In this case, Web Finger look-up would discover the email domain rather than the IdP's one.

Again, existing protocols offer discovery and RP registration capabilities but implementations are missing these optional functionalities. This mechanism is nonetheless compatible with manual configurations of IdP. As such RP could nonetheless implement it to demonstrate interest and support IdP allowing dynamic registration. However, this would impose an additional component to add to the login page and an associated implementation. Users would also need to know and enter their OIDC identifier or provider for the discovery mechanism, which may not always be obvious. There is clearly a usability limitation compared to the "click to sign in" use of IdP button.

IdP	RP	OIDC	Metadata
www.facebook.com	63	×	×
accounts.google.com	52	✓	✓
oauth.vk.com	6	×	×
graph.qq.com	5	×	×
login.live.com	3	×	✓
account.live.com	3	×	×
www.linkedin.com	3	×	×
connect.ok.ru	3	×	×
login.microsoftonline.com	1	✓	×
services.Adobe.com	1	✓	×
github.com	1	×	×
feedly.com	1	×	×
www.livejournal.com	1	×	×
connect.mail.ru	1	×	×
open.weixin.qq.com	1	×	×
api.weibo.cn	1	×	×
mixi.jp	1	✓	×
oauth.riotgames.co	1	✓	×
Total	103	5	3

Table 3.3: Observed OIDC and discovery features Implementations

3.3.4 RQ3.3: Do RP requires a trust relationship with the supported IdP?

The decision by the RP to implement a particular IdP may rely on a trust relation. For instance, a RP may expect verified profile information from a governmental institution or a secure authentication process with a two-factor authentication. Authentication Assurance Levels (AAL) are usually used to characterise the strength of an identification process during the user enrolment and authentication. Similarly, an IdP may trust a RP, or more particularly monetise access to an API. This implies that RP and IdP got into an agreement involving registration of payment methods which is out of scope of existing dynamic registration specification.

Trust relations can be either implicit or explicit. Implicit relations are difficult to characterise as they are not clearly visible from the user point of view. For instance, the web site `service-public.fr`, which provides direct access to French governmental service, only offers to log in with an account from other public services such as the tax department, the social security service, or the national postal service. Other implicit trust relations may be due to strategic decisions, for instance limiting RP access to IdP of the same company. This is the case for `developer.microsoft.com`, as it only allows login through the `login.windows.net` IdP, both being Microsoft's services. In these scenarios, the RP would not allow the user to choose any IdP.

Explicit relations may be more easily characterised as the RP clearly request a solid information. For instance, Orange's OIDC IdP offers the scope `form_filling`. This scope substitutes to OIDC profile scope and allows the RP to access qualified Orange information, for instance, the telephone number linked to the user subscription. Such relation may either fit into the Specialised or Profile class, depending on the RP will

or capacity to accept generic information, such as standard OIDC profile. We classify these relations as profile+, as shown in Figure 3.11. Similarly, OIDC RP can verify if the email was verified by the IdP through the `email_verified` boolean value. As this verification is done on the server side, it is not visible from the user point of view. Explicit trust relations may also be linked to a contract between the RP and IdP. Note that other common scope may also be subject to an agreement, though it is impossible to determine without investigating actual IdP API terms of use.

Trusting a presented identity and the associated authentication process is a complex, and sometimes subjective matter, especially on the web. For instance, social networks such as Facebook and services using them, claim their identity to be trusted. These claims are backed by social relationships, evaluation between users, or real-name usage policies. But stricter organisation, such as governments and banks, would not accept these identities. OpenID Connect allows requesting a specific authentication level with the Authentication Context Class Reference (ACR) parameter, referring to the ISO AAL. Data collected during our OAuth 2 investigation did not reveal `acr` parameter usage in OIDC requests. Out of the hundred and three observed RP, we estimate that fourteen have an implicit trust relationship with their IdP. In most cases, these RP only offer a single compatible IdP from the same company. Four of these RP are also classified as Specialised/-, indicating that they would not be able to accept any IdP in the first place. We found no occurrence of an explicit trust relationship in our panel.

It appears difficult to judge if trust is an issue that would impose manual configurations of IdP/RP relationships. To some extent, a decision to support a particular IdP can be considered as an implicit trust decision. But whether RP are willing to trust other IdP in order to offer more control to their user remains an open question. It seems to us, that a solution to simplify the discovery and registration of IdP endpoint should nonetheless give RP the option to control the range of compatible IdP and the authentication strength for trust reasons.

3.3.5 Developer Survey

To further investigate the reasons why some IdP are implemented rather than others, we want to know the developer point of view. Of course, the choice of an IdP depends on a lot of other factors than the developer's opinion. First of all the actual project and its constraints are decisive parameters. But we believe that developer's preferences also play an important role in the technology and providers chosen for implementations.

We conducted a survey during the BreizhCamp 2017, a regional conference targeted at developers from local information technology companies. During this conference we distributed questionnaire (see Figure 3.12) on tables and at our own exposition booth [12]. The questionnaire consists in rating the value of six properties of IdP in the developer's opinion from 0 (low value) to 5 (high priority). The properties to rate are as follows:

12: We participated to the BreizhCamp as members of the reThink project which was sponsoring the event.

- The **strength of the enrolment** process.
- The **strength of the authentication** process.
- A popular IdP would have a large **user base size** and allow many users to connect to the developer's website.
- The ability for the developer's website to access IdP's **API and users' data**.
- The **implementation complexity**, *i. e.* maturity, trending technology, documentation availability, SDK availability, ...
- The **user experience** when enrolling and authenticating with the IdP.

Service Trust in Identity Providers

This forms purpose is to collect the properties service developers are expecting from an identity provider they would trust.

Rank the following by level of priority:

If/when you had to implement an authentication/authorization delegation solution, what features would be/were decisive in your choice to implement a particular identity provider.

1. **Strength of the user enrolment process ?**

	0	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	◯	High

2. **Strength of the authentication process ?**

	0	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	◯	High

3. **Size of the user base ?**

	0	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	◯	High

4. **Access to API and users data ?**

	0	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	◯	High

5. **Implementation complexity ? (maturity, trending technology, documentation availability, SDK, ...)**

	0	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	◯	High

6. **Quality of the user experience ?**

	0	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	◯	High

Your Actual Experience in the field of Identity Management

7. **What is your profile?**
   - ◯ Developer
   - ◯ Commercial
   - ◯ Architect
   - ◯ Academic
   - ◯ Student

8. **What is your familiarity with the concepts of Authentication/Authorization Delegation?**

	1	2	3	4	5	
End-user	◯	◯	◯	◯	◯	Expert

9. **In which role did you already worked on Authentication Delegation?**
   - ◯ As an Identity Provider.
   - ◯ As a Client Service.

Figure 3.12: BreizhCamp Developer Survey

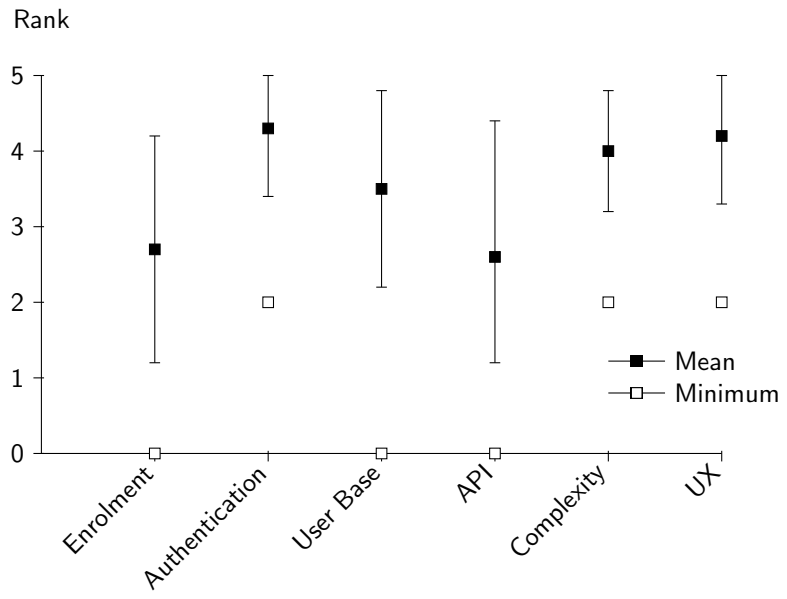


Figure 3.13: Mean, minimum, and standard deviation of the trust survey results.

Participants are also required to rate their experience in the field of identity management. This allowed us to discard a single commercial profile and only consider answers from technical profiles. In total, we thus have 24 exploitable surveys. All considered respondents are at least familiar with identity management as end-users, and most are capable of quoting some protocols.

Figure 3.13 shows the minimum, mean, and standard deviation for each parameter of our survey. We observe a separation of parameters into two main categories:

The first category regroups properties with a high-value and a low standard deviation: *i. e.* authentication strength, implementation complexity, and user experience. All parameters in this categories demonstrate a minimum rank of 2, a high mean value close to 4, and a standard deviation under 1. In our opinion, this shows that a consensus from developers exists on the importance of these properties. This is easily explained as developers have all in common the core task of implementing practical and secure websites, with library they are able to use.

On the contrary, enrolment strength, user base, and access to API all show a low value and a high standard deviation. The enrolment strength and access to API parameters have a low mean value of 2.6, and a high standard deviation of 1.5 and 1.8 respectively. The need for access to an API is highly dependent on the website use-case, which may explain the absence of consensus. Enrolment strength is also dependent on use case, but from discussions with participants who completed the survey at our booth we also believe that this property may not be as well understood as the authentication strength. Finally, in the low-value category, we can further distinguish the importance of the user base which shows a higher mean of 3.5 and lower deviation of 1.3. This may indicate an important but not critical property. Supposing a developer would like to implement multiple IdP, even with small user-base, this property would enter in conflict with the quality of the user experience and with the implementation complexity, which are clearly higher priority properties.

Based on our results, we argue that a solution to let users choose their own IdP should preserve a simple user experience, and ease the implementation complexity. Whether the website needs to trust the IdP remains an open question. On one hand, the enrolment process is not seen as a priority, which may mean that self-asserted identities may not be a problem for some websites. On the other hand, the authentication strength is clearly

a top priority. We thus also argue that a solution to let users choose the IdP should probably let websites request trusted IdP or AAL.

3.4 Summary

The WebRTC security architecture [47] claims that trust in the signalling layer can be replaced by trust in the IdP. However, in our state of the art (see Chapter 2) we did not observe research considering the IdP as a potential threat in WebRTC scenario. In this chapter, our contribution was to study additional privacy implications focusing on the WebRTC identity architecture and on the role of the IdP.

We first presented our implementation of the WebRTC identity architecture and in the particular the integration of the IdP Proxy component with the OIDC protocol. This work reveals that while OIDC facilitates the creation and signature of WebRTC identity assertion its integration is not straightforward. In particular, although WebRTC offers an abstract authentication delegation interface it is not particularly suited to manage authorization delegation. We thus answer to RQ1.1 by showing additional privacy risks that IdP should take in consideration to implement the WebRTC identity architecture. We also show how the IdP can compromise users' privacy without their explicit consent. The central role and responsibility of IdP in the web ecosystem is reinforced by their inclusion in WebRTC call setup.

We then focused on answering RQ3 to find if we can let users chose actors they trust to participate in the communication setup. Previous studies reported that users are presented with a limited choice of IdP when authenticating on the Web. We conducted a survey of the top-500 website's usage of OAuth 2 and OIDC to identify possible reasons for this situation. We classified RP by the types of authorization they request to users. Our results show that a majority of RP, 58% of 103, do not require specialised data. Although OIDC proposes standardised profile claims, scopes, endpoint, and data format, we observe that it is implemented by only a few IdP. Similarly, OIDC offers optional dynamic discovery and registration of RP but these features are not implemented at all on surveyed IdP. That RP and IdP require pre-established trust relations could explain this lack of implementation. As we focused our survey on interactions observable from the user's browser it does not allow us to answer on that matter. To further investigate this question, we believe that it should be directly asked to the professional responsible for the deployment of IdP and RP alike. Finally, we conducted a survey on developer to identify important IdP's properties in the developer's opinion. Our results show a consensus on the need for a strong authentication and well-crafted user experience but not on other properties. We answer on RQ3.1 and RQ3.2 that while technical solutions for allowing users to choose their IdP exist, these are not implemented by IdP and RP alike.

Chapter 4

Controlling the WebRTC Identity Parameters

The WebRTC identity architecture allows to bind the media session to a validated peer identity. In Chapter 3, we presented privacy issues related to this specification and to SSO on the Web in general. From our point of view, the fact that users lack control over which identity services they want to use on the Web is at the heart of these privacy issues. Furthermore, users may also ask themselves whether they should trust their peers' IdP and their peers' authentication strength? In order to raise users' trust in a communication, both from a privacy and a security perspective, we propose to give users more control over WebRTC identity parameters. Firstly, in Section 4.1, we define a SDP extension to negotiate the other peer's IdP and authentication strength during the SDP call setup. We implement our solution which shows that such negotiation mechanism would benefit from more flexibility in the choice of IdP for peer authentication. Yet, as we explained previously, IdP choices are limited on the Web. The WebRTC identity specification poses some interesting concepts but for the limited scope of user-to-user authentication. We are interested in studying how it can be extended to support more use cases. In Section 4.2 we present WebConnect, a prototype for an Identity Metasystem API based on the WebRTC identity specification. We show that on the long run, adoption of such an API could reduce the implementation burden for websites developers and allow users to preserve their privacy by choosing trusted IdP.

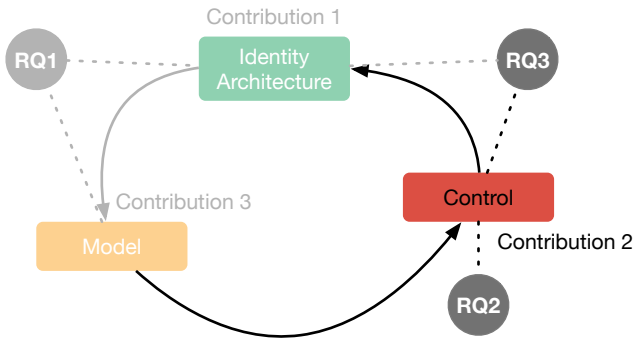


Figure 4.1: Overview of our Contributions: Controlling the WebRTC Identity Parameters.

4.1 An SDP Extension to Allow Identity Negotiation

The WebRTC identity specification allows binding the media session to a validated peer identity. However, indications as to whether the Identity Provider (IdP) and identity assertion should be trusted are limited[1]. Alice may ask herself the following questions: "should I trust my peer's IdP" and more importantly "what is the strength of my peer's authentication"? In order to address RQ2 "can we act on a WebRTC session to raise the trust and security level", we propose to let users negotiate their identity parameters. In this section we thus address the following research question:

1: The only guarantee is that the IdP Proxy is loaded over HTTPS.

- **RQ2.1**: How to let users negotiate the other peer's identity parameters?

As a peer's authentication strength is known by and depends on the peer's IdP, trust in the security of the communication depends directly on Alice's trust in her peer's IdP. Though Alice may not be capable to evaluate this IdP's trustworthiness, she may not have previous relations with this IdP or, just not enough observable feedback to judge. Recommendation from a trusted source could solve Alice's lack of knowledge and allows for identity parameter negotiation.

4.1.1 Recommendation Sources

Three actors, potentially trusted by the user and already involved in the WebRTC communication setup, would be well-suited to recommend trust in the peer's identity and conduct the identity negotiation. These are the user's Communication Service (CS) and/or its associated client, the user's IdP, and the browser. However, each of these actors has a different role and visibility on WebRTC identity management. We use the following properties to evaluate where to best place the negotiation responsibility:

- Trusted by the user or not.
- Capability to recommend trusted IdP.
- Knowledge of the call context.

Supposing that Alice is allowed to choose a trusted IdP, we can consider that Alice's IdP occupies a trusted position in the communication setup. As an IdP, Alice's IdP would also be well suited to understand and to evaluate other IdP's trustworthiness. However, web IdP are organised in silos and do not interact with each other. In order to serve as recommendation sources, they would need to build trusted identity federations, circles of trust, or individual trust relations. Besides, IdP are not aware of the call context (used call service, risk level, etc.) and only deal with user authentication. Configuring user preferences for call security depending on call context may prove to be difficult.

Conversely, the CS is fully aware of the call-context and it would be quite simple for users to define separate preferences for each CS. CS are also dealing with IdP to authenticate their own users. Evaluating the trustworthiness of an authentication from an IdP they already trust would not be different. However, CS trusting IdP that they do not implement, *i. e.* without explicit configuration and contractual agreement, would also require a kind of trusted federated identity service. CS may also not be trusted by their user and this is the reason why the WebRTC identity specification exists in the first place. Even if the user's CS is trusted, the WebRTC identity architecture may still be relevant in interoperable scenarios as these involve multiple CS.

Finally, the web browser may also be an adequate actor to evaluate the trustworthiness of the other party authentication. Being the WebRTC Trusted Computing Base, the browser is considered to be trusted. It has also some knowledge of the call context,

Actors	Trusted	Recommendation	Context
Identity Provider	+	++	-
Communication Service	-	+	++
Browser	++	-	+

Table 4.1: Comparison of communication setup actors to act as an identity recommendation source.

although less than the CS, and could easily be configured by the user. In the current specification, the browser is in charge of IdP's origin validation through HyperText Transfer Protocol Secured (HTTPS). This already constitutes a kind of low-level trust recommendation towards asserting identity assertion security. However, browsers are not usually dealing with authentication strength and IdP trustworthiness.

Table 4.1 summarises our simple comparison of the IdP, CS and browser as trusted recommendation sources. None of these three actors appears clearly more suited to evaluate and negotiate over the other party authentication. CS could implement such functionality as the need arises without waiting for a standardisation process. However, this raises the question of whether or not the CS can be considered as part of the Trusted Computing Base (TCB) for WebRTC call. Trusted CS would probably be more suited for enterprise scenario, but not in the general web ecosystem. For a more generic identity negotiation implementation working on any WebRTC service, the web browser would be the best suited to implement such functionality. Indeed, the browser is already responsible for verifying identity assertion but also for the management of Application Programming Interface (API) and plugin authorizations.

4.1.2 SDP Extension

As we described in the previous section, the security of a WebRTC session depends on the peer's authentication strength and the trustworthiness of the IdP asserting the peer's identity. Supposing that knowledge of these two parameters is available, a user may act on the session to negotiate a higher security level. This would either be done by requesting a higher authentication level, or an identity assertion from another IdP.

To convey these requests in the negotiation we define two identity parameters. The Authentication Class Request (ACR): `List<ACRValue>`, a list ordered by preference of accepted authentication class values. And the Origin Request (OR): `List<Origin>`, a list ordered by preference of accepted IdP's origins. The identity assertion is transmitted in Session Description Protocol (SDP) offer and answer as the `a=identity` session level attribute (see Section 1.3). Defining extensions to this attribute in order to convey identity parameter requests would be possible. However, the `identity` attribute grammar specifies that an extension must follow a valid identity assertion. This implies that it would not be possible to negotiate identity parameters without providing an identity assertion. However, anonymous calling is often cited as an important use case for communication services [3, 6]. The need may arise for a user to negotiate identity parameters while being anonymous.

We instead propose to define a new type of SDP session-level attribute to negotiate these identity parameters. The Authentication Class and Origin Request (ACOR) SDP attribute defines a list of accepted authentication class and IdP domain for the other peer identity.

- `a=acor:LIST<ACRValue> ; List<Origin>`

An SDP negotiation is a sequence of offer and answer messages, with an offer always followed by an answer. We detail two scenarios where the negotiation for identity pa-

rameters either starts from an offer or from an answer. To accept the requested ACOR attributes, a peer must thus reply a SDP message with a compatible identity assertion.

SDP Offer with Identity Request

Figure 4.2 shows a sequence diagram of the offer scenario. In this scenario, Alice's browser (UA_A) gets negotiation parameters from Alice's Recommendation source (R_A). Upon receiving the SDP offer and the requested identity parameters, Bob's browser (UA_B) checks whether the requested IdP Origins are acceptable. We do not detail the way to get this information or how to select the IdP, but the solution would probably rely on Bob's Recommendation source (R_B) and registered IdP in UA_B. If no requested origin is available, the browser must answer with a SDP containing no identity attribute. Otherwise, it proceeds to request an assertion from the selected IdP. If no requested ACR matches the current authentication level a standard `IdpLoginError` is returned. If this error does not contain a `loginURL` parameter, then no requested Authentication Context Class Reference (ACR) are supported by the IdP and a SDP with no identity attribute is returned by UA_B. However, if the error contains a `loginURL`, the user can be authenticated with a procedure matching one of the ACR by following the provided login Uniform Resource Locator (URL). Once this standard authentication procedure is done, Bob's Assertion (ASSERTION_B) is returned in the SDP answer.

On receiving ASSERTION_B, UA_A asks Bob's IdP, through it's own IdP Proxy instance, to validate the assertion. The `IdentityValidationResult` returned by the proxy should contain the assertion ACR value, allowing UA_A to open the media channel.

SDP Answer with Identity Request

The answer scenario is similar to the offer scenario but differs in that the assertion is received before the receiving user sent any identity request. Figure 4.3 shows a sequence diagram for this scenario. Upon receiving the offer, UA_B gets identity parameters from R_B. If the received assertion origin is accepted, UA_B checks ASSERTION_A through a local instance of IdP Proxy A. If one of the identity parameters does not match with the received assertion, UA_B returns a SDP answer with its identity parameters request. This would trigger a new offer from UA_A if possible.

Alternatively, UA_B could accept the first offer and later send a new offer to renegotiate A's identity.

4.1.3 Validation on the current specification

Both browser and CS get access to SDP messages during the call setup and could support identity negotiation in a similar way. The browser would be more difficult to modify than a WebRTC JavaScript client due to its large code base. In contrast, implementation of identity negotiation at the IdP level would be something quite different. As the IdP does not actually see SDP messages, the WebRTC API would need to be heavily modified. This kind of modification to the specification is out of scope of our research[2]. Our objectives for implementing our solution to negotiate identity parameters over SDP are the following:

2: The definition and evolution of proposed standard is a heavy process. A constraint of our work is to aim for compatibility with existing specification rather than proposing heavy modifications.

- Evaluate the complexity of deploying this solution.
- Validate the feasibility of an implementation given the current state of WebRTC implementation on web browser.
- Verify that adding identity parameters does not compromise inter-operability with other services.

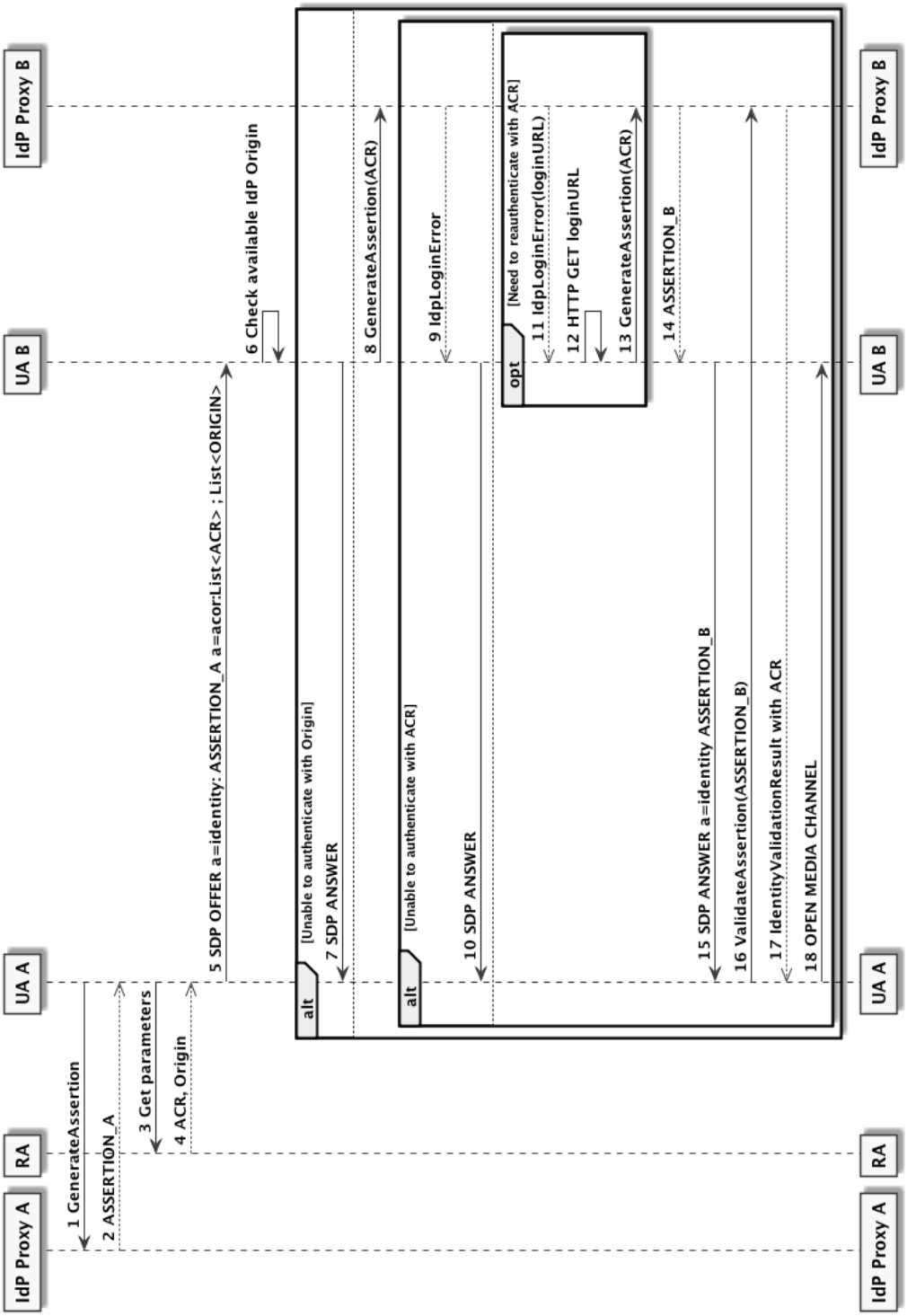


Figure 4.2: SDP Offer

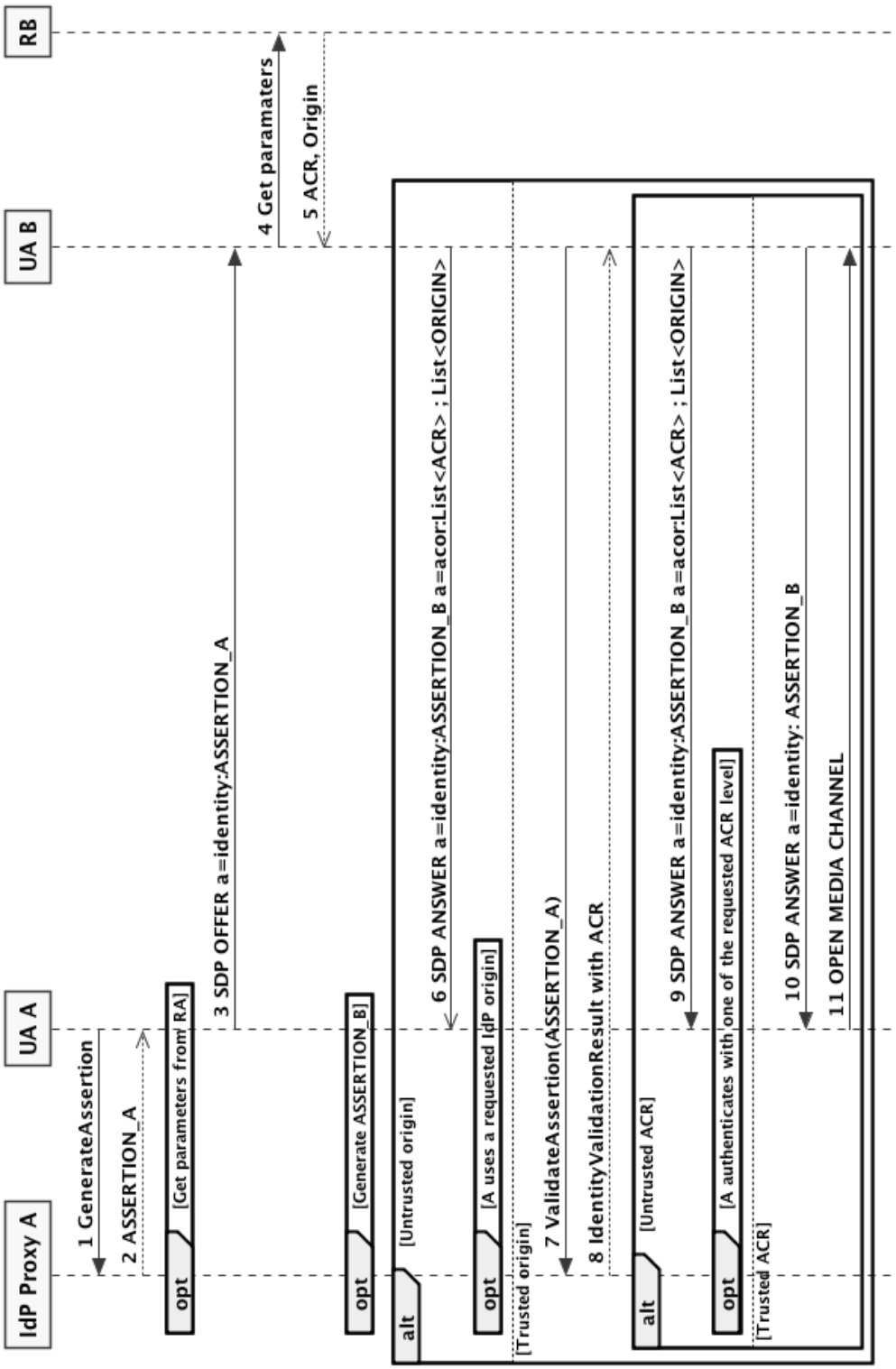


Figure 4.3: SDP Answer

Parameter	Can be set (input)	Can be verified (output)
Origin Request	✓	Specified
Authentication Class Request	✓	×

Table 4.2: Capability to Implement the Proposed Solution

Ultimately, we implement the negotiation functionalities at the CS level and integrate it to our WebRTC service[3]. Figure 4.4 shows the identity request interface. The web interface allows both users to request new ACR and OR parameters for the other peer. SDP messages are returned by the `createOffer` and `createAnswer` functions offered by the `PeerConnection` object. Once generated, the client code appends an ACOR attribute to the generated SDP offer or answer. The SDP is then sent to the other peer's client. On receiving a message, the client code looks for the requested ACOR attribute, and at the same time verifies that the received peer-identity, the other peer's identity assertion, follows the request previously specified. The resulting negotiation solution is implemented in under 100 JavaScript code lines, for a very simple client. Renegotiation allows both clients to make new offer once the session has been established. For instance, this allows asking an anonymous user to authenticate itself. We, however, identify some important limitations, mostly due to the specifications.

3: Available at `https://github.com/Sparika/ACOR_SDP`

The `generateAssertion` function from the IdP Proxy has for parameters `contents`, `origin`, and `usernameHint`. Request for a particular authentication class to the IdP is thus not defined. We use the `usernameHint` parameters to pass ACR parameters to the IdP. The IdP is modified accordingly to understand this parameter. However, the `IdentityValidationResult` dictionary do not represent the ACR. Dictionaries are JavaScript objects that cannot be extended by adding new members. It is thus impossible for the validating IdP Proxy to return a certified ACR value to the client inside the `IdentityValidationResult`. In clear, even-though the client can request a particular authentication strength, it cannot verify that the IdP complied with the request. A solution to solve this issue could be to let the client directly read the identity assertion contained by the SDP message to check the asserted authentication strength. The downside of such solution is that it loses the benefits of the WebRTC identity abstraction as the client must be able to natively understand the identity assertion.

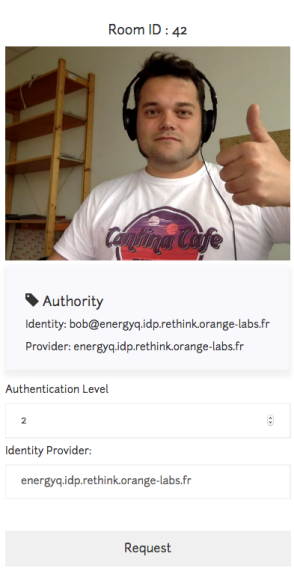


Figure 4.4: Identity negotiation interface on a WebRTC service.

It also appears impossible to change identity at call runtime or use multiples identities simultaneously. The WebRTC specification states that if the PeerConnection object has *"previously authenticated the identity of the peer [...], then this also establishes a target peer identity. The target peer identity cannot be changed once set"* [28]. Our tests demonstrate that modifying the remote peer identity effectively closes the connection. Once a first identity has been set, it cannot be changed. If this is an issue and if several IdPs are available, the peer should first wait to receive an ACOR attribute from the other peer, before setting an IdP for the session.

In the end and to answer RQ2.1, we are able to establish two anonymous sessions and then request the other peer to authenticate with a particular identity domain. We are however unable to verify the strength of the authentication, hampering the capability to request a particular authentication strength. In addition, our modified SDP messages are effectively ignored by other services and by the user-agent. Interoperability with other services should not be compromised by this new attribute. Our observations are summarised in Table 4.2 which shows for both OR and ACR parameters whether they can be passed as input to the assertion generation function and returned as output of the assertion validation function.

4.2 WebConnect

In Section 1.5, we presented some privacy threat mitigation techniques, amongst which is data minimisation. Authorization delegation frameworks give users some control over the information they want to share to other websites. Used as Single Sign-On (SSO) solutions, they put users in a situation where they have to choose between the burden of setting up a new account or sharing some private information. Allowing users to choose privacy-preserving solutions may be an incentive for providers to respect their user's privacy. However, we have shown in Section 3.3.2 that many websites abuse users' privacy by requesting unnecessary privacy-sensitive data. In addition, Vapen et al. [149] observed that in practice Relying Party (RP) offer few choices of IdP, an observation confirmed by our study (see Section 3.3.1). We thus believe that users should be given more control over which IdP they want to use when authenticating on the Web.

In Section 3.3, we consider some of the reasons that could prevent the implementation of IdP discovery mechanisms. We estimate that at least 58% of observed websites could make use of an IdP discovery mechanism. However, we also found several hindrances to the deployment of such solution:

- the lack of implementation of this feature by IdP and RP.
- the difficulty for users of knowing and entering their identifiers in the discovery process.
- the possible trust relations between the IdP and RP.

Several tentatives have been made to develop Internet's "Missing Identity Layer" [158], without clear success. A proposition of the "Seven Laws of Identity"[158] is that a single protocol would not fit all needs on the Web, but that "different identity systems must exist in a metasystem". This implies the need for a simple encapsulating protocol and a unified user experience. First released in 2007, Windows CardSpace[159] was an implementation of an Identity Metasystem [160] and Microsoft's solution to propose an integrated identity management experience to Windows' users. It allowed users to select InfoCards to prove identity claims, *e. g.* name, age, to requesting applications, as presented in Figure 4.5.

An identity aware web browser would have several advantages. In their 2011 empirical study, Sun et al. [152] observe users concerns about web SSO and recommend that "identity support into the browser [would provide user with] a consistent, intuitive and trustworthy user experience". We also note that our survey presented in Section 3.3.5 reveals that by default the authentication strength, the complexity of implementation, and the user experience are the most important factors for developers when considering to implement an SSO solution. A web Identity Metasystem should thus aim to offer these advantages to developers in order to facilitate its adoption.

Recent trends in web browser development tend to indicate that browser makers are looking to offer a complete browser experience. Personal preference synchronisation, official plugins bringing new functionalities[4], or simply access to an application marketplace. Particularly related to our interest, Google Chrome offers an OAuth 2 API[5] to Chrome Apps[6]. This API implies that users can use a Chrome interface to choose an identity, sign-in, and modify some of their information such as their profile picture. However, Chrome only offers integration with Google's identity.

The WebRTC identity specification also enriches the browser with a kind of identity management capability but limited to the scope of WebRTC user-to-user authentication. As we described already, the specification offers interesting features. Firstly, it offers an IdP discovery functionality by serving the IdP Proxy from `/.well-known` standard location on the IdP. And secondly, the WebRTC identity specification exposes a

4: Firefox Hello was a WebRTC service integrated with Firefox. It was released with the Firefox 35 update, but later discontinued since Firefox 49.

5: `https://developer.chrome.com/apps/app_identity`

6: Chrome Apps are third-party web applications running inside Chrome.

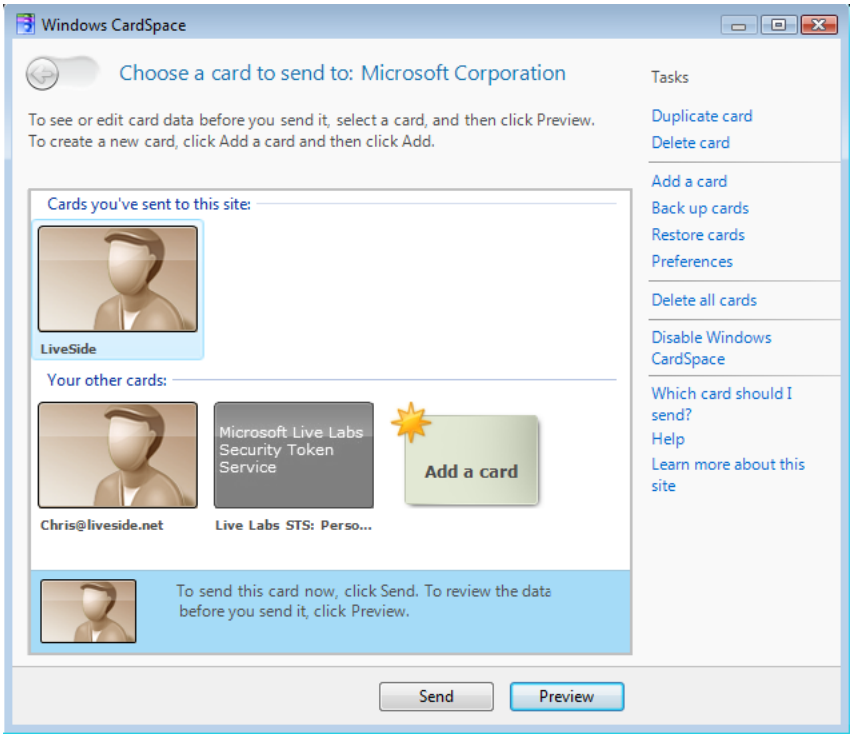


Figure 4.5: CardSpace user interface. When a website requests an authentication, the CardSpace user interface prompts the user with a choice of identity cards. The user interface displays the identity of the requesting website and the user can select one identity card to present to the website. Alternatively, he can preview data contained by his cards, modify them, or add a new one.

simple protocol for authentication based on the generation, exchange, and verification of identity assertions. We believe that an identity-enabled web browser exposing an authentication API to websites could be the basis for a new web Identity Metasystem. The browser would provide the functionalities and associated interfaces to configure new identities, register passwords, display login prompt, and define website preferences, *i. e.* which identity to use with which web site. Some of these functionalities are already provided by web browser, *e. g.* login and password storage, but without the coherency of a full identity management experience.

The IdP Proxy is the core component of the WebRTC identity architecture. It is discoverable and exposes the API for handling identity assertions. Theoretically, it is also protocol independent and we proposed multiple implementations of this component in Section 3.1. In this Section, we address RQ3 "can we let users chose actors they trust to participate in the communication setup?". More precisely, we are interested in answering the following research question:

- **RQ3.4**: Can we leverage the WebRTC identity architecture to let users chose their IdP for user-to-server authentication?

4.2.1 Implementation

To answer our research question, we implement a prototype of a browser modification[7]. Our proposed architecture (see Figure 4.6) relies on three components interacting together:

- A JavaScript API accessible to websites and a user interface for identity selection. Providing a web API is necessary for WebConnect to be a standard functionality of web browser and facilitate its integration by developers.
- An IdP Server able to provide a suitable identity assertion through a WebRTC compatible IdP Proxy.

7: `https://github.com/Sparika/WebConnect`

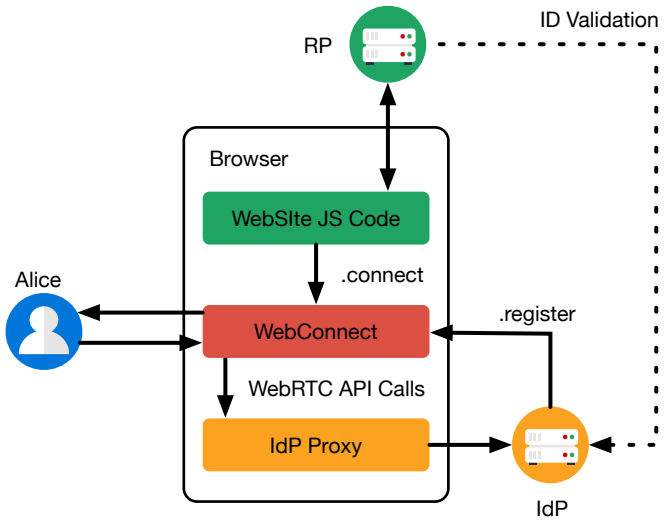


Figure 4.6: WebConnect Architecture

- An RP website –client and server side– able to understand and verify the provided identity assertion

Figure 4.7 shows sequence diagrams representing the interaction between the different actors of our implementation. After the user requested to login, the JavaScript client code calls the `WebConnect.connect` function which returns a promise for an identity assertion. The browser then asks the user to choose one of its registered identity providers and then proceed to instantiate the corresponding IdP proxy. Once the IdP Proxy returns the identity assertion to the browser, the browser resolves the `connect` promise and the website client receives the assertion. It is then up to the client code to transfer the assertion to the server side. In our example, the client calls the GET method on a login URL and passes the assertion as a URL query parameter. The server then extracts the JSON Key URL (JKU) parameter from the identity assertion header, and get the public key from the IdP over HTTPS at the provided location. This public key is then used to verify the assertion signature. Once the assertion authenticity and integrity is confirmed, the server logs in the user and responds to the client GET.

Browser modification

Our browser modification takes the form of a browser extension. This solution was chosen for its simplicity in comparison to browser source modifications. As a browser extension does not expose function to the global window scope, its exposed functions cannot be called by a website. To simulate access to a web API, we also provide a JS shim exposing our API functions to the website client code. This script can then communicate with the extension code through the `postMessage` API.

The API exposed by the `WebConnect` object offers two functions: `connect` and `register` as specified in Figure 4.8. The register function is called by the IdP to let users store identity cards in the browser. The `iss` and `type` parameters allows to discover the IdP proxy `/.well-known` location, while `sub` is an identifier for the user on the IdP. `name` and `picture` are used for display on the user identity card. The `connect` function is called by websites to request an identity assertion authenticating the user. Its only parameter is a JavaScript Object Notation (JSON) object to convey additional parameters such as constraints, authentication level, or authorization requests. We do not implement scenario using this parameter, although we discuss possible usages in Section 4.2.3.

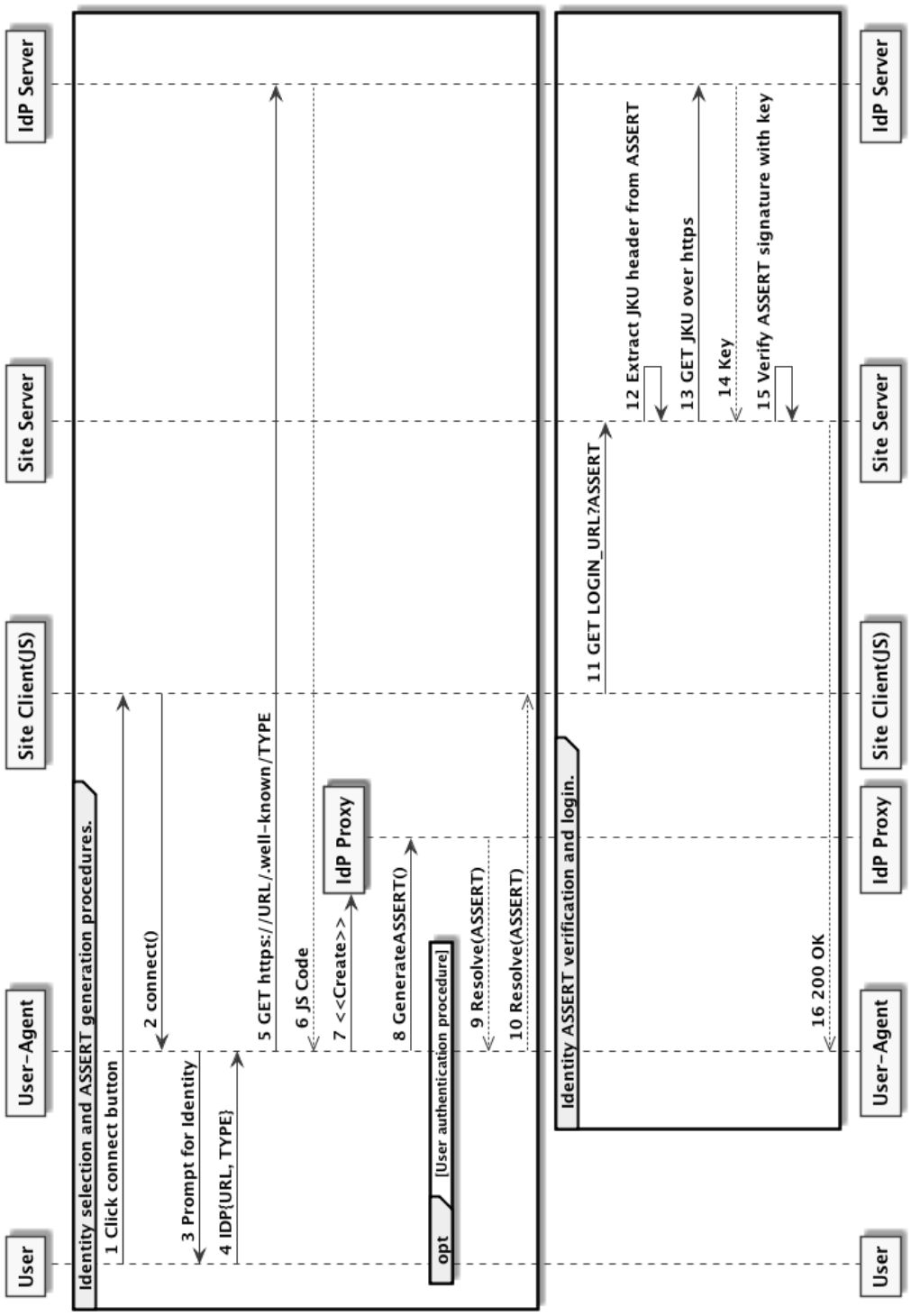


Figure 4.7: WebConnect identity assertion management sequence diagram.

```
interface WebConnect {
    void register (String iss,
           String proxy,
           String sub,
           String name,
           String picture);

    Promise<JWT> connect(Object request);
};
```

Figure 4.8: WebConnect interface specification in WebIDL.

On the technical side, our extension implements a graphical user interface for identity selection and configuration on top of the `PeerConnectionIdP` Firefox module. As we leverage and reuse the WebRTC identity specification, we developed our prototype for Firefox which is the only browser to support it. Initially the extension was using the Firefox Add-on Software Development Kit (SDK) and in particular the `chrome` module. This module allows accessing privileged low-level API. We were thus able to load the `PeerConnectionIdP.jsm` module and directly instantiate and interact with the IdP proxy through it. However, the Firefox Add-on SDK is being deprecated in favor of the WebExtension API[8]. Extensions developed with these API cannot use browser specific low-level API such as the `chrome` module. In order to instantiate the IdP proxy and get the identity assertion, the extension now instantiates a new `RTCPeerConnection`. Though the connection is never initiated, the `RTCPeerConnection` object can be used to call the `setIdentityProvider` and `getIdentityAssertion` functions. While this effectively allows the extension to retrieve an identity assertion, the `RTCPeerConnection.getIdentityAssertion` functions offers less control than the one exposed by the `PeerConnectionIdP.jsm` module.

8: The WebExtension API is compatible with the Browser Extensions API, a cross-browser effort supported by a World Wide Web Consortium (W3C) Community Group for interoperable extensions [161].

Identity provider implementation

The role of the IdP is to provide an IdP Proxy at a standard location, and through it, authenticate the user and return an identity assertion. We reuse our OpenID Connect (OIDC) IdP Proxy implementations described in Chapter 3. The returned assertion is thus a signed JSON Web Token (JWT). In WebRTC the party wanting to verify the assertion validity is supposed to also download the IdP proxy to verify the assertion. However, as we wanted to avoid IdP Proxy sandboxing on the website server, we used the `jku` header in the JWT assertion. This allows the verifying party to retrieves, from the IdP, the public key used to sign the assertion and verify the JWT authenticity.

As standard for OIDC, the assertion payload contains Issuer Identifier (`iss`) and Subject Identifier (`sub`), respectively identifying the IdP and the user to the requesting website. The payload may also include OIDC user-info claims, such as name, address, or email.

Website implementation

Besides calling the API to get an identity assertion, a compatible website must also be able to understand and verify the assertion. In our prototype implementation, the JavaScript code on the client side sends the assertion to its backend server for verification and log in. This is done by a GET to a login URL with the assertion transmitted as a URL query parameter.

The assertion authenticity is then verified by the server. To do so, developer can use several libraries for JWT support. We implement a JWT strategy for Passport[9] –a popular NodeJS authentication library– by adding support for JKU verification.

9: http://passportjs.org/

Once the assertion has been verified, the server extracts relevant information from it and lookup for existing users in its database. If no user exists, the server creates a new entry on the fly. The login procedure thus serves the dual purpose of enrolment and authentication. Ultimately, the user is returned to the relevant page through the HyperText Transfer Protocol (HTTP) response.

4.2.2 Analysis

Security analysis

In comparison to a standard OAuth 2 flow, our implementation introduces two major changes that may have security implications. We discuss these changes in this section.

Firstly, in order to verify the validity of claims covered by a received assertion, the RP must verify the assertion's signature. In OAuth 2 this signature would have been produced by the IdP using a key pair exchanged with the RP during the registration process. In our solution, we replaced the registration process, including the key exchange, by a verification of the `jku`'s origin. However, the OpenID Connect specification states "ID Tokens SHOULD NOT use the JWS [...] jku, or jwk header parameter fields". From Internet Engineering Task Force (IETF) mail archives[10], it appears that assertions claims, including the `iss` and `jku` parameters, are considered to be self asserted until verified by a trusted key. To solve this issue, additional constraint could be added to the key's origin verification. For instance, using a standard `/.well-known` [40] path for the `jku` URL also matching the `iss` domain would prevent attacker from specifying any key. We note that similarly, the WebRTC identity specification specifies that the identity's origin and IdP proxy's origin must match and be served with the HTTPS protocol. Imposing the same constraint for JWT verification should provide a similar security level.

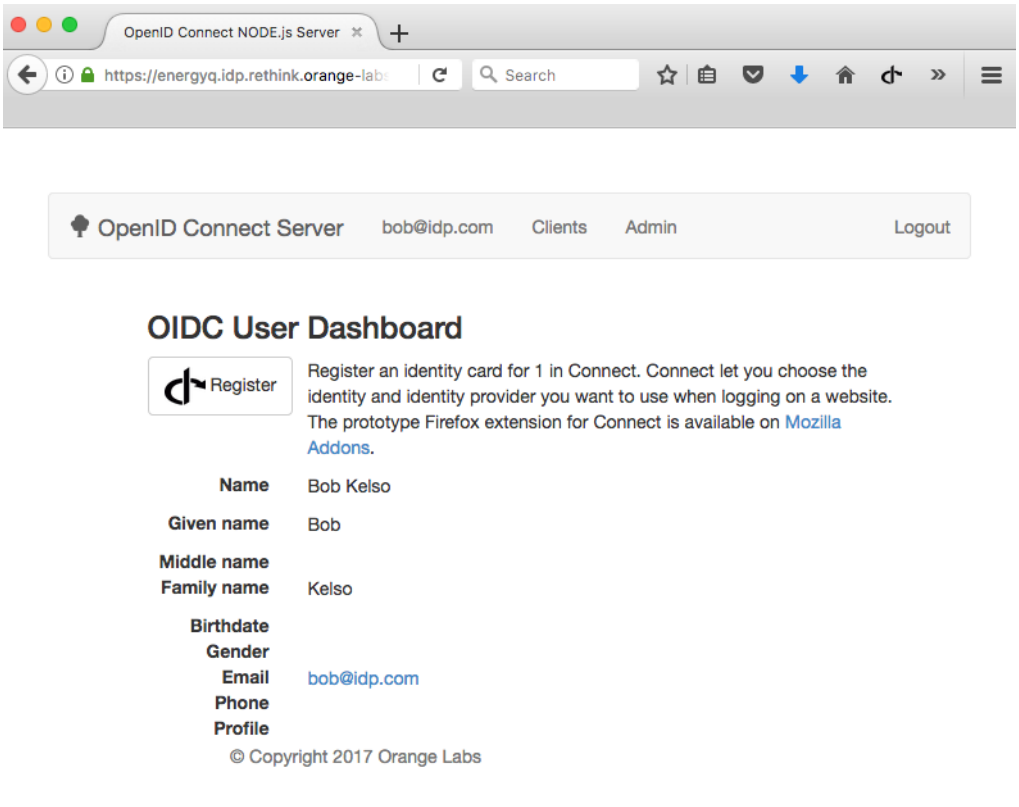
10: `https://www.ietf.org/mail-archive/web/jose/current/msg03929.html`

Secondly, assertions manipulated by the javascript client code and returned by the API's promise response are added to the page's global scope. The assertion could thus be read, and used, by a malicious cross-origin script embedded on the same page. This issue is similar to what can happen on an OAuth 2 implicit flow, where the client directly receives an access token. In OAuth 2, the code flow lets the client exchange a code and authentication with the token endpoint to get the access token, and the ID token in OIDC. But as in our solution, the RP/client cannot be authenticated by the IdP, the code flow cannot be used. An alternative solution could be to use a sort of code exchange, leveraging Transport Layer Security (TLS) mutual authentication between the RP and IdP. Alternatively, the browser could protect the assertion from the JavaScript code and transmit it directly to the RP sever. Action 10 and 1 from Figure 4.7 would be replaced by a single message from the User-Agent to the Site Server. The website redirection Uniform Resource Identifier (URI) would be passed as a parameter to the connect function during action 2 on Figure 4.7.

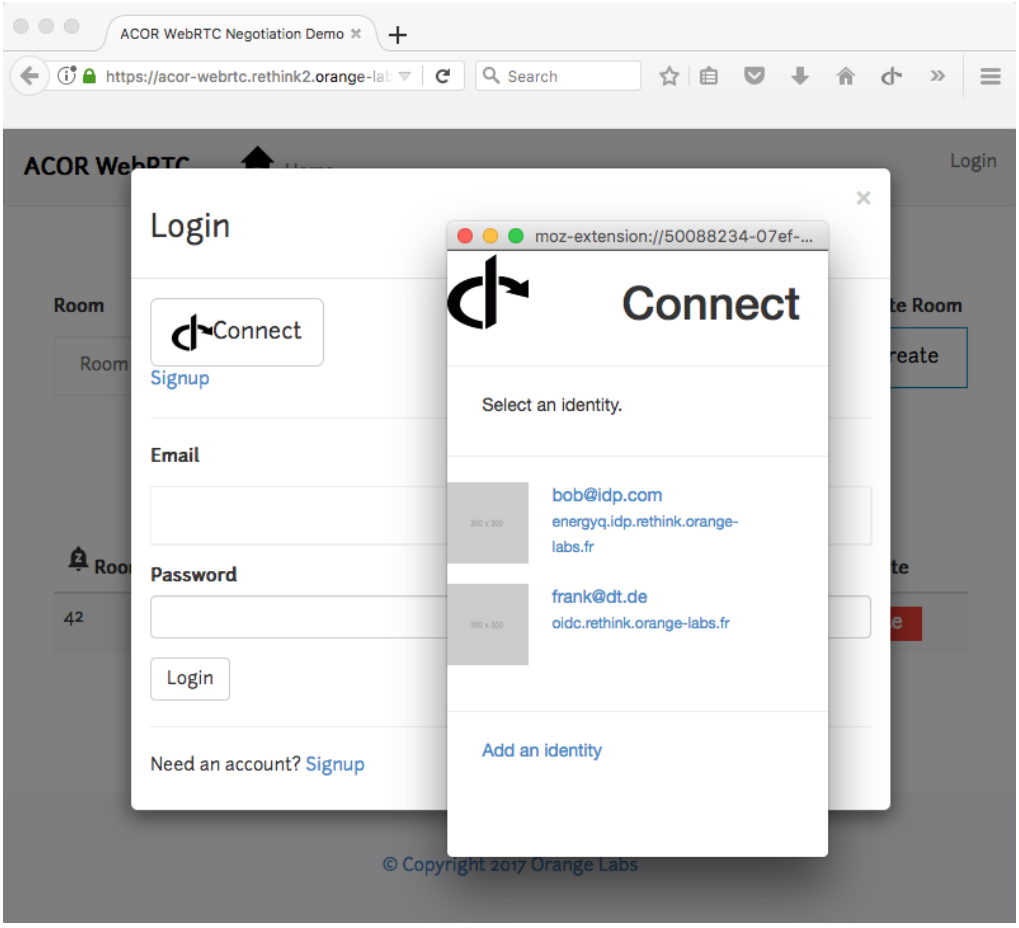
Usability

From the end-user perspective, the overhead is quite limited. Compared to current authentication process, users have to register their identity cards on their browsers. This configuration can be done in a single action with the `register` function of the API. Identity selection on login request is also similar to current SSO solutions, for instance comparing to Google's identity selection interface in Figure 3.8. If that is an issue, preferences storage by the browser may help reduce it further. However, the user experience does not constitute our field of expertise and we did not conduct user studies.

On the developer side of things, we also evaluate the additional work to be limited. Table 4.3 compares the number of new code lines for our prototype implementation



(a) IdP profile page with the WebConnect register button.



(b) Web Connect identity selection interface

Figure 4.9: Prototype user interface for the Authentication web API.

(a): The IdP profile page allows the user to configure a new identity cards to WebConnect. Clicking on register calls the `WebConnect.register` function.

(b): On the visited website, the user can click on connect to authenticate. This opens the WebConnect user interface and allows the user to choose the identity the user wants to authenticate with.

Module	Total code lines	New code lines
Firefox Addon	0	417
IdP Proxy	0	197
Client site (.js)	693	66
Client site (.conf)	457	70
Passport JWS	1242	60
Total	2392	810

Table 4.3: Code lines written for the prototype implementation

compared to the total code lines of each module. In proportion, the biggest tasks are to develop the browser modification and the IdP Proxy, which are new concepts. We note that these developments would be done once by browser makers and IdP developers and not by web developers. For client website developers, the main task is to configure the new authentication method and verify the assertion authenticity. However, as we noted, library for JSON Web Token Signature (JWS) and OIDC ID Token support already exists. Modifying the existing JWS verification library required 60 new lines over a total 1242 code lines, while configuring the new strategy required 70 code lines, mostly copy-pasted from other strategies.

The objective of an Identity Metasystem is to offer a simple API to access multiple identity services. On the long run, the adoption of a web API for authentication delegation should reduce the amount of work for web developers.

4.2.3 Validation

We present a prototype of a user-to-server authentication mechanism reusing already implemented IdP Proxy. Integrated into an identity selector interface provided by the web browser, it effectively allows users to select trusted IdP for authentication on compatible websites. Due to the identity continuity principle (see Section 1.3.4), such freedom of choice would extend to user-to-user authentication in WebRTC context. Through experiment, we answer RQ3.4 and demonstrate that the WebRTC identity architecture can be leveraged to build a user-to-server authentication mechanism. We thus believe that a web Identity Metasystem such as WebConnect is a good way to give users more control over which identity services they want to use both in WebRTC and on the Web in general.

Nonetheless, some control may have to remain in the hands of RP websites. In Section 3.3 we classified at least 58% of observed RP websites as authentication or profile. Websites from this class could request user's authentication without any scope or authorization constraints. We also observed that some RP websites may have a trust relationship with some IdP. Our conclusion stated that "a solution to simplify the discovery and registration of IdP endpoint should nonetheless give RP the option to control the range of compatible IdP and the authentication strength".

The `request` parameter of the `WebConnect.connect()` function allows RP websites to specify constraints on the range of compatible IdP when calling the API. This parameter thus serves as a hook for future extensions allowing to specify:

- trusted IdP's origin,
- authentication strength requests,
- and required scopes for authorization.

These constraints would then be used by the WebConnect interface to only display to the user a range of compatible IdP. Note that the IdP's origin and authentication strength constraints are similar to how we described users negotiating authentication of the other peers in Section 4.1.

4.3 Summary

In some WebRTC scenarios, users may not trust their communication service or the signalling layer. Using the WebRTC identity architecture, users instead rely on IdP to bind the signalling to a peer-to-peer authentication. In such case, users trust in these IdP and their authentication process is necessary for the communication to be trusted. In this chapter, we have looked at solutions to give users more control over WebRTC identity parameters: their peer authentication and their own IdP.

Firstly, we presented ACOR, a SDP extension to negotiate the Authentication Class and the IdP's Origin for the authentication of the other party during a WebRTC call. We implemented our solution in a WebRTC service and tested it using Firefox to answer RQ2.1. Our tests reveal that while it is possible to request identity parameters to the other peer, obtaining feedback on the peer's authentication class is not possible at the moment. We believe that this missing feature may be useful even outside of a negotiation use case and that it could easily be supported by the WebRTC identity architecture.

We then presented WebConnect, a web identity metasystem to let users select their trust IdP. WebConnect answers RQ3.4 and shows that the WebRTC identity architecture can be leveraged to build a user-to-server authentication mechanism. We implemented a prototype version based on a Firefox extension and reusing IdP Proxy implemented in Section 3.1. We believe that a web Identity Metasystem such as WebConnect is a good way to give users more control over which identity services they want to use both in WebRTC and on the Web in general.

Chapter 5

WebRTC Trust and Security Model

In this chapter, we propose a trust and security model of a WebRTC session. Our model integrates into a single metric the security parameters used to establish the session, the media encryption parameters, and user's trust in actors of the WebRTC session. The first objective of our model is to answer RQ1: "What are the risks for the user of a WebRTC session and which abstractions can we use to show these risks to the user"? In particular, we intend our model to help advanced users: i. e. users that would be susceptible to look under the hood, but without expertise in web security. We first present our methodology to build our model in Section 5.1, and then detail our WebRTC trust and security model in Section 5.2. The model evaluates security with regards to the risk presented on the confidentiality and integrity of the communication and shows which trust relations must be valid in order for the security level to be trusted too. In order to validate our approach, we present in Section 5.3 a preliminary study on the understanding of our model by advanced users. This study is based on a survey containing a dynamic implementation of our model which allows modifying the trust and security parameters.

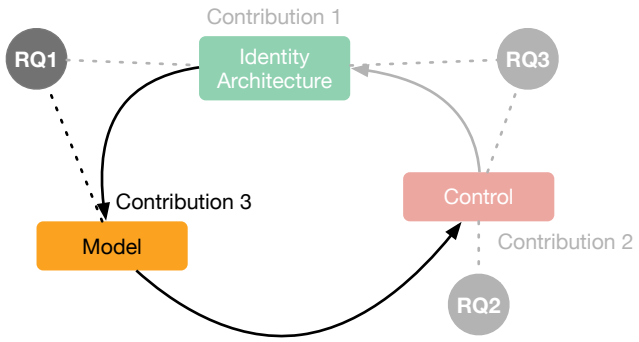


Figure 5.1: Overview of our Contributions: Modelling the WebRTC Trust and Security.

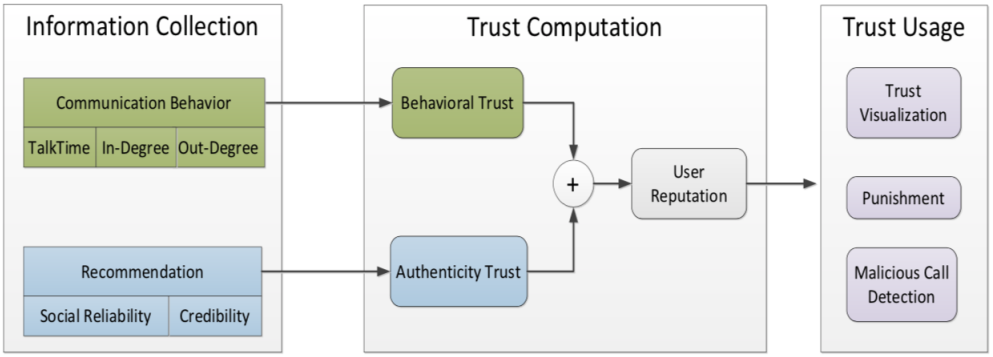


Figure 5.2: Javed *et al.* TrustCall architecture [148].

5.1 Methodology

As we stated in our Introduction chapter:

> *"Our intuition is that users should be given more information and control over the security and trust level of their communications. This is our global objective. For this to be possible, we need to build a model that could represent the communication setup, the different channels, protocols, and the actors in operation. This model would allow us to forge a single metric characterising the risk of using the communication system, i. e. the trust and security level."*

Several methods and models have already been proposed to model the security or trust of a communication system. We reported in our state of the art (see Section 2.2) that Beltran *et al.* [142] propose a trust model of the WebRTC identity architecture in a single Communication Service (CS) scenario. Building on this work, Javed *et al.* [145, 146, 148] work on reputational trust models for WebRTC representing trust, distrust, and mistrust[1]. In their last proposed model [148], the user's reputation is the weighted sum of the user's behavioural trust and a measure of its authenticity as presented in Figure 5.2. They explicitly refer to trust visualisation and help to users' decision as possible uses of their trust model. However, their approach does not completely fit our objective of a security and trust model. Indeed, security considerations in their model are quite limited and only consider the authenticity of the other peer, a remark we also made on their earlier [146] in our state of the art. Furthermore, in this model the authenticity of a peer is actually a reputation score rather than a measure of actual authentication. While it would be acceptable to aggregate authentication score from multiple recommenders, *i. e.* identity providers, the model does not discuss the binding of these identity assertions with the actual WebRTC session. It seems that authenticity is here used with a different meaning as the one we use in this thesis.

We also reported on a model by Alia *et al.* [124] which propose a component-based adaptation model to manage the trade-offs between Quality of Service (QoS) and Security. They model an adaptable Voice over IP (VoIP) system as a composition of components each providing different QoS and security properties. A utility function aggregating QoS and security dimensions, shown in Figure 5.3, allows discriminating between different configurations. Considered dimensions are the latency and video scheme quality for QoS and the confidentiality, anonymity, and authentication for security. Their model also uses user's preferences as weight in the utility function and risk context as minimal required value for each security dimension. While we do not consider QoS in our approach, we pursue a similar long-term objective of managing the security level. However, in our view, this model also shows a limited security model. For instance, the confidentiality function (presented in Table 5.1) only considers security over the media

1: This idea of representing trust, distrust, and uncertainty was first defined by Josang[162] as subjective logic.

$$U = W^{lat}.F(lat) + W^{qua}.F(qua) + W^{conf}.F(conf) + W^{anon}.F(anon) + W^{auth}.F(auth)$$

Figure 5.3: Overall utility function [124] where $F(k)$ are the utility functions and W^k user preference weights for dimensions k as latency, video scheme quality, confidentiality, anonymity, and authentication.

Encryption Algorithm	Key Length	$F(conf)$	Performance Overhead
DES	56	0.2	0.2
AES	128	0.3	0.3
Blowfish	128	0.4	0.4
Blowfish	448	0.5	0.5

Table 5.1: Confidentiality utility function $F(conf)$ and performance overheads from Alia *et al.* [124].

path. Other parameters of the session setup, such as the security of the signalling path or the authenticity of the other peer, would have a major impact on the confidentiality level but are not considered in this function. We also note that the value attributed to the utility functions seems arbitrarily defined. This is, for instance, the case with the confidentiality function shown in Table 5.1 which follows a linear increase.

Another methodology for modelling security of systems is to build attack trees. These models are an analytical approach towards identifying possible attack vectors. Our researches on the state of the art (see Section 2.2) and specific research on "WebRTC attack tree" did not reveal any publication of a WebRTC attack tree. While the STREWS D1.2 document [136] and STREWS D1.1 document [135] refer to an attack tree as a step of their methodology to map threats to WebRTC assets, no such tree has been published in these documents.

As described by Bruce Schneier [163], creating an attack tree is an iterative process starting with the identification of possible attack goals. Then for each goal, possible attacks are added to the tree which will, in turn, require the completion of new attack goals for the attacker. By default, the relation between an attack goal and its sub-goals is defined as an *OR* relation, *i. e.* at least one of the subgoal must be accomplished for the parent goal to succeed. Some other relations can be defined with an *AND* operator. In these cases, each subgoal must be accomplished for the parent goal to succeed. Various values can be further derived from an attack tree by assigned value to leaf or nodes of the tree. For instance, these values may be the cost, difficulty, or conditions required to accomplish an attack. This allows for instance to compute the cheapest possible attacks requiring no special equipment, as in Figure 5.4. This decomposition process is iterated as necessary until no more attacks are found.

To build our trust and security model we follow an iterative decomposition process, in a similar way to attack tree decomposition. However, the intent of the model is to inform the user of the security of its communication setup. It needs to be instantiated with measurable security properties. We thus stop the decomposition of security properties at measurable elements or where policies can be defined.

In Section 4.1 we discussed which actor would be best suited to provide trusted identity recommendation and negotiation capabilities. We concluded that between the Identity Provider (IdP), CS, and Browser, no actors appeared clearly more suited to evaluate and negotiate over the other party authentication. The other peer authentication is only a subset of the security parameters that need to be monitored, but in our opinion, a similar reasoning applies to the overall WebRTC security parameters. We

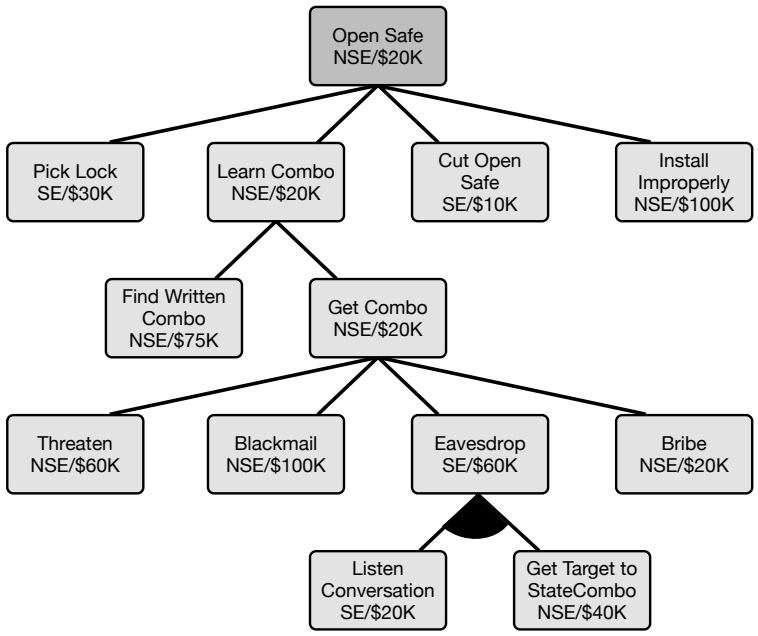
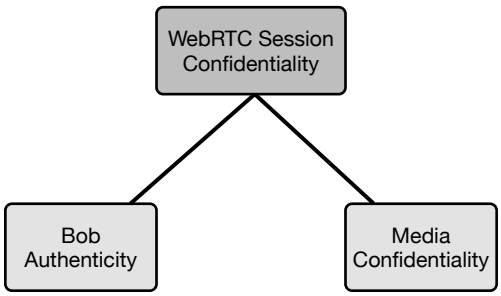


Figure 5.4: Cheapest attack requiring no special equipment against a safe [163]. The black ark denotes an *AND* relation, empty arks denote *OR* relations.

thus suppose that the browser is instantiating the trust and security model and consider only elements that can be observed by the browser.

We previously cited Jøsang and Presti defining trust as "the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible" [15]. To build our model, we start from a high-level trust property node: the trust of the user Alice in the confidentiality of her communication. We then iteratively decompose nodes into their dependencies which are either other high-level trust, security properties or trust relations with other actors. We note trust as $T_X(Y)$ where X is the truster and Y is a trusted property (for high-level trust properties) or a trusted actor (for trust relations). The decomposition process is carried on until a satisfying level of details is reached. Dependencies in attack trees are represented either using an *OR* or *AND* relations as explained previously and we use the same logic in our decomposition. Note that as we model security rather than attacks, for the same scenario the signification of operators is the reciprocal of the equivalent attack tree. For instance, if each attack subgoals must be accomplished as represented by an *AND* operator, our security model would use the *OR* operator meaning that at least one defence dependency must hold. We both model our security and trust tree in graphical and outline form. Graphically we use the same representation as in Figure 5.4 while in outline form we use the postfix decomposition operators $\otimes$ and $\oplus$ respectively for *AND* and *OR*.

As we model the trust and security from an end-user point of view, not all security properties are visible and leafs of this model are thus either security properties or trust relations. For instance, the secure connection between both CS_A and CS_B is not visible by Alice. Thus Alice can monitor her secure connection with CS_A but must trust CS_A for the rest of the signalling path. In order to offer an alternative omniscient model, we continue the decomposition process from terminal trust relations to observe the trust and security model beyond. We define terminal trust relations as being transitive trust relations from trust in an actor to trust in a security context. Graphically, such transitive trust relations are represented as dashed lines while in outline form we use a grey font to represent this omniscient model and $\rightarrow$ as the transitive operator.



(a) Graphical representation.

$$T_A(conf(Session_{A,B,M})) \quad \otimes \quad \begin{array}{l} T_A(auth(B)) \\ T_A(conf(M)) \end{array}$$

(b) Outline form.

Figure 5.5: Alice's trust in the confidentiality of her WebRTC session.

5.2 Building the WebRTC Trust and Security Model

In this section, we model the trust of Alice in the confidentiality of her WebRTC session with Bob. In this scenario, Alice and Bob setup their WebRTC connection each on a different communication service, respectively CS_A and CS_B and both use an Identity Provider, respectively IdP_A and IdP_B[2]. We do not consider attacks conducted by the web application such as interception or redirection attacks. Similarly, we do not consider User-Agent and Operating System trust relationships with users in our model. As the WebRTC specification consider the User-Agent to be the Trusted Computing Base, we adopt the same point of view. Clearly, if the Trusted Computing Base or lower stack software and hardware are corrupted, the whole communication is at risk.

2: The session architecture is similar to the one presented in Figure 1.14a.

5.2.1 Session Confidentiality

In order to trust that the communication setup ensures the confidentiality of the communication, Alice must have trust in the two following properties:

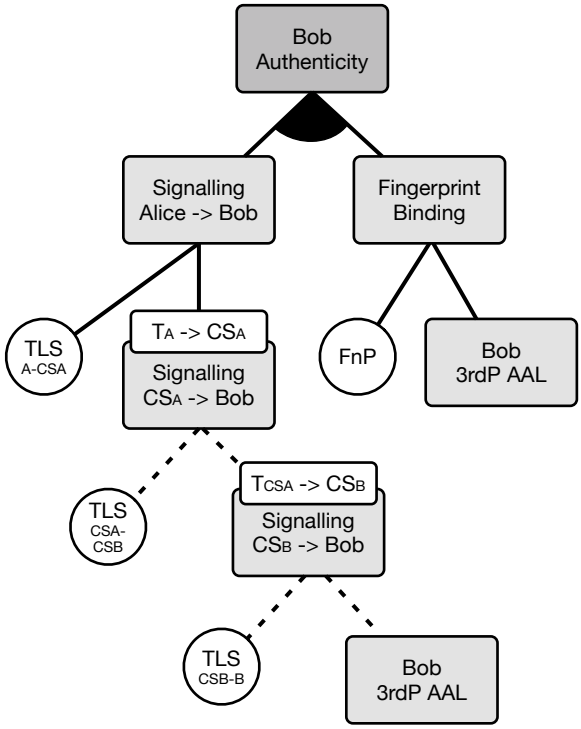
- The other peer's authenticity, *i. e.* that Bob is who he claims to be and that no man-in-the-middle is present on the wire.
- The confidentiality of the peer-to-peer communication between Alice and Bob, *i. e.* that an attacker cannot decrypt the media stream.

Attacks against both these properties are described in the STREWS D1.2 document [136]. As both properties must hold, the trust of Alice in the confidentiality of her WebRTC session is equal to the minimum trust in each of these properties. We represent this relation graphically as in Figure 5.5a and in outline form in Definition 5.5b. Without explicit authentication, the other peer's authenticity depends on actors operating on the signalling path.

5.2.2 Signalling Path Security

Alice relies on the signalling path to establish a peer-to-peer session with Bob. She must trust both communication services, CS_A and CS_B, to correctly route call offers and answers to Bob. The risk faced is that a man-in-the-middle may be set up, either by one of the communication service or by an external attacker leveraging an insecure signalling path. We decompose trust in the signalling path as in Figure 5.6.

To trust the signalling from Alice to Bob, Alice has to trust the security of the first link, *i. e.* the Transport Layer Security (TLS) connection between her and CS_A, as well as the remaining of the signalling path from CS_A to Bob. To this end, Alice has to trust CS_A to behave honestly and securely. The rest of the signalling process is however invisible from Alice's point of view and could not be instantiated. Nonetheless,



(a) Graphical representation.

$T_A(auth(B)) \quad \oplus$
$T_A(sig(A,B)) \quad \otimes$
$TLS(A, CS_A)$
$T_A(CS_A) \rightarrow$
$T_{CS_A}(sig(CS_A, B)) \quad \otimes$
$TLS(CS_A, CS_B)$
$T_{CS_A}(CS_B) \rightarrow$
$T_{CS_B}(sig(CS_B, B)) \quad \otimes$
$TLS(CS_B, B)$
$T_{CS_B}(3pAAL(B))$
$T_A(binding(FnP_B, B)) \quad \otimes$
FnP_B
$T_A(3pAAL(B))$

(b) Outline form.

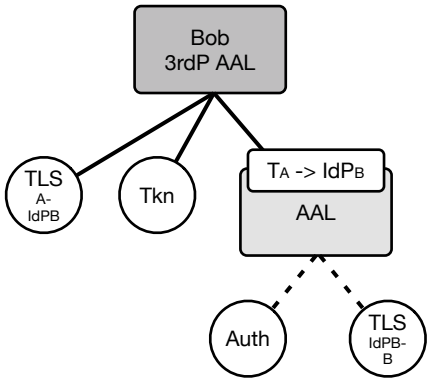
Figure 5.6: Alice's trust in Bob's authenticity resulting from the signalling process.

we represent it in our model for later discussions using dashed lines in graphical form and grey font in outline form. In the simplest scenario, two users setup their WebRTC connection using a single communication service operating in silo. With multiple CS as we consider in this case, the user chooses its own service but may not know through which service the call signalling transits. Communication services may be federated around a unique signalling federation or operate using a circle of trust model. For Alice, its trust relationship with the other party service, CS_B, transits through its own service CS_A. Thus we then consider CS_A trust in the remaining of the signalling path from CS_B to Bob. Similarly to Alice's trust, CS_A has to trust CS_B to behave honestly and that CS_B uses a secure connection to Bob. CS_A does not authenticate Bob and thus has to rely on CS_B trust in Bob's authenticity. In our case, this last property relies on CS_B using an authentication delegation protocol with a third-party IdP.

As explained previously, Bob's fingerprint, noted FnP_B, binds his Datagram Transport Layer Security (DTLS) key used in the media plane to the signalling plane [53]. Optionally, the WebRTC specification also allows binding this fingerprint to a third party identity assertion. If such an identity assertion is provided, this binding has to be broken in order to set up a man-in-the-middle attack. Thus the fingerprint binding depends on both the strength of the fingerprint and on the third party authentication noted $3pAAL(B)$. We detail this part of the model in the following section.

5.2.3 Identity Path Security

As we explain, the WebRTC identity assertion mechanism binds Bob's fingerprint received over the signalling path to an identity assertion issued by Bob's identity provider IdP_B. This assertion is verified by the browser instantiating an IdP Proxy (see Section 1.3.4). The security of this process relies on four dependencies presented in Figure 5.7.



(a) Graphical representation.

$T_A(3pAAL(B)) \otimes$
$TLS(A, IdP_B)$
Tkn_B
$T_A(IdP_B) \rightarrow$
$T_{IdP_B}(AAL(B)) \otimes$
$Auth(B)$
$TLS(B, IdP_B)$

(b) Outline form.

Figure 5.7: Alice trust in Bob's identity path.

Regarding the verification of the identity assertion, Alice must be able to communicate securely with IdP_B. This includes the download of the IdP Proxy over HyperText Transfer Protocol Secured (HTTPS), a policy enforced by the WebRTC specification, but also subsequent communications between the IdP Proxy and the IdP server. The verification of the identity assertion must also be secure against integrity attacks. We note the trust in the identity assertion as $T_A(Tkn_B)$ although the exact nature of the identity assertion depends on the protocol implemented by the IdP. For instance, one of our implementation presented in Section 3.1.2 uses a JSON Web Token (JWT) containing the user's fingerprint. Compromising the integrity of the JWT could allow an attacker to impersonate one of the peers and mount a Man-in-the-Middle (MitM) attack.

Alice also needs to trust Bob's IdP, noted $T_A(IdP_B)$, and its authentication of Bob. However, this process is not visible from Alice. Nonetheless, we represent it in dashed lines in graphical form and grey font in outline form. In some cases, the IdP may recommend Bob's authentication strength, for instance using the OpenID Connect (OIDC) Authentication Context Class Reference (ACR) claim. We note this recommendation $T_{IdP_B}(AAL(B))$ which depends on the authentication strength of Bob and the presence of a secure connection between Bob and his IdP.

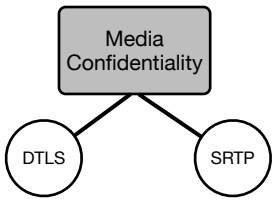
Supposing that CS_B authenticated Bob using an authentication delegation protocol, the same decomposition could be applied to the trust of CS_B in Bob's authenticity noted $T_{CS_B}(3pAAL(B))$ and presented in Figure 5.6. As we remarked in Section 1.3.4, in most cases the same IdP would be used by Bob to authenticate to CS_B and Alice.

5.2.4 Media Path Confidentiality

The confidentiality of the media path is ensured by the Secure Real-time Transport Protocol (SRTP) protocol using DTLS for handshake and keying and vulnerabilities or weak security level on either of these protocols could lead to a compromised encryption of the media path. The WebRTC security architecture [47] mandates the implementation of DTLS 1.0 and recommends the implementation of DTLS 1.2. Guidelines regarding the implementation of cypher suites and SRTP profiles are also given. The trust of Alice in the media path confidentiality thus depends on the strength of the protocols and the cypher suite in use as presented in Figure 5.8.

5.2.5 Overall Trust and Security Tree, Instantiation and Computational Models

Combining the previous subtree, we reconstruct the whole trust and security tree as presented in Figure 5.9. Leafs of the decomposition tree are either security elements



(a) Graphical representation.

$$T_A(conf(M_{\{A,B\}})) \quad \otimes$$
$$DTLS$$
$$SRTP$$

(b) Outline form.

Figure 5.8: Alice's trust in the confidentiality of the peer-to-peer media streams.

e. g. Tkn_B, or trust in actors of the communication setup *e. g.* $T_A(IdP_B)$. In order to instantiate the model, it is necessary to assign values to these leaf elements and a computational model to $\otimes$ and $\oplus$ nodes. The model can then be evaluated to return an overall value representing the security of the WebRTC session to the user.

In Section 1.4, when introducing the concept of trust, we explained that trust models are generally categorised between policy and reputation-based trust. Supposing the definition of some policy rules, trust in actors could be represented as a boolean. Users could configure such policies in their browsers for instance as a list of trusted actors. A similar functionality is already implemented by browsers for the management of web Application Programming Interface (API) authorization granted to web origins. Alternatively, trust model can rely on reputation trust, often represented as a continuous score, for instance on $[0; 1]$. This is the approach followed by the many reputation models such as from Javed *et al.* [145] which suppose either a centralised recommendation source or a distributed reputation model based on users' opinions. However, managing too many permissions may be difficult for users and their permanent nature may result in vulnerabilities as reported by the STREWS project [136]. The web browser or a third-party could also provide such recommendation lists, for instance in a similar fashion to add-blocking extensions[3].

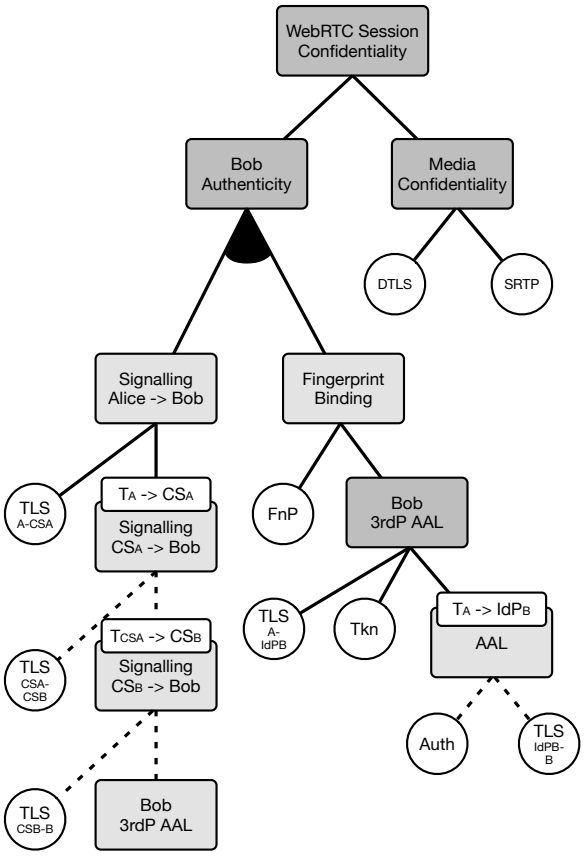
3: Although this model also has its limits as add blockers gain a powerful and somewhat illegitimate position.

Regarding values for security elements, we reported in the state of the art (Section 2.1) how national security agencies publish recommendations on security algorithms implementation and usages based on prior work by Lenstra [121]. As with trust, policies can define good enough security level, for instance, based on implemented cypher suites and Authentication Assurance Levels (AAL). While compliance with such policies can be represented as boolean values an alternative is to use numerical values. For instance, Alia *et al.* [124] use such values for their security utility functions, although apparently arbitrarily defined. In order to derive numerical values for a security utility function, a solution is to divide the implemented key length by the recommended key length bounded to one:

$$F(k, r) = \begin{cases} 2^{k-r} & \text{if } k \leq r \\ 1 & \text{else.} \end{cases}$$

Figure 5.10: Comparative security utility function with k the actual key size and r the recommended key size both expressed in bits.

Table 5.2 gives an example instantiation of such function over a WebRTC setup. In this example, Alice uses the Chrome web browser to visit a WebRTC website and connect to Bob. The website CS_A provides a "Let's encrypt certificate" and the connection is encrypted using the AES symmetric encryption algorithm. The ANSSI recommends a 100 bits key size for symmetric encryption algorithms [122] and in this instance, a 256 bits key is used. DTLS and SRTP are then used to establish the WebRTC session. More particularly, only the DTLS ECDHE key exchange and the ECDSA authentication mechanism are used with the P-256 curve, meeting the recommended key size for elliptic curves. The certificate's fingerprint is a SHA256 hash which also meets the recommen-



(a) Graphical representation.

$T_A(conf(Session_{A,B,M}))$ $\otimes$
- $T_A(auth(B))$ $\oplus$
  - $T_A(sig(A,B))$ $\otimes$
    - $TLS(A, CS_A)$
    - $T_A(sig(CS_A, B))$ $\otimes$
      - $T_A(CS_A)$
      - $TLS(CS_A, CS_B)$
      - $T_{CS_A}(sig(CS_B, B))$ $\otimes$
        - $T_{CS_A}(CS_B)$
        - $TLS(CS_B, B)$
        - $T_{CS_B}(3pAAL(B))$
  - $T_A(binding(FnP_B, B))$ $\otimes$
    - FnP_B
    - $T_A(3pAAL(B))$ $\otimes$
      - $T_A(IdP_B)$
      - $TLS(A, IdP_B)$
      - $T_A(Tkn_B)$
      - $T_{IdP_B}(AAL(B))$ $\otimes$
        - $auth(B)$
        - $TLS(B, IdP_B)$
- $T_A(conf(M_{\{A,B\}}))$ $\otimes$
  - $DTLS$
  - $SRTP$

(b) Outline form.

Figure 5.9: Overall trust of Alice in the confidentiality of her WebRTC session.

dations. Once the connection is established Alice asks Bob to authenticate using the solution described in Section 4.1. The TLS session between Alice and IdP_B uses the same configuration as for CS_A. Finally, the identity assertion is a JWT signed with the RS256, *i. e.* RSASSA-PKCS1-v1_5 using SHA256 which also meets the recommendations.

The operator $\otimes$ represents weakest link dependencies, *i. e.* all trust or security dependencies must hold for the parent property to hold. The value of an $\otimes$ node is thus the minimum value of each of its dependencies. Symmetrically, the operator $\oplus$ represents situations where at least one dependency must hold for the parent property to hold. We thus define the value of an $\oplus$ node as the maximum value of each of its dependencies. These two operators allow us to represent the trust and security view from the user's point of view, *i. e.* limited to what the browser can monitor. However, we also extended our trust and security tree to relations between other actors, *e. g.* Bob's authentication by his IdP in Figure 5.7. These extensions are characterised by transitive trust relations, *e. g.* Alice's trust in Bob's IdP in Figure 5.7. To evaluate such transitive trust, we use the trust relationship as a weight to its dependency value. Figure 5.11 presents the overall formula for evaluating the trust of Alice from both her point of view and from an omniscient point of view.

Table 5.2: Security element instantiation based on ANSSI recommendations [122].

Security Element	Key Length	Category (rec)	Value
$TLS \quad A-CS_A/A-IdP_B$ TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	256	Sym (100)	1
$DTLS$ TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	P-256	EC (200)	1
$SRTP$ AES_CM_128_HMAC_SHA1_32	128	Sym (100)	1
FnP_B SHA256	256	Hash (200)	1
Tkn_B RSASSA-PKCS1-v1_5 using SHA256	256	Hash (200)	1

5.3 Validation

Our WebRTC trust and security model is designed to help end-users understand the security of their communications and how their trust in actors of their WebRTC setup may influence their security. Similarly to the secure connection indications implemented by web browser, *i. e.* the HTTPS green lock, our model can be instantiated and evaluated to return a single value. It could also be presented in an instantiated graphical form to provide a detailed view of the situation intended for advanced users, again similarly as to how web browser display details on HTTPS certificates. While the usage of a coloured icon as a way to provide a security indicator is already deployed in browser, this is not the case of our trust and security decomposition model. In this section, we intend to answer the following questions:

- **RQ1.2** How do users understand the definition of trust in actors of the communication setup?
- **RQ1.3** Does our trust and security model helps users understand the security of their WebRTC communication?

5.3.1 WebRTC Trust and Security Model Survey

To this end, we conducted an online survey that we distributed to researchers in our team. Compared to a more generic end-user population, we believe that this population is more concerned with the security of their communications and as such more susceptible to search for a detailed view of a communication security setup. However, compared to security experts, communication security is not an area of focus for our research team. We thus estimate that this population corresponds to the category of end-users that would like to understand what's happening "behind the hood" but may need the help of an high-level model to grasp the situation. As the size of the surveyed population is however quite limited, we do not claim any representativity of our results but we see them as preliminary. More investigations may be needed.

One of the design goals of our survey is that we want participants to express their own intuitive understanding of the trust and security setup helped by our model. The difficulty lies in how we explain our model and WebRTC security, without influencing too much the participants. For instance, giving a crash course on WebRTC basically saying "WebRTC is secure if a third-party IdP is used" would help participants understand the WebRTC security architecture, but it would not reveal if this understanding was due to our model or not. Throughout the survey, we thus walk a narrow line where we provide a limited description of the VoIP or WebRTC scenarios and architectures.

The survey is organised in four successive page[4]. On the first page, we recall how web browser asserts the confidentiality and authenticity of HTTPS connection to website by

4: The survey is accessible at `http://kcorre.github.io/webrtcsurvey`.

$$
\begin{aligned}
&T_A(Session_{A,B,M}) = \\
&MIN(\\
&\quad MAX(\\
&\quad\quad MIN(\\
&\quad\quad\quad TLS(A, CS_A), \\
&\quad\quad\quad T_A(CS_A) \\
&\quad\quad), \\
&\quad\quad MIN(\\
&\quad\quad\quad FnP_B, \\
&\quad\quad\quad TLS(A, IdP_B), \\
&\quad\quad\quad Tkn_B, \\
&\quad\quad\quad T_A(IdP_B) \\
&\quad)), \\
&\quad DTLS, \\
&\quad SRTP \\
&)
\end{aligned}
$$

(a) Alice's point of view.

$$
\begin{aligned}
&T_A(Session_{A,B,M}) = \\
&MIN(\\
&\quad MAX(\\
&\quad\quad MIN(\\
&\quad\quad\quad TLS(A, CS_A), \\
&\quad\quad\quad T_A(CS_A) * MIN(\\
&\quad\quad\quad\quad TLS(CS_A, CS_B), \\
&\quad\quad\quad\quad T_{CS_A}(CS_B) * MIN(\\
&\quad\quad\quad\quad\quad TLS(CS_B, B), \\
&\quad\quad\quad\quad\quad T_{CS_B}(IdP_B) * MIN(\\
&\quad\quad\quad\quad\quad\quad Auth(B), \\
&\quad\quad\quad\quad\quad\quad TLS(B, IdP_B)))) \\
&\quad\quad), \\
&\quad\quad MIN(\\
&\quad\quad\quad FnP_B, \\
&\quad\quad\quad TLS(A, IdP_B), \\
&\quad\quad\quad Tkn_B, \\
&\quad\quad\quad T_A(IdP_B) * MIN(\\
&\quad\quad\quad\quad AAL(B), \\
&\quad\quad\quad\quad TLS(B, IdP_B)) \\
&\quad)), \\
&\quad DTLS, \\
&\quad SRTP \\
&)
\end{aligned}
$$

(b) Omniscient point of view.

Figure 5.11: Overall computational formula for Alice trust in her WebRTC session.

displaying a green lock icon. We then briefly explain our objective and that the survey will be used to evaluate the interest, usefulness, and clarity of our security model. The intended duration of the survey, ten minutes, is also stated. The last page is used to let participants qualify their expertise in the fields of web technologies, computer security, and real-time communication technologies. For each field of expertise, participants are instructed to choose an expertise level between end-user, intermediate, and expert. The bulk of the survey's questions are on page two and three which we detail below.

In the second page, the survey focuses on trust in audio and video communications. We first present the role of CS in VoIP services and their responsibility of the signalling and authentication of participants. After defining WebRTC illustrated with a basic WebRTC architecture, we explain how identity providers allow users to authenticate by exchanging identity assertions. We then define several real-time communication scenarios and request participants to evaluate their trust in such scenarios regarding the confidentiality of their communications. We propose a trust scale between 0 and 10 in the context of communication confidentiality and privacy, defined as follows:

> *"On this scale, 10 represents an absolute trust that actors in the communications setup or external attackers are not breaching, or are not able to breach,*

the confidentiality and privacy of your communication. While 0 stands for a total distrust. i.e. an attack could be mounted as with a weak security level."

Formulated as such, trust is a subjective measure. Our intent here is both to exemplify various real-time communication deployments and to see if users feel comfortable in attributing trust score to VoIP scenarios and their underlying communication services. The scenarios described in our survey are:

- A national mobile phone call.
- An international mobile phone call.
- A well-known web communication service, *e. g.* Skype, Messenger, or Whatsapp.
- A web meeting service provided by your company.
- A web service for hosting work meeting and discussions, *e. g.* Slack, Fleep, Appear.in.
- A WebRTC enabled webpage providing a call widget to a customer service; *e. g.* a banking website.
- An untrusted WebRTC communication service with a third-party identity assertion from a known identity.
- A Real-Time Communication service using an old plugin and unspecified security parameters, *e. g.* with a Flash plugin.

Basically, these scenarios describe three communication service categories: legacy phone services, web Over The Top (OTT) services, and ubiquitous WebRTC services. Comparisons can be drawn between scores attributed to scenarios in the same categories and between categories. In scenarios describing mobile phone services, we insist on the difference between national and international call. As we explained previously, the telco model relies on trust circles to integrate multiples operators. While telco operators are not explicitly visible when receiving a call, national operators are at least known to the users which may less be the case for international operators. In scenarios describing web OTT services, we insist on three categories of services: big OTT player often feared and criticised for their dominant position, smaller communication services which may benefit from a better image, and services officially recommended or provided by the user's company. These scenarios are thus intended to reveal trust based on subjective opinion rather than technical setup. In the WebRTC service category, we intend to see how users may intuitively trust the WebRTC identity architecture. The WebRTC enabled webpage scenario is exemplified with a banking website which should suggest high-security standards and thus trust. On the contrary, in the second WebRTC scenario, we specifically describe an untrusted WebRTC CS, although backed by a trusted IdP. In the WebRTC security architecture [47], such scenario is used to justify the use of the third party IdP. Finally, the last scenario suggests a service using a weak security level or with existing vulnerabilities.

Submitted results are presented in Table 5.3. As the surveyed population is not representative we do not draw any conclusions but bring some remarks to the reader's attention. We expected attributing trust values to be a clearly subjective task. However, as we observe that trust values in a given scenario are quite different for each participant, we also observe that participants use different trust scales. This may reveal that a single trust value may not have the same meaning for two users. We also note that each participant who rated the insecure scenario with a trust value superior to 0 also qualified their experience with computer security as "end-users". Another interesting observation

Figure 5.12: Survey: Trust in Communication Scenarios.

Scenario	u1	u2	u3	u4	u5	u6	u7	u8	u9	u10	Mean
Mobile	3	6	9	9	3	6	6	7	7	0	5.6
Int. Mobile	2	4	7	4	3	5	5	4	6	0	4
Big OTT	3	6	6	6	3	6	6	6	7	3	5.2
Small OTT	3	6	4	6	4	8	8	5	4	3	5.1
Company	4	6	6	7	3	9	8	8	8	0	5.9
WebRTC	3	3	3	5	5	8	8	5	2	3	4.5
WebRTC IdP	2	1	5	6	3	6	0	5	1	3	3.2
Insecure	1	2	0	3	3	0	0	0	0	0	0.9
Trust scale	1-4	1-6	0-9	3-9	3-5	0-8	0-8	0-8	0-8	0-3	-

Table 5.3: Survey results: trust in audio and video communications.

is the mean trust of the WebRTC IdP scenario. With the exception of the insecure scenario, the WebRTC IdP scenario has the lowest mean trust. Further investigation may thus show that providing an identity assertion may not be enough for users to trust any WebRTC enabled website.

In the third page of the survey, we let participants play with a dynamic implementation of our trust and security model in graphical form as presented in Figure 5.13. The model represents high-level trust property as coloured nodes depending on their actual trust values, and security properties as small black nodes. A panel in the top-left corner let participants define various trust relations. Similarly, a list of radio button elements corresponding to communication scenarios on the previous page let participants select trust configurations. In this mode, the actual trust values depend on the trust values previously defined by the participant. For instance, if one participant set a trust value of 6 to the WebRTC IdP scenario, selecting this scenario on the dynamic trust model would set the following trust relations: $T_A(CSA) = 0$ and $T_A(IdP_B) = 0.6$. Participants can also interact with black security nodes by double-clicking on them in order to modify the nodes' value. In order to explain the model, a limited description of the modelled WebRTC scenario is provided as reproduced in Figure 5.14a. The dynamic model also displays tip note when hovering over high-level and security nodes, mainly used to explain acronyms.

Finally, participants are invited to evaluate their interest for the presented WebRTC security model and the trust layer, as in Figure 5.14b. The proposed values for both questions are *Interesting*, *Slightly interesting*, and *Not interesting*. Participants are also instructed to state up to which decomposition level are they able to understand the model. The proposed values are *I don't understand the model*, *Top value*, *One or two level*, and *The whole tree*. The survey also offers a text box for comments and we gathered other feedbacks in face to face discussions. Results of this part of the survey are presented in Table 5.4.

Again, our results are not representative and we limit ourselves to observations. Although the survey is anonymous, rating the interest of the model in such manner probably gives biased results probably. It is thus more interesting to look at the evaluation of the interest in conjunction with the level of decomposition the participants were able to understand the model. Three answerers *-u6, u7, u9-* declared that they understood the whole model and that it is an interesting representation of WebRTC security and trust relations. None of these declared themselves an expert in any categories. Other participants *-u1, u2, u5, u10-* found an interest in the model but did not understand the whole decomposition declared. Finally, some participants *-u3, u8-* only

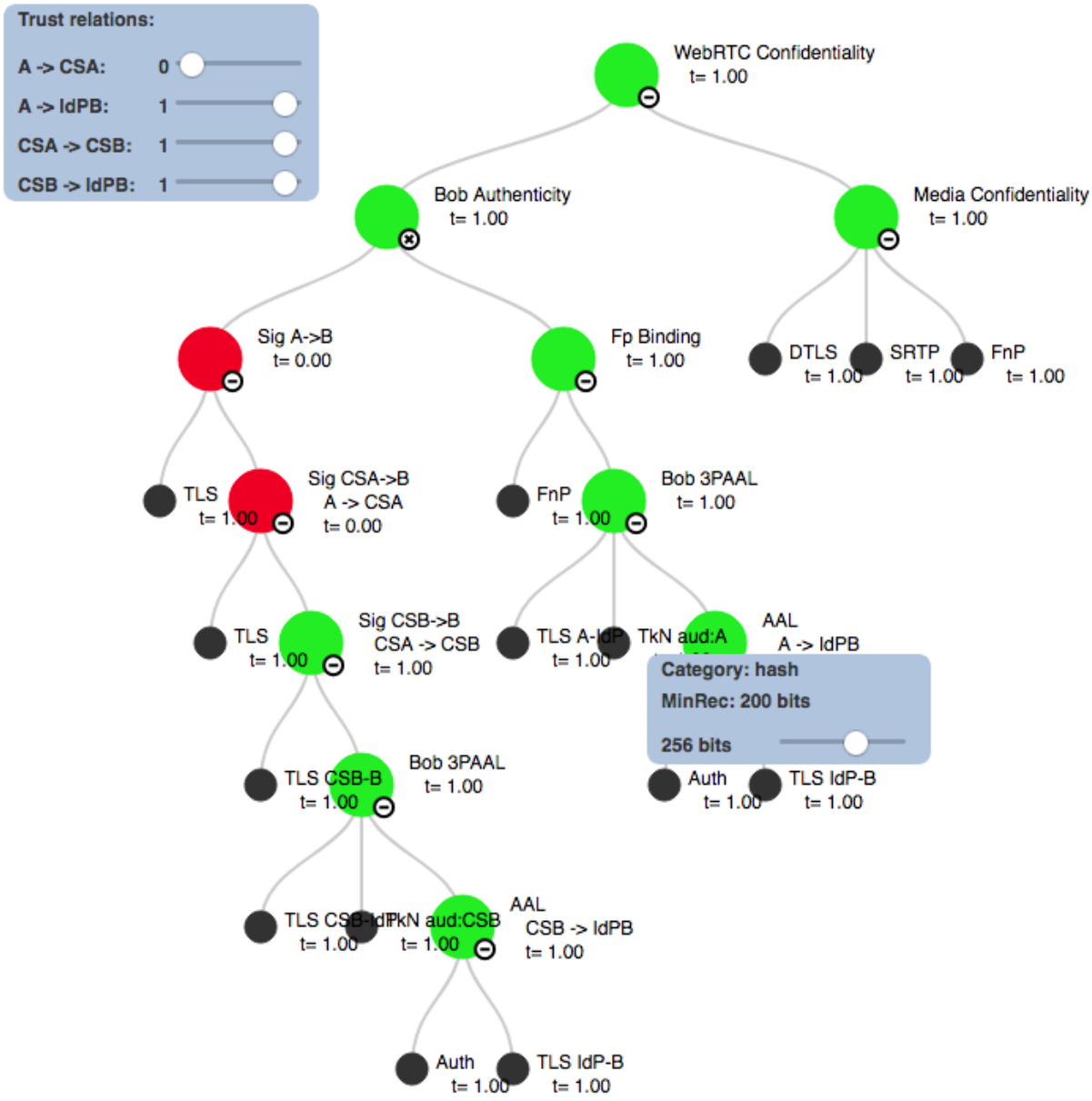


Figure 5.13: WebRTC Trust and Security Model implementation in D3.js.

(a) Description of the WebRTC scenario of the dynamic model.

(b) Survey Questions

Figure 5.14: Survey: Interest in the WebRTC Trust and Security Model.

Interest	u1	u2	u3	u4	u5	u6	u7	u8	u9	u10
WebRTC model	2	2	1	1	2	2	2	1	2	2
Trust layer	2	2	1	1	1	2	2	1	2	2
Detail level	Some	Some	Top	Some	Some	All	All	Top	All	Some

Table 5.4: Survey results: Interest of the trust and security model. Interest for the WebRTC model and trust layer correspond to *Interesting* (2), *Slightly interesting* (1), and *Not interesting*(0). On the detail level line, *Top* stands for only the top value, *Some* stands for only one or two decomposition level, and *All* stands for the whole model

understood the top value and only found a slight interest in the model *-u3, u4, u8-*. The comments and discussions also indicated that the meaning of lower decomposition levels where difficult to understand and not adapted for end-users. Even with the help of tip notes to explain nodes. It is also interesting to see that participants who declared to understand the whole tree *-u6, u7, u9-* did not rate themselves as web technologies experts on the contrary to some participants who did not understand the whole model *-u1, u2, u5, u10-*. As a participant reported "the model [...] is interesting for people who do not have much knowledge in the field but still are interested in knowing how it roughly works." On the other hand, it may not be satisfying for people with more experience in web technologies who would have higher expectations before declaring themselves able to understand the model.

5.3.2 Discussions

As we explain, the validation of our WebRTC trust and security model is limited to a preliminary experiment. In the future, we intend to conduct further investigation on a larger population. Evaluating trust in a survey's artificial setting is a complex and difficult task. Nevertheless, the preliminary study helped in identifying avenues for refinement.

In the survey, we explicitly instructed participants to consider trust in the context of the confidentiality and privacy of their communications. However, some participants reported a difficulty to contextualise this trust without visible communication scenario. We envision that rather than explaining the WebRTC architecture, our large-scale survey should include an instance of a WebRTC communication service. This communication service would give a sense of context for the model and its trust and security parameters to the participants. Similarly, the expression of context in the model is limited to some high-level trust nodes. In Section 1.4 we explained that a trust relation depends on a specific context and that recommendations may allow transitivity from one trust context to others. A more explicit representation of trust context and context transitivity may help in understanding the model. We thus intend to explore how a coloured trust tree could be used to explicitly represent trust context in the model. We also observe a variation in the numerical trust scales used by users. While this variation may be due to different perception of trust for a single scenario, it may also reveal that users do not agree on the meaning of numerical trust values. Although our model uses a numerical representation of trust, it may be preferable to let users define trust using natural language. This questions the practicality of trust value provided by reputation systems; as a single trust value may not have the same meaning across the whole user population. This may be worth exploring in future research.

In the survey, and in particular on dynamic model, we represented the whole decom-

position tree including transitive relations. From the interest and understanding scores, we observe that different level of details caters to the expectations of three categories of users. In the perspective of a larger scale study, a functionality to let users expand the decomposition tree will be implemented. This would let participants select their preferred level of details and focus on what they are able to understand. The dynamic model also reveals that in practice, the confidentiality utility function presented in Table 5.2 is equivalent to a boolean policy. Indeed, slight change in key size results in large decrease in the security value. Moreover, cypher suites key size implementations generally vary by more than a few bits. Instantiating the model's security properties using WebRTC security statistics seems a more practical approach. Building the survey and the dynamic model around an actual WebRTC communication service will help the implementation of this approach. However, this requires access to those security statistics provided by browser. At the time of writing, Firefox does not publish security statistics for WebRTC session[5]. On Chrome, these statistics are accessible from the browser WebRTC statistics[6] but not from JavaScript code as in Figure 5.15.

5: `about:webrtc`

6: `chrome://webrtc-internals/`

Figure 5.15: This JavaScript code prints `RTCTransportStats`. However, on Firefox it returns no element, and on Chrome the returned stat does not include `dtlsCipher` or `srtpCipher` elements.

```
var pc = new RTCPeerConnection()
    [...]
    pc.getStats()
    .then(stats => {
        stats.forEach(stat => {
            if(stat.type == 'transport')
            console.log(stat)
        })
    })
```

Finally, we remark that Alice's IdP does not appear in the trust and security model, neither from Alice's point of view or from an omniscient point of view. On an untrusted CS, Alice's IdP may be the only actor in the communication setup trusted by Alice. This is paradoxical but easy to explain: as Alice authenticates with her IdP, for all she knows she may be authenticating to a MitM attacker. It is Bob's IdP that proves to Alice that the communication is secure. For this reason, it is important that Alice trusts Bob's IdP, for instance through a negotiation process as we presented in Section 4.1. An alternative solution could be to use a confirmation mechanism so that Alice's IdP may be enough for Alice to trust the session. Such mechanism could draw from ZRTP (see Section 1.3.6) or simply be a modification of the existing WebRTC identity validation protocol.

Summary

In our previous contributions, we have implemented the WebRTC identity architecture and proposed solutions for users to have more control over the WebRTC identity parameters. We believe that for users to understand, and trust, the security of their communications, they should have more information about the security of their communications. Web browsers are currently pushing for a secure web and are educating users to look for website security configurations. However, the security of WebRTC is more difficult to understand as it involves more actors than a simple client-server connection. In this chapter, we have proposed a model representing the security of a WebRTC session.

We presented our methodology based on an iterative decomposition process. Our model uses the security parameters of the signalling process, media encryption, but also trust parameters configured by the user. It is a representation of the user's trust in the confidentiality of its session, from its own point-of-view. However, we also consider the transitive nature of trust relations and propose an omniscient view of the session confidentiality model. We then discuss an instantiation of our model and propose a utility function for security parameters.

To validate our model, we conducted a preliminary experiment on non-experts users. In this study, we evaluated how users understand the definition of trust in actors of the communication setup and how our trust model helps them understand the security of their WebRTC communication. This study was based on an online survey offering participants to interact with a dynamic implementation our model. This survey helped identify potential difficulties for the understanding of the model and the measure of users perception.

Part III

Conclusion and Perspectives

Chapter 6

Conclusion

In this thesis, we have studied how users can control the trust and security level of their Web Real-Time Communications. WebRTC is a standardisation effort for interoperable real-time communication in the Web, in line with the specifications of HTML5 technologies. These specifications aim to provide for dynamic webpages, running in a compatible browser, and suitably authorized by the user, with the capability to set up audio, video, or data communication. We first conducted a research survey on Voice over IP (VoIP) and WebRTC security research and observed that the WebRTC identity architecture attracted a lot of interest from the community. This architecture decouples the signalling path from the identity path by binding media session security certificates to an identity asserted by an Identity Provider (IdP). The claim of the specification is that this architecture allows users to trust the security of their sessions even if the Communication Service (CS) is not trusted. However, we noted that the security and privacy of the specification have only been studied from a theoretical point of view [139]. In particular, the cross-implementation issues between Single Sign-On (SSO) protocol and WebRTC are rarely considered [141, 139]. The specification itself only sketches an implementation with the OAuth protocol in its annexe [47]. We neither observed research considering the IdP as a possible attacker of the user's privacy.

Paradoxically, the WebRTC identity path is left for the CS to configure and control as it is the CS who sets the IdP to use. From our point of view, it is not clear if the identity path can be trusted independently of the CS. Our intuition is that users should have more information and control over the security and trust level of their communications. With this objective in mind, we studied the following research questions:

- **RQ1**: What are the risks for the user of a WebRTC session and which abstractions can we use to show these risks to the user?
  - **RQ1.1**: Are there any security vulnerabilities in the identity path of the WebRTC security architecture?
  - **RQ1.2**: How do users understand the definition of trust in actors of the communication setup?
  - **RQ1.3**: Does our trust and security model helps users understand the security of their WebRTC communication?
- **RQ2**: Can we act on a WebRTC session to raise the trust and security level?
  - **RQ2.1**: How to let users negotiate the other peer's identity parameters?
- **RQ3**: Can we let users choose actors they trust to participate in the communication setup?
  - **RQ3.1**: Do RP require specialised API?
  - **RQ3.2**: Is dynamic discovery and registration commonly available for RP?
  - **RQ3.3**: Do RP requires a trust relationship with the supported IdP?
  - **RQ3.4**: Can we leverage the WebRTC identity architecture to let users chose their IdP for user-to-server authentication?

To answer these research questions, we have proposed three main contributions.

In our first contribution (in Chapter 3), we have studied additional privacy implications focusing on the WebRTC identity architecture and on the role of the IdP.

We first presented our implementation of the WebRTC identity architecture and in particular the integration of the IdP Proxy component with the OpenID Connect (OIDC) protocol. This work reveals that while OIDC facilitates the creation and signature of WebRTC identity assertion, its integration is not straightforward. In particular, although WebRTC offers an abstract authentication delegation interface it is not particularly suited to manage authorization delegation. We then answered to RQ1.1 by showing additional privacy risks that IdP should take in consideration to implement the WebRTC identity architecture. We also showed how the IdP can compromise users' privacy without their explicit consent. The central role and responsibility of IdP is reinforced by their inclusion in WebRTC call setup.

We then focused on answering RQ3 to find if we can let users chose actors they trust to participate in the communication setup. We conducted a survey of the top-500 websites' usage of OAuth 2 and OIDC to identify possible reasons for this situation. We classified Relying Party (RP) by the types of authorization they request to users. Our results show that a majority of RP, 58% of 103, do not require specialised data. However, we also observed that OIDC proposes standardised profile claims, scopes, endpoint, and data format but is implemented by only a few IdP. Similarly, OIDC offers optional standards for dynamic discovery and registration of RP but these are not implemented at all on surveyed IdP. This may be due to necessary trust relations between IdP and RP but our survey did not allow us to answer for that matter. Finally, we conducted a survey, targeted at developers, to identify important IdP's properties in the developer's opinion. Our results show that a consensus exists on the need for a strong authentication and well-crafted user experience but not on other properties. Our conclusion on RQ3.1 and RQ3.2 is that while technical solutions for allowing users to choose their IdP exist, these are not implemented by IdP and RP alike.

In our second contribution (in Chapter 4), we have looked at solutions to give users more control over WebRTC identity parameters: their peers' authentication and their own IdP.

We first proposed Authentication Class and Origin Request (ACOR), a Session Description Protocol (SDP) extension to negotiate the Authentication Class and the IdP's Origin for the authentication of the other party during a WebRTC call. We implemented our solution in a WebRTC service and tested it using Firefox to answer RQ2.1. Our tests revealed that while it is possible to request identity parameters to the other peer, obtaining feedback on the peer's authentication class is not possible at the moment. We believe that this missing feature may be useful even outside of a negotiation use case and that it could easily be supported by the WebRTC identity architecture.

We then presented WebConnect, a web identity metasystem to let users select their trusted IdP. WebConnect answers RQ3.4 and show that the WebRTC identity architecture can be leveraged to build a user-to-server authentication mechanism. We implemented a prototype version based on a Firefox extension and reusing IdP Proxy implemented in Section 3.1. We thus believe that a Web Identity Metasystem such as WebConnect is a good way to give users more control over which identity services they want to use both in WebRTC and on the Web in general.

In our third contribution (in Chapter 5), we have proposed a model representing the security of a WebRTC session so that users may have more understanding of the security of their WebRTC session.

We presented our methodology based on an iterative decomposition process. Our model uses the security parameters of the signalling process, media encryption, but also trust parameters configured by the user. It is a representation of the user's trust in the confidentiality of its session, from its own point-of-view. In addition, we considered the transitive nature of trust relations and proposed an omniscient view of the session confidentiality model. We then discussed an instantiation of our model and propose an utility function for security parameters.

To validate our model, we conducted a preliminary experiment on non-experts users. In this study, we evaluated how users understand the definition of trust in actors of the communication setup and how our trust model helps them understand the security of their WebRTC communication. This study was based on an online survey offering participants to interact with a dynamic implementation our model. This survey helped to identify potential difficulties for the understanding of the model and the measure of users perception.

Chapter 7

Perspectives

In this chapter, we present some research directions emerging from the work presented in this thesis. A few of these perspectives were already envisioned as long-term objective at the start of our work. However, other unveiled from our research results and we believe that researching them may yield interesting contributions. We already started exploring some of these directions. We first expose how we would like to extend our WebRTC trust and security model in Section 7.1. Then we propose ways to continue work on the WebRTC IdP Proxy interface in Section 7.2. Finally, in Section 7.3 we argue for a comparison of our WebConnect solution with the work of the W3C WebPayment working group.

7.1 On the WebRTC Trust and Security Model

Instantiation @Runtime

We discussed in Section 5.3.2 of how we intend to conduct a larger scale survey to validate our trust model. One of our objective is to integrate the model in to a running WebRTC service to give a sense of context to participants. Ultimately, the model is to be used by the browser to display a trust and security indicator to the users. In both situations: the Communication Service (CS) or the browser running the model, we need to instantiate the model from the actual security configuration, *i. e.* we need to access relevant WebRTC statistics. We have proposed what we call an *omniscient* view of our WebRTC trust and security model. To instantiate this omniscient model on an actual WebRTC service, we need to implement a trusted introspection function in the communication setup. This function would let actors feed their own security information to the model. The question of trust in these inputs may pose a difficult challenge to solve.

Our contributions focused on "manual" reconfiguration of the identity parameters, *i. e.* negotiation and free choice of used Identity Provider (IdP) and authentication level. Our work opened the possibility to work on automatic reconfiguration of a WebRTC session for an increased security level. It should be our next step in the direction of allowing users to control their WebRTC security. For instance, we believe that our trust and security model and identity negotiation solution could be integrated with the approach from Alia *et al.* [124] for dynamic reconfiguration of the security and Quality of Service (QoS) parameters. We believe that setting up this work on a given signalling architecture rather than on a signalling-agnostic approach may help in determining reconfiguration options.

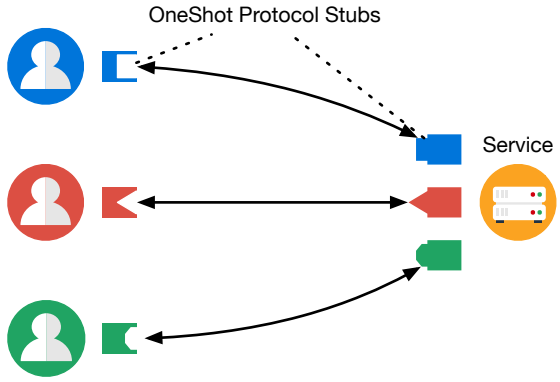


Figure 7.1: One-Shot Protocols Architecture

Trust Contextualisation

Another way to contextualise the WebRTC trust and security model could be to explicitly add context information to the trust and security decomposition. For instance, we considered the possibility to represent explicit context information using a coloured tree in the graphical representation. Using such formalism means that the decomposition nodes in the model are actually typed. Firstly, this would allow to express the exact purpose of a trust relation between two actors. Secondly, the typed decomposition nodes and typed security properties could be extracted from the model. We believe that researching this direction could open the path of a protocol composition language. Such protocol composition language could be used to build on demand security mechanisms based on high-level requirements.

One possible application of a protocol composition language may be to implement protocol diversification *@Runtime*. Automatic diversification techniques [164] aim at reducing software mono-culture and its inherent weakness, *i. e.* break-once break-everywhere vulnerabilities. The idea of automatic protocol diversification is thus to generate protocols on-the-fly for each opened connection. Whether such architecture would actually increase communication security is an open question.

7.2 On the IdP Proxy Interface

Loopback Interface

In some scenarios, especially those for which the WebRTC identity architecture is designed, Alice's IdP may be the only trusted actor in the communication setup. Observing our WebRTC trust and security model, we remarked that Alice's IdP has no influence on Alice's trust in her security. This is a quite important paradox. Of course, it is the responsibility of Bob's IdP to authenticate Bob, but what if this IdP is also not trusted? In Section 4.1, we proposed a solution to let Alice negotiate which IdP may be used by Bob. However, if Alice authenticates too, Alice's IdP Proxy will be instantiated on Bob's browser. This is an important asset that could be leveraged so that Alice's IdP participates in Alice's trust.

Inspired from the ZRTP protocol, a loopback feature, as presented in Figure 7.2, could be implemented so that Alice's IdP confirms who verified the identity assertion. As Alice's IdP does not know Bob, this pose some interesting challenges that need to be solved first. Ultimately, Alice's IdP may not authenticate Bob on a first call. Nevertheless, supposing a first secure call, Alice's IdP may re-authenticate Bob on subsequent call. A simple solution to implement such mechanism would be to use cookies, as we described in Section 3.2.2. Obviously, this implies that the IdP Proxy interface would

have to be modified to incorporate this functionnality. While the WebRTC identity architecture aims at offering an abstract identity interface, this raise the question as to whether such an interface is possible and how it should be designed.

We would also like the opportunity to explore other mechanisms to authenticate the other peer. In particular we would like to explore the possibility to fingerprint the other peer's browser through its WebRTC media, stream, and network parameters. For instance during the Session Description Protocol (SDP) negotiation, offered and selected Interactive Connectivity Establishment (ICE) candidates, offered codecs or other parameters may allow to establish a fingerprint of the browser. It would be interesting to know if such fingerprint could be used in a peer authentication use case or even in a generic web fingerprinting script as in the work of Laperdrix *et al.* [88].

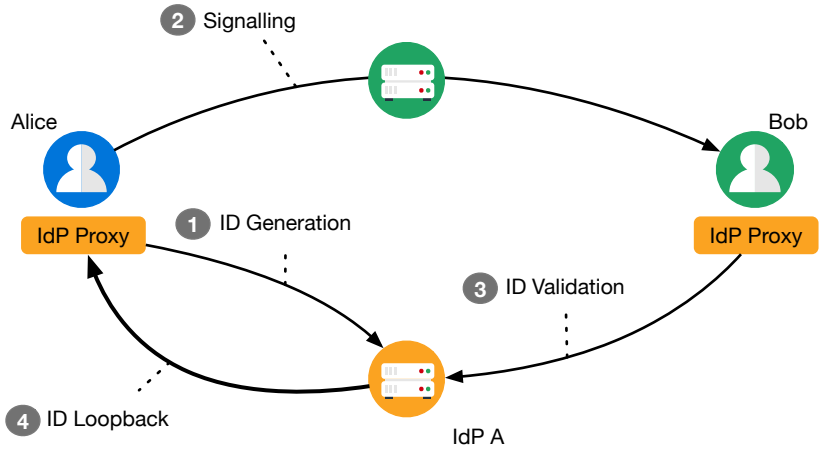


Figure 7.2: ID Loopback Sequence

WebID TLS Implementation

WebID [153] is a distributed identification mechanism which enables each user to control its identity and link to other identities forming a decentralised social network. A WebID document contains claims of an identity in the resource description framework format and is hosted on a secure domain. WebID-TLS [165] is an authentication protocol that leverages WebID and Transport Layer Security (TLS) client authentication. In order to use this protocol, a user first installs its public/private key pair and certificate referring to the WebID document in its browser. The public key is also added to the user's WebID document. When a website requires WebID-TLS authentication, the user select its certificate and the browser performs a TLS client authentication with it. As the certificate refer to the WebID containing the certificate's public key, the protocol proves to the website that the user is in control of the WebID document, *i. e.* it is authenticated.

Work from the WebID W3C working group seems to have ceased. However, as WebID-TLS lets users control their identity and authentication it may be a practical solution to privacy issues related to the role of IdP on the Web. We started working on an integration of WebID-TLS with the WebRTC identity architecture during our thesis. Our idea is to host an IdP Proxy on the same domain as the WebID, without using any IdP. Implementing a WebID-TLS IdP Proxy would allow users to easily host their identity for WebRTC, for instance on their own blog.

The main difficulty of this scenario is that the IdP Proxy must be able to access cryptographic material from the browser stored in client certificates. For the moment we did not manage to solve this issue and we would like to continue to work on this implementation if given the opportunity. An alternative solution could be to relax the use of the WebID-TLS protocol and instead rely on WebID-TLS-like approach. For

instance, the IdP Proxy could use the WebCrypto API [166] to generate, import, and export the private key necessary to sign the identity assertion. This key could then be stored in the browser storage [167]. However, this solution requires to deploy in the WebID host some JavaScript code capable of managing the private key lifecycle, loosing the benefits of the simple WebID-TLS solution. Ultimately, it may be preferable to integrate the WebCrypto API with the browser exposed interface for TLS client certificates selection.

7.3 On WebConnect and the WebPayment Working Group

In Section 3.1 we have demonstrated that the WebRTC identity architecture can be implemented with OpenID Connect or with a more ad-hoc solution, in Section 4.1 we have proposed a solution for negotiating the other party identity parameters and in Section 4.2 we have adapted the architecture to a user-to-server authentication scenario. As we explained in the previous section, we were not able to implement IdP Proxy with the WebID-TLS protocol and in each other of our solutions we have encountered small issues that restrict some functionalities. For instance, compared to OpenID Connect, the WebRTC identity architecture does not allow to obtain an authentication strength information or to authenticate the validating party. In our opinion, *to what extent is the WebRTC identity architecture a generic authentication protocol and what features can be implemented with it* is an interesting and open question.

Web Connect is our solution for letting user choose their identity provider on the Web. The work by the World Wide Web Consortium (W3C) Web Payment working group recently came to our attention. This working group proposes a WebPayment Application Programming Interface (API) [168] that would let browsers expose an API for installing payment application. Web sites can then use this API to request payment to users through the payment application of their choice. Figure 7.3 shows a sequence diagram for a payment. The proposed architecture is actually quite similar to the WebConnect architecture, *i. e.* the payment app is equivalent to the IdP Proxy, the mediator to the browser, and the payment network to the IdP. If the web payment architecture gains traction, it may mean that a similar interface for authentication may benefit from some support.

Offering an Identity Metasystem through the WebPayment API

In order to promote our solution for freely choosing an IdP, it may be interesting to explore the similarities between the IdP Proxy/WebConnect and the WebPayment API. The parallel between payment and authentication protocol are already known. For instance the Diameter protocol for authentication, authorization, and accountability can be extended with credit control applications, while spending bitcoins on a blockchain first requires authentication through a private key. Starting from the idea that emitting a bank cheque with a null value and identity claims is similar to emitting an identity assertion, we would like to test if an authentication protocol can actually be integrated and served through the WebPayment API. This would demonstrate the feasibility of the idea and serve to measure the interest of extending the WebPayment API for authentication scenarios.

Going further, the coupling of identity and payment functions in a single API may have additional uses than simply choosing the IdP. For instance, some solutions against SPam over Internet Telephony (SPIT) propose that users pay an initial fees on first call to deter spammers. In such solution, if the call is SPIT the money is kept by the callee and the caller is added to a blacklist, if the caller is genuine the money is

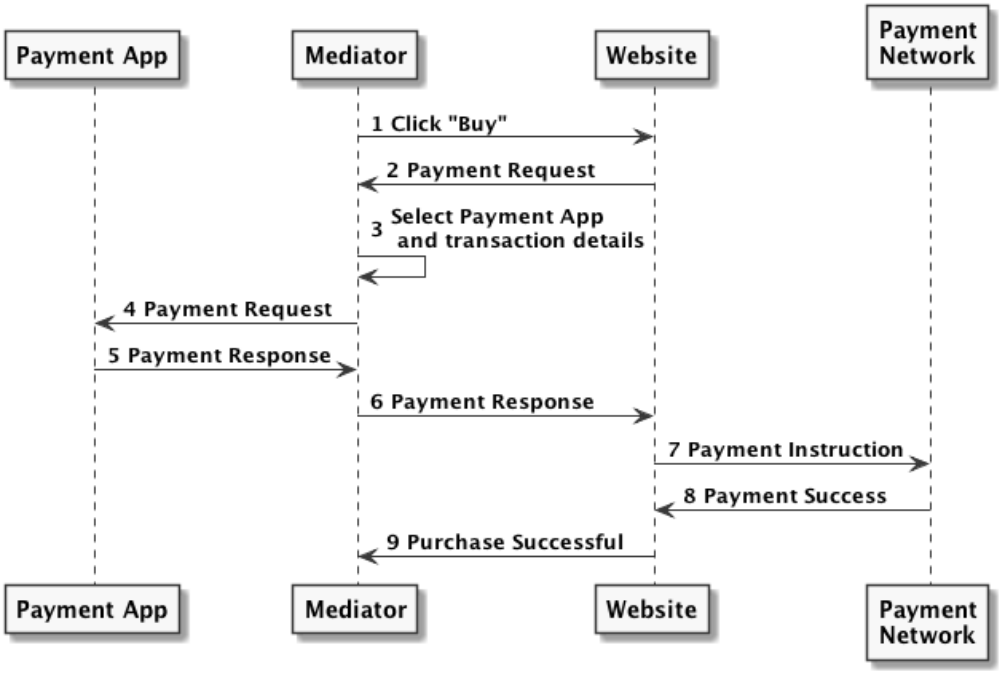


Figure 7.3: Payer Makes a Purchase.

paid back. Supposing that the same interface would offer payment application and peer authentication in WebRTC, such anti-SPIT systems may be quite easy to setup.

A generic API for authorization, authentication, and payment

In this thesis, we have questioned the WebRTC identity architecture as a generic interface for authentication. We have tested its practical implementation with existing protocols and proposed extensions to other use cases. In Telco architectures, authentication, authorization, and accounting are regrouped under the term AAA and often provided by the same protocols. As we remark the similarity between authentication and payment, authentication and authorization are also similar functions as demonstrated by OpenID Connect being an extension to OAuth 2. We believe that the practicality of a generic interface for all-three AAA functions, for instance considering authentication and accounting as a special case of authentication, may be an interesting research direction.

Author's Publications

[1] Kevin Corre, Simon Bécot, Olivier Barais, and Gerson Sunyé. "A WebRTC Extension to Allow Identity Negotiation at Runtime". *Web Engineering - 17th International Conference, ICWE 2017, Rome, Italy, June 5-8, 2017, Proceedings*. Ed. by Jordi Cabot, Roberto De Virgilio, and Riccardo Torlone. Vol. 10360. Lecture Notes in Computer Science. Springer, 2017, pp. 412–419.

[2] Kevin Corre, Olivier Barais, Gerson Sunyé, Vincent Frey, and Jean-Michel Crom. "Why can't users choose their identity providers on the web?" *PoPETs* 2017.3 (2017), pp. 72–86.

[3] Rebecca Copeland, Kevin Corre, Ingo Friese, and Saad El Jaouhari. *Requirements for Trust and Privacy in WebRTC Peer-to-peer Authentication*. Internet-Draft draft-copeland-rtcweb-p2p-idp-auth-00. IETF Secretariat, 2016.

[4] Kevin Corre and Vincent Frey. "Method of managing the authentication of a client in a computing system". WO2017006013 A1 Patent App. PCT/FR2016/051,601. 2016.

[5] Rebecca Copeland, Ahmed Bouabdallah, Ibrahim Javed, Eric Paillet, Simon Bécot, Ewa Janczukowicz, Kevin Corre, Jean-Michel Crom, Paulo Chainho, Felix Beierle, Sebastian Göndör, Frédéric Luart, Adel Al-Hezmi, Andreea Ancuta Corici, Marc Emmelmann, Ricardo Lopes Pereira, Ricardo Chaves, and Nuno Santos. *Framework Architecture Definition*. Deliverable D2.1. reThink Project, 2015.

[6] Jean-Michel Crom, Kevin Corre, Simon Bécot, Ingo Friese, Felix Beierle, Sebastian Göndör, Ahmed Bouabdallah, Marc Emmelmann, Andrea Ancuta Corici, Ricardo Chaves, and Ricardo Pereira. *Management and Security features specifications*. Deliverable D4.1. reThink Project, 2015.

[7] Jean-Michel Crom, Kevin Corre, Simon Bécot, Felix Beierle, Sebastian Göndör, Ahmed Bouabdallah, Saad El Jaouhari, Rebecca Copeland, Marc Emmelmann, Ricardo Chaves Andrea Ancuta-Corici Robert Ende, and Ricardo Pereira. *Implementation of Governance and identity management components for phase 1*. Deliverable D4.2. reThink Project, 2016.

[8] Jean-Michel Crom, Kevin Corre, Ingo Friese, Felix Beierle, Sebastian Göndör, Ahmed Bouabdallah, Hao Jiang, Rebecca Copeland, Ibrahim Tariq Javed, Marc Emmelmann, Andrea Ancuta Corici, Robert Ende, Ricardo Chaves, Nuno Santos, and Ricardo Pereira. *Implementation of Governance and identity management components for phase 2*. Deliverable D4.3. reThink Project, 2017.

[9] Ibrahim Tariq Javed, Rebecca Copeland, Noël Crespi, Marc Emmelmann, Ancuta Corici, Ahmed Bouabdallah, Tuo Zhang, Saad El Jaouhari, Felix Beierle, Sebastian Göndör, Axel Küpper, Kevin Corre, Jean-Michel Crom, Frank Oberle, Ingo Friese, Ana Caldeira, Gil Dias, Nuno Santos, Ricardo Chaves, and Ricardo Lopes Pereira. "Cross-domain identity and discovery framework for web calling services". *Annales des Télécommunications* 72.7-8 (2017), pp. 459–468.

Bibliography

[10] Perseus - Latin Word Study Tool. *communicatio*.

[11] Oxford Dictionaries. *communication*.

[12] Simon Singh. *The code book: the secret history of codes and code-breaking*. 2000.

[13] Mozilla Security Blog. *Communicating the Dangers of Non-Secure HTTP*.

[14] D Harrison McKnight and Norman L Chervany. "The meanings of trust" (1996).

[15] Audun Jøsang and Stéphane Lo Presti. "Analysing the Relationship between Risk and Trust". *Trust Management, Second International Conference, iTrust 2004, Oxford, UK, March 29 - April 1, 2004, Proceedings*. Ed. by Christian Damsgaard Jensen, Stefan Poslad, and Theodosis Dimitrakos. Vol. 2995. Lecture Notes in Computer Science. Springer, 2004, pp. 135–145.

[16] Roy T. Fielding, James Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. `http://www.rfc-editor.org/rfc/rfc2616.txt`. RFC Editor, 1999.

[17] Huahong Tu, Adam Doupé, Ziming Zhao, and Gail-Joon Ahn. "SoK: Everyone Hates Robocalls: A Survey of Techniques Against Telephone Spam". *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 2016, pp. 320–338.

[18] 33700 SPAM. *33700, la plateforme de lutte contre les spams vocaux et SMS*.

[19] ARCEP. *Observatoire des marchés des communications électroniques en France*.

[20] David Gelles and Vindu Goel. *Facebook Enters $16 Billion Deal for WhatsApp*.

[21] Matt Rosoff. *Microsoft Insider: Here's Why We Bought Skype*.

[22] J. Rosenberg and C. Jennings. *The Session Initiation Protocol (SIP) and Spam*. RFC 5039. `http://www.rfc-editor.org/rfc/rfc5039.txt`. RFC Editor, 2008.

[23] Martin Thomson and Keith Griffin. *Screen Capture*. W3C Editor's Draft. W3C, 2017.

[24] François Toutain, Emmanuel Le Huérou, and Eric Beaufils. "On webco interoperability". *Proceedings of the 1st Workshop on All-Web Real-Time Systems, AWeS@EuroSys 2015, Bordeaux, France, April 21, 2015*. Ed. by Emmanuel Bertin, Noël Crespi, and Roch H. Glitho. ACM, 2015, 5:1–5:6.

[25] Matrix.org. *Matrix Specification*.

[26] reThink H2020 project. *reTHINK - Deliverables*.

[27] Harald Alvestrand. *Overview: Real Time Protocols for Browser-based Applications*. Internet-Draft draft-ietf-rtcweb-overview-18. `http://www.ietf.org/internet-drafts/draft-ietf-rtcweb-overview-18.txt`. IETF Secretariat, 2017.

[28] Daniel Burnett, Taylor Brandstetter, Adam Bergkvist, Bernard Aboba, Anant Narayanan, and Cullen Jennings. *WebRTC 1.0: Real-time Communication Between Browsers*. W3C Working Draft. https://www.w3.org/TR/2017/WD-webrtc-20170822/. W3C, 2017.

[29] Daniel Burnett, Anant Narayanan, Bernard Aboba, Cullen Jennings, and Adam Bergkvist. *Media Capture and Streams*. Candidate Recommendation. http://www.w3.org/TR/2016/CR-mediacapture-streams-20160519/. W3C, 2016.

[30] Justin Uberti, Cullen Jennings, and Eric Rescorla. *JavaScript Session Establishment Protocol*. Internet-Draft draft-ietf-rtcweb-jsep-22. `http://www.ietf.org/internet-drafts/draft-ietf-rtcweb-jsep-22.txt`. IETF Secretariat, 2017.

[31] Mark Handley and Van Jacobson. *SDP: Session Description Protocol*. RFC 2327. http://www.rfc-editor.org/rfc/rfc2327.txt. RFC Editor, 1998.

[32] J. Rosenberg. *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols*. RFC 5245. http://www.rfc-editor.org/rfc/rfc5245.txt. RFC Editor, 2010.

[33] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. *Session Traversal Utilities for NAT (STUN)*. RFC 5389. http://www.rfc-editor.org/rfc/rfc5389.txt. RFC Editor, 2008.

[34] R. Mahy, P. Matthews, and J. Rosenberg. *Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)*. RFC 5766. http://www.rfc-editor.org/rfc/rfc5766.txt. RFC Editor, 2010.

[35] *Standards*. 2017.

[36] *Javascript APIs Current Status*. 2017.

[37] A. Barth. *The Web Origin Concept*. RFC 6454. http://www.rfc-editor.org/rfc/rfc6454.txt. RFC Editor, 2011.

[38] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. STD 66. http://www.rfc-editor.org/rfc/rfc3986.txt. RFC Editor, 2005.

[39] Mike West. *Content Security Policy Level 3*. W3C Working Draft. W3C, 2016.

[40] M. Nottingham and E. Hammer-Lahav. *Defining Well-Known Uniform Resource Identifiers (URIs)*. RFC 5785. http://www.rfc-editor.org/rfc/rfc5785.txt. RFC Editor, 2010.

[41] Robert Braden. *Requirements for Internet Hosts - Communication Layers*. STD 3. http://www.rfc-editor.org/rfc/rfc1122.txt. RFC Editor, 1989.

[42] E. Rescorla. *HTTP Over TLS*. RFC 2818. http://www.rfc-editor.org/rfc/rfc2818.txt. RFC Editor, 2000.

[43] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. http://www.rfc-editor.org/rfc/rfc5246.txt. RFC Editor, 2008.

[44] Cormac Herley. "So long, and no thanks for the externalities: the rational rejection of security advice by users". *Proceedings of the 2009 Workshop on New Security Paradigms, Oxford, United Kingdom, September 8-11, 2009*. Ed. by Anil Somayaji and Richard Ford. ACM, 2009, pp. 133–144.

[45] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. *OpenPGP Message Format*. RFC 4880. http://www.rfc-editor.org/rfc/rfc4880.txt. RFC Editor, 2007.

[46] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography engineering: design principles and practical applications*. John Wiley & Sons, 2011.

[47] Eric Rescorla. *WebRTC Security Architecture*. Internet-Draft draft-ietf-rtcweb-security-arch-13. http://www.ietf.org/internet-drafts/draft-ietf-rtcweb-security-arch-13.txt. IETF Secretariat, 2017.

[48] Ilya Grigorik. *High Performance Browser Networking*. OReilly.

[49] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. *The Secure Real-time Transport Protocol (SRTP)*. RFC 3711. http://www.rfc-editor.org/rfc/rfc3711.txt. RFC Editor, 2004.

[50] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. STD 64. http://www.rfc-editor.org/rfc/rfc3550.txt. RFC Editor, 2003.

[51] J. Postel. *User Datagram Protocol*. STD 6. http://www.rfc-editor.org/rfc/rfc768.txt. RFC Editor, 1980.

[52] Jon Postel. *Transmission Control Protocol*. STD 7. http://www.rfc-editor.org/rfc/rfc793.txt. RFC Editor, 1981.

[53] J. Fischl, H. Tschofenig, and E. Rescorla. *Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)*. RFC 5763. RFC Editor, 2010.

[54] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2*. RFC 6347. http://www.rfc-editor.org/rfc/rfc6347.txt. RFC Editor, 2012.

[55] D. McGrew and E. Rescorla. *Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)*. RFC 5764. `http://www.rfc-editor.org/rfc/rfc5764.txt`. RFC Editor, 2010.

[56] R. Stewart. *Stream Control Transmission Protocol*. RFC 4960. `http://www.rfc-editor.org/rfc/rfc4960.txt`. RFC Editor, 2007.

[57] Ewa Janczukowicz, Arnaud Braud, Stéphane Tuffin, Ahmed Bouabdallah, and Jean-Marie Bonnin. "Evaluation of network solutions for improving WebRTC quality". *24th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2016, Split, Croatia, September 22-24, 2016*. IEEE, 2016, pp. 1–5.

[58] Ewa Janczukowicz, Arnaud Braud, Stéphane Tuffin, Gaël Fromentoux, Ahmed Bouabdallah, and Jean-Marie Bonnin. "Specialized network services for WebRTC: TURN-based architecture proposal". *Proceedings of the 1st Workshop on All-Web Real-Time Systems, AWeS@EuroSys 2015, Bordeaux, France, April 21, 2015*. Ed. by Emmanuel Bertin, Noël Crespi, and Roch H. Glitho. ACM, 2015, 3:1–3:6.

[59] *Net neutrality - The current regulatory framework - September 2015*. 2015.

[60] Gergely Alpár, Jaap-Henk Hoepman, and Johanneke Siljee. "The Identity Crisis. Security, Privacy and Usability Issues in Identity Management". *CoRR* abs/1101.0427 (2011).

[61] Abhilasha Bhargav-Spantzel, Jan Camenisch, Thomas Groß, and Dieter Sommer. "User centricity: A taxonomy and open issues". *Journal of Computer Security* 15.5 (2007), pp. 493–527.

[62] Natsuhiko Sakimura, J Bradley, Mike Jones, B de Medeiros, and C Mortimore. *Openid connect core 1.0*. Tech. rep. 2014.

[63] *Information technology – Security techniques – Entity authentication assurance framework*. Standard. Geneva, CH: International Organization for Standardization, 2013.

[64] Julien Hatin, Estelle Cherrier, Jean-Jacques Schwartzmann, V. Frey, and Christophe Rosenberger. "A Continuous LoA Compliant Trust Evaluation Method". *Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016, Rome, Italy, February 19-21, 2016*. Ed. by Olivier Camp, Steven Furnell, and Paolo Mori. SciTePress, 2016, pp. 355–363.

[65] Ahmad Montaser Awal and Abdullah Almaksour. "Classification et extraction des documents complexes à partir des images issues d'un périphérique mobile : application aux documents d'identité". *CORIA 2016 - Conférence en Recherche d'Informations et Applications-13th French Information Retrieval Conference. CIFED 2016 Colloque International Francophone sur l'Ecrit et le Document, Toulouse, France, March 9-11, 2016, Toulouse, France, March 9-11, 2016*. Ed. by Sylvie Calabretto, Bertrand Coüasnon, Lorraine Goeuriot, and Sabine Barrat. ARIA-GRCE, 2016, pp. 575–588.

[66] Ronan Sicre, Ahmad Montaser Awal, and Teddy Furon. "Identity Documents Classification as an Image Classification Problem". *Image Analysis and Processing - ICIAP 2017 - 19th International Conference, Catania, Italy, September 11-15, 2017, Proceedings, Part II*. Ed. by Sebastiano Battiato, Giovanni Gallo, Raimondo Schettini, and Filippo Stanco. Vol. 10485. Lecture Notes in Computer Science. Springer, 2017, pp. 602–613.

[67] Audun Jøsang. "Identity management and trusted interaction in internet and mobile computing". *IET Information Security* 8.2 (2014), pp. 67–79.

[68] M. Jones, J. Bradley, and N. Sakimura. *JSON Web Token (JWT)*. RFC 7519. `http://www.rfc-editor.org/rfc/rfc7519.txt`. RFC Editor, 2015.

[69] T. Bray. *The JavaScript Object Notation (JSON) Data Interchange Format*. RFC 7159. `http://www.rfc-editor.org/rfc/rfc7159.txt`. RFC Editor, 2014.

[70] M. Jones, J. Bradley, and N. Sakimura. *JSON Web Signature (JWS)*. RFC 7515. `http://www.rfc-editor.org/rfc/rfc7515.txt`. RFC Editor, 2015.

[71] M. Jones and J. Hildebrand. *JSON Web Encryption (JWE)*. RFC 7516. RFC Editor, 2015.

[72] *BrowserID: Specifications for Mozilla's Identity Effort*.

[73] Johann Vincent, Sahin Kale, and Vincent Frey. *TIM: Trusted Identity Module*.

[74] E. Hammer-Lahav. *The OAuth 1.0 Protocol*. RFC 5849. `http://www.rfc-editor.org/rfc/rfc5849.txt`. RFC Editor, 2010.

[75] D. Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749. http://www.rfc-editor.org/rfc/rfc6749.txt. RFC Editor, 2012.

[76] F. Andreasen, M. Baugher, and D. Wing. *Session Description Protocol (SDP) Security Descriptions for Media Streams*. RFC 4568. RFC Editor, 2006.

[77] Oscar Ohlsson. *Support of SDES in WebRTC*. Internet-Draft draft-ohlsson-rtcweb-sdes-support-01. http://www.ietf.org/internet-drafts/draft-ohlsson-rtcweb-sdes-support-01.txt. IETF Secretariat, 2012.

[78] P. Zimmermann, A. Johnston, and J. Callas. *ZRTP: Media Path Key Agreement for Unicast Secure RTP*. RFC 6189. http://www.rfc-editor.org/rfc/rfc6189.txt. RFC Editor, 2011.

[79] Dominik Schürmann, Fabian Kabus, Gregor Hildermeier, and Lars C. Wolf. "Wiretapping End-to-End Encrypted VoIP Calls: Real-World Attacks on ZRTP". *PoPETs* 2017.3 (2017), p. 4.

[80] Audun Jøsang and Simon Pope. "Semantic Constraints for Trust Transitivity". *Conceptual Modelling 2005, Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005), Newcastle, NSW, Australia, January/February 2005*. Ed. by Sven Hartmann and Markus Stumptner. Vol. 43. CRPIT. Australian Computer Society, 2005, pp. 59–68.

[81] Donovan Artz and Yolanda Gil. "A survey of trust in computer science and the Semantic Web". *J. Web Sem.* 5.2 (2007), pp. 58–71.

[82] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. "A Computational Model of Trust and Reputation for E-businesses". *35th Hawaii International Conference on System Sciences (HICSS-35 2002), CD-ROM / Abstracts Proceedings, 7-10 January 2002, Big Island, HI, USA*. IEEE Computer Society, 2002, p. 188.

[83] Audun Jøsang, Stephen Marsh, and Simon Pope. "Exploring Different Types of Trust Propagation". *Trust Management, 4th International Conference, iTrust 2006, Pisa, Italy, May 16-19, 2006, Proceedings*. Ed. by Ketil Stølen, William H. Winsborough, Fabio Martinelli, and Fabio Massacci. Vol. 3986. Lecture Notes in Computer Science. Springer, 2006, pp. 179–192.

[84] Mozilla developer's blog. *Battery Status API*.

[85] R. Shirey. *Internet Security Glossary, Version 2*. RFC 4949. http://www.rfc-editor.org/rfc/rfc4949.txt. RFC Editor, 2007.

[86] Céline Castets-Renard. "Quels liens établir entre les USA et l'UE en matière de vie privée et protection des données personnelles ?" *Dalloz IP/IT* (2016), p. 115.

[87] A. Cooper, H. Tschofenig, B. Aboba, J. Peterson, J. Morris, M. Hansen, and R. Smith. *Privacy Considerations for Internet Protocols*. RFC 6973. RFC Editor, 2013.

[88] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. "Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints". *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 2016, pp. 878–894.

[89] European Parliament. "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46". *Official Journal of the European Union (OJ)* 59 (2016), pp. 1–88.

[90] Aaron Johnson, Rob Jansen, Nicholas Hopper, Aaron Segal, and Paul Syverson. "PeerFlow: Secure Load Balancing in Tor". *PoPETs* 2017.2 (2017), pp. 74–94.

[91] James Ball, Glenn Greenwald, and Bruce Schneier. *NSA and GCHQ target Tor network that protects anonymity of web users*. 2013.

[92] Justin Uberti and Guo-wei Shieh. *WebRTC IP Address Handling Requirements*. Internet-Draft draft-ietf-rtcweb-ip-handling-04. IETF Secretariat, 2017.

[93] Jukka K Nurminen, Antony JR Meyn, Eetu Jalonen, Yrjo Raivio, and Raúl Garcıa Marrero. "P2P media streaming with HTML5 and WebRTC". *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*. IEEE. 2013, pp. 63–64.

[94] Christian Vogt, Max Jonas Werner, and Thomas C. Schmidt. "Leveraging WebRTC for P2P content distribution in web browsers". *2013 21st IEEE International Conference on Network Protocols, ICNP 2013, Göttingen, Germany, October 7-10, 2013*. IEEE, 2013, pp. 1–2.

[95] Roberto Roverso and Mikael Högqvist. "Hive.js: Browser-Based Distributed Caching for Adaptive Video Streaming". *2014 IEEE International Symposium on Multimedia, ISM 2014, Taichung, Taiwan, December 10-12, 2014*. IEEE Computer Society, 2014, pp. 143–146.

[96] Liang Zhang, Fangfei Zhou, Alan Mislove, and Ravi Sundaram. "Maygh: building a CDN from client web browsers". *Eighth Eurosys Conference 2013, EuroSys '13, Prague, Czech Republic, April 14-17, 2013*. Ed. by Zdenek Hanzálek, Hermann Härtig, Miguel Castro, and M. Frans Kaashoek. ACM, 2013, pp. 281–294.

[97] Brice Nédelec, Julian Tanke, Davide Frey, Pascal Molli, and Achour Mostefaoui. "Spray: an Adaptive Random Peer Sampling Protocol". PhD thesis. LINA-University of Nantes; INRIA Rennes-Bretagne Atlantique, 2015.

[98] Angelos D. Keromytis. "A Comprehensive Survey of Voice over IP Security Research". *IEEE Communications Surveys and Tutorials* 14.2 (2012), pp. 514–537.

[99] Jonathan Zar, D Endler, D Ghosal, R Jafari, A Karlcut, M Kolenko, N Nguyen, and W Walkoe. "VoIP security and privacy threat taxonomy". *Public Release* 1 (2005), p. 24.

[100] Christoph Sorge and Jan Seedorf. "A Provider-Level Reputation System for Assessing the Quality of SPIT Mitigation Algorithms". *Proceedings of IEEE International Conference on Communications, ICC 2009, Dresden, Germany, 14-18 June 2009*. IEEE, 2009, pp. 1–6.

[101] Kumar Srivastava and Henning G Schulzrinne. "Preventing spam for sip-based instant messages and sessions" (2004).

[102] N Croft and M Olivier. "A model for spam prevention in voice over IP networks using anonymous Verifying Authorities". *Proceedings of the 5th Annual Information Security South Africa Conference (ISSA)*. 2005.

[103] C. Perkins and JM. Valin. *Guidelines for the Use of Variable Bit Rate Audio with Secure RTP*. RFC 6562. RFC Editor, 2012.

[104] Andrew M. White, Austin R. Matthews, Kevin Z. Snow, and Fabian Monrose. "Phonotactic Reconstruction of Encrypted VoIP Conversations: Hookt on Fon-iks". *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*. IEEE Computer Society, 2011, pp. 3–18.

[105] Carlos Aguilar Melchor, Yves Deswarte, and Julien Iguchi-Cartigny. "Closed-Circuit Unobservable Voice over IP". *23rd Annual Computer Security Applications Conference (ACSAC 2007), December 10-14, 2007, Miami Beach, Florida, USA*. IEEE Computer Society, 2007, pp. 119–128.

[106] Ge Zhang and Simone Fischer-Hübner. "Peer-to-Peer VoIP Communications Using Anonymisation Overlay Networks". *Communications and Multimedia Security, 11th IFIP TC 6/TC 11 International Conference, CMS 2010, Linz, Austria, May 31 - June 2, 2010. Proceedings*. Ed. by Bart De Decker and Ingrid Schaumüller-Bichl. Vol. 6109. Lecture Notes in Computer Science. Springer, 2010, pp. 130–141.

[107] Mudhakar Srivatsa, Ling Liu, and Arun Iyengar. "Preserving Caller Anonymity in Voice-over-IP Networks". *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*. IEEE Computer Society, 2008, pp. 50–63.

[108] Stephan Heuser, Bradley Reaves, Praveen Kumar Pendyala, Henry Carter, Alexandra Dmitrienko, William Enck, Negar Kiyavash, Ahmad-Reza Sadeghi, and Patrick Traynor. "Phonion: Practical Protection of Metadata in Telephony Networks". *PoPETs* 2017.1 (2017), pp. 170–187.

[109] Christoph Fuchs, Nils Aschenbruck, Felix Leder, and Peter Martini. "Detecting VoIP based DoS attacks at the public safety answering point". *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*. Ed. by Masayuki Abe and Virgil D. Gligor. ACM, 2008, pp. 148–155.

[110] Ming Luo, Tao Peng, and Christopher Leckie. "CPU-based DoS attacks against SIP servers". *IEEE/IFIP Network Operations and Management Symposium: Pervasive Management for Ubioquitous Networks and Services, NOMS 2008, 7-11 April 2008, Salvador, Bahia, Brazil*. Ed. by Marcus Brunner, Carlos Becker Westphall, and Lisandro Zambenedetti Granville. IEEE, 2008, pp. 41–48.

[111] William Conner and Klara Nahrstedt. "Protecting SIP Proxy Servers from Ringing-Based Denial-of-Service Attacks". *Tenth IEEE International Symposium on Multimedia (ISM2008), December 15-17, 2008, Berkeley, California,*

USA. IEEE Computer Society, 2008, pp. 340–347.

[112] Sven Ehlert, Dimitris Geneiatakis, and Thomas Magedanz. "Survey of network security systems to counter SIP-based denial-of-service attacks". *Computers & Security* 29.2 (2010), pp. 225–243.

[113] Dimitris Geneiatakis, Georgios Kambourakis, and Costas Lambrinoudakis. "A Mechanism for Ensuring the Validity and Accuracy of the Billing Services in IP Telephony". *Trust, Privacy and Security in Digital Business, 5th International Conference, TrustBus 2008, Turin, Italy, September 4-5, 2008, Proceedings*. Ed. by Steven Furnell, Sokratis K. Katsikas, and Antonio Lioy. Vol. 5185. Lecture Notes in Computer Science. Springer, 2008, pp. 59–68.

[114] Prateek Gupta and Vitaly Shmatikov. "Security Analysis of Voice-over-IP Protocols". *20th IEEE Computer Security Foundations Symposium, CSF 2007, 6-8 July 2007, Venice, Italy*. IEEE Computer Society, 2007, pp. 49–63.

[115] Humberto J. Abdelnur, Radu State, Isabelle Chrisment, and C. Popi. "Assessing the security of VoIP Services". *Integrated Network Management, IM 2007. 10th IFIP/IEEE International Symposium on Integrated Network Management, Munich, Germany, 21-25 May 2007*. IEEE, 2007, pp. 373–382.

[116] Salman Baset and Henning Schulzrinne. "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol". *INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 23-29 April 2006, Barcelona, Catalunya, Spain*. IEEE, 2006.

[117] Tom Berson. "Skype security evaluation". *ALR* 31 (2005).

[118] Danny Dolev and Andrew Chi-Chih Yao. "On the security of public key protocols". *IEEE Trans. Information Theory* 29.2 (1983), pp. 198–207.

[119] Claude E Shannon. "Communication theory of secrecy systems*". *Bell system technical journal* 28.4 (1949), pp. 656–715.

[120] Thorsten Kleinjung, Arjen K. Lenstra, Dan Page, and Nigel P. Smart. "Using the Cloud to Determine Key Strengths". *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*. Ed. by Steven D. Galbraith and Mridul Nandi. Vol. 7668. Lecture Notes in Computer Science. Springer, 2012, pp. 17–39.

[121] Arjen Lenstra. *Handbook of Information Security*. Wiley, 2004.

[122] Agence nationale de la sécurité des systèmes d'information. *Le Référentiel général de sécurité (RGS)*. Tech. rep. 2014.

[123] National Institute for Standards and Technology. *Recommendation for Key Management*. Tech. rep. 2016.

[124] Mourad Alia, Marc Lacoste, Ruan He, and Frank Eliassen. "Putting together QoS and security in autonomic pervasive systems". *Q2SWinet'10, Proceedings of the Sixth ACM Symposium on QoS and Security for Wireless and Mobile Networks, Bodrum, Turkey, October 20-21, 2010*. Ed. by Mario Gerla, Matteo Cesana, and Jalel Ben-Othman. ACM, 2010, pp. 19–28.

[125] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. *SIP: Session Initiation Protocol*. RFC 3261. `http://www.rfc-editor.org/rfc/rfc3261.txt`. RFC Editor, 2002.

[126] Feng Cao and Cullen Jennings. "Providing Response Identity and Authentication in IP Telephony". *Proceedings of the The First International Conference on Availability, Reliability and Security, ARES 2006, The International Dependability Conference - Bridging Theory and Practice, April 20-22 2006, Vienna University of Technology, Austria*. IEEE Computer Society, 2006, pp. 198–205.

[127] Mario Di Mauro and Maurizio Longo. "A decision theory based tool for detection of encrypted WebRTC traffic". *18th International Conference on Intelligence in Next Generation Networks, ICIN 2015, Paris, France, February 17-19, 2015*. IEEE, 2015, pp. 89–94.

[128] Mario Di Mauro and Maurizio Longo. "Revealing Encrypted WebRTC Traffic via Machine Learning Tools". *SECRYPT 2015 - Proceedings of the 12th International Conference on Security and Cryptography, Colmar, Alsace, France, 20-22 July, 2015*. Ed. by Mohammad S. Obaidat, Pascal Lorenz, and Pierangela Samarati. SciTePress, 2015, pp. 259–266.

[129] Cullen Jennings, Ted Hardie, and Magnus Westerlund. "Real-time communications for the web". *IEEE Communications Magazine* 51.4 (2013), pp. 20–26.

[130] Richard L. Barnes and Martin Thomson. "Browser-to-Browser Security Assurances for WebRTC". *IEEE Internet Computing* 18.6 (2014), pp. 11–17.

[131] Salvatore Loreto and Simon Pietro Romano. "Real-Time Communications in the Web: Issues, Achievements, and Ongoing Standardization Efforts". *IEEE Internet Computing* 16.5 (2012), pp. 68–73.

[132] Salvatore Loreto and Simon Pietro Romano. "How Far Are We from WebRTC-1.0? An Update on Standards and a Look at What's Next". *IEEE Communications Magazine* 55.7 (2017), pp. 200–207.

[133] Md Habibur Rahaman. "A Survey on Real-Time Communication for Web". *Scientific Research Journal (Scirj)* 3.VII (2015), pp. 39–45.

[134] Alessandro Amirante, Tobia Castaldi, Lorenzo Miniero, and Simon Pietro Romano. "On the seamless interaction between webRTC browsers and SIP-based conferencing systems". *IEEE Communications Magazine* 51.4 (2013), pp. 42–47.

[135] Philippe De Ryck, Wouter Joosen, Frank Piessens, Martin Johns, Elwyn Davies, Bert Bos, Thomas Roessler, Lieven Desmet, Sebastian Lekies, Jan Tobias MÃŒhlberg, et al. *Web-platform security guide: Security assessment of the web ecosystem*. Deliverable D1.1. STREWS Project, 2013.

[136] Bert Bos, Elwyn Davies, Lieven Desmet, Stephen Farrell, Martin Johns, and Rigo Wenning. "Case study 1 Report: WebRTC". D1.2 (2014).

[137] Andreas Reiter and Alexander Marsalek. "WebRTC: your privacy is at risk". *Proceedings of the Symposium on Applied Computing, SAC 2017, Marrakech, Morocco, April 3-7, 2017*. Ed. by Ahmed Seffah, Birgit Penzenstadler, Carina Alves, and Xin Peng. ACM, 2017, pp. 664–669.

[138] Nasser Mohammed Al-Fannah. "One Leak Will Sink A Ship: WebRTC IP Address Leaks". Vol. abs/1709.05395. 2017. arXiv: 1709.05395.

[139] Willem De Groef, Deepak Subramanian, Martin Johns, Frank Piessens, and Lieven Desmet. "Ensuring endpoint authenticity in WebRTC peer-to-peer communication". *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*. Ed. by Sascha Ossowski. ACM, 2016, pp. 2103–2110.

[140] J Muranyi and I Kotuliak. "Identity management in WebRTC domains". *Emerging eLearning Technologies and Applications (ICETA), 2013 IEEE 11th International Conference on*. IEEE. 2013, pp. 289–293.

[141] Li Li, Wu Chou, Zhihong Qiu, and Tao Cai. "Who Is Calling Which Page on the Web?" *IEEE Internet Computing* 18.6 (2014), pp. 26–33.

[142] Victoria Beltran, Emmanuel Bertin, and Noël Crespi. "User Identity for WebRTC Services: A Matter of Trust". *IEEE Internet Computing* 18.6 (2014), pp. 18–25.

[143] Victoria Beltran and Emmanuel Bertin. "Unified communications as a service and WebRTC: An identity-centric perspective". *Computer Communications* 68 (2015), pp. 73–82.

[144] Victoria Beltran and Emmanuel Bertin. "Identity management for Web business communications". *18th International Conference on Intelligence in Next Generation Networks, ICIN 2015, Paris, France, February 17-19, 2015*. IEEE, 2015, pp. 103–107.

[145] Ibrahim Tariq Javed, Khalifa Toumi, and Noel Crespi. "Browser-to-browser authentication and trust relationships for WebRTC". *UBICOMM 2016 : 10th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*. Venice, Italy, 2016, pp. 9–16.

[146] Ibrahim Tariq Javed, Khalifa Toumi, Noël Crespi, and Amir Mohammadinejad. "Br2Br: A Vector-Based Trust Framework for WebRTC Calling Services". *18th IEEE International Conference on High Performance Computing and Communications; 14th IEEE International Conference on Smart City; 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016, Sydney, Australia, December 12-14, 2016*. Ed. by Jinjun Chen and Laurence T. Yang. IEEE, 2016, pp. 522–529.

[147] Rebecca Copeland and Michael Copeland. "A Question of Quality–VoIP, WebRTC or VoLTE?" *19th Conference on Innovations in*

Clouds, Internet and Networks, ICIN 2016, Paris, France, March, 2016. 2016.

[148] Ibrahim Tariq Javed, Khalifa Toumi, and Noël Crespi. "TrustCall: A Trust Computation Model for Web Conversational Services". *IEEE Access* 5 (2017), pp. 24376–24388.

[149] Anna Vapen, Niklas Carlsson, Anirban Mahanti, and Nahid Shahmehri. "Information Sharing and User Privacy in the Third-Party Identity Management Landscape". *ICT Systems Security and Privacy Protection - 30th IFIP TC 11 International Conference, SEC 2015, Hamburg, Germany, May 26-28, 2015, Proceedings*. Ed. by Hannes Federrath and Dieter Gollmann. Vol. 455. IFIP Advances in Information and Communication Technology. Springer, 2015, pp. 174–188.

[150] Alan Johnston, John Yoakum, and Kundan Singh. "Taking on webRTC in an enterprise". *IEEE Communications Magazine* 51.4 (2013), pp. 48–54.

[151] Kundan Singh, John Yoakum, and Alan Johnston. "Enterprise WebRTC Powered by Browser Extensions". *Proceedings of the Principles, Systems and Applications on IP Telecommunications, IPTComm 2015, Chicago, IL, USA, October 6-8, 2015*. ACM, 2015, pp. 1–6.

[152] San-Tsai Sun, Eric Pospisil, Ildar Muslukhov, Nuray Dindar, Kirstie Hawkey, and Konstantin Beznosov. "What makes users refuse web single sign-on?: an empirical investigation of OpenID". *Symposium On Usable Privacy and Security, SOUPS '11, Pittsburgh, PA, USA - July 20 - 22, 2011*. Ed. by Lorrie Faith Cranor. ACM, 2011, p. 4.

[153] Manu Sporny, Toby Inkster, Henry Story, Bruno Harbulot, and Reto Bachmann-Gmür. *Webid 1.0: Web identification and discovery*. W3C Editor's Draft. W3C, 2011.

[154] Nat Sakimura, John Bradley, M Jones, and Edmund Jay. *OpenID Connect Discovery 1.0*. OpenID Specification. OpenID Foundation, 2014.

[155] N Sakimura, J Bradley, and M Jones. *OpenID connect dynamic client registration 1.0*. OpenID Specification. OpenID Foundation, 2014.

[156] P. Jones, G. Salgueiro, M. Jones, and J. Smarr. *WebFinger*. RFC 7033. RFC Editor, 2013.

[157] J. Richer, M. Jones, J. Bradley, M. Machulak, and P. Hunt. *OAuth 2.0 Dynamic Client Registration Protocol*. RFC 7591. RFC Editor, 2015.

[158] Kim Cameron. "The laws of identity." *Microsoft Whitepaper* (2005).

[159] David Chappell. "Introducing Windows CardSpace, April 2006". *Microsoft Whitepaper* ().

[160] Mike Jones. "Microsoft's vision for an identity metasystem". *Microsoft Whitepaper* (2005).

[161] Pietraszak Mike. *Browser Extensions*. Community Group Draft Report. W3C, 2017.

[162] Audun Jøsang. "Artificial reasoning with subjective logic". *Proceedings of the second Australian workshop on commonsense reasoning*. Vol. 48. Perth:[sn]. 1997, p. 34.

[163] Bruce Schneier. "Attack trees". *Dr. Dobb's journal* 24.12 (1999), pp. 21–29.

[164] Simon Allier, Olivier Barais, Benoit Baudry, Johann Bourcier, Erwan Daubert, Franck Fleurey, Martin Monperrus, Hui Song, and Maxime Tricoire. "Multi-tier diversification in Web-based software applications". *IEEE Software* 32.1 (2015), pp. 83–90.

[165] *WebID-TLS, WebID Authentication over TLS*. W3C Editor's Draft. W3C, 2013.

[166] Mark Watson. *Web Cryptography API*. W3C Recommendation. https://www.w3.org/TR/2017/REC-WebCryptoAPI-20170126/. W3C, 2017.

[167] Ian Hickson. *Web Storage (Second Edition)*. W3C Recommendation. http://www.w3.org/TR/2016/REC-webstorage-20160419/. W3C, 2016.

[168] Domenic Denicola, Marcos Caceres, Adrian Bateman, Roy McElmurry, and Zach Koch. *Payment Request API*. Candidate Recommendation. https://www.w3.org/TR/2018/CR-payment-request-20180123/. W3C, 2018.

Glossary

3GPP The 3rd Generation Partnership Project is a cooperation organisation between telecommunication standard organisations.

AAA Authentication, Authorization, and Accounting
AAL Authentication Assurance Level
ACME The Automatic Certificate Management Environment is a protocol that a certification authority (CA) and an applicant can use to automate the process of verification and certificate issuance [**I-D.ietf-acme-acme**].
ACOR Authentication Class and Origin Request is our proposed extension to SDP for the negotiation of identity parameters.
ACR Authentication Class Request is a parameter of out ACOR extension to SDP.
ACR Authentication Context Class Reference (OIDC claim)
API An Application Programming Interface is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API.
ARCEP Autorité de Régulation des Communications Électroniques et des Postes is the French regulation authority for postal and electronic communications.
AS Authorization Server (OAuth 2 role)

CA Certification Authority are responsible for signing and issuing cryptographic certificates.
CDN A Content Delivery Network is constituted of proxy servers geographically distributed to be as close to clients as possible in order to transparently provide high availability and performance.
CORS Cross-Origin Resource Sharing is a web security mechanism to allow resources on a webpage to be requested from an origin outside of the webpage's origin (see RFC6454 [37]).
CS Communication Service
CSCF Call Session Control Function
CSP Content Security Policies is a mechanism by which web developers can control the resources which a particular page can fetch or execute, as well as a number of security-relevant policy decisions [39].
CSRF Cross-Site Request Forgery is a type of vulnerability allowing an attacker to issue unauthorized commands in name of the user.
CVE The Common Vulnerabilities and Exposure database references public security vulnerabilities.

DDoS Distributed Denial of Service
DHT A Distributed Hash Table is a distributed storage infrastructure providing a key/value lookup functionality.
DoS A Denial of Service attack targets the availability of a machine or network, usually by flooding the target with illegitimate requests.
DTLS The DTLS protocol provides communications privacy for datagram protocols. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. The DTLS protocol is based on the Transport Layer Security (TLS) protocol and provides equivalent security guarantees. Datagram semantics of the underlying transport are preserved by the DTLS protocol (see RFC6347 [54]).

Ecma The European association for standardizing information and communication systems is a standardisation organisation responsible in particular for the ECMAScript language, *i. e.* JavaScript.
EU European Union.

GDPR The General Data Protection is an European Union regulation on the protection of natural persons with regard to the processing of personal data and on the free movement of such data.
GUID Globally Unique IDentifier (reThink project)

HTTP The Hypertext Transfer Protocol is an application-level protocol for distributed, collaborative,

hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers (see RFC2616 [16]).
HTTPS HyperText Transfer Protocol Secured (see RFC2818 [42]).

ICE Internet Connectivity Establishment is a protocol for Network Address Translator (NAT) traversal for UDP-based multimedia sessions established with the offer/answer model. ICE makes use of the Session Traversal Utilities for NAT (STUN) protocol and its extension, Traversal Using Relay NAT (TURN). ICE can be used by any protocol utilizing the offer/answer model, such as the Session Initiation Protocol (SIP) (see RFC5245 [32]).
IdP Identity Provider
IETF The Internet Engineering Task Force (IETF) is the premier Internet standards body, developing open standards through open processes.
IMS The IP Multimedia Subsystem aims at merging telecommunication technologies under an all-IP environment.
IP The Internet Protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses (see RFC791 [**RFC791**]).

JKU JSON Key URL (a JWT header parameter).
JOSE JavaScript Object Signing and Encryption
JSEP JavaScript Session Establishment Protocol allows a JavaScript application to control the signaling plane of a multimedia session via the interface specified in the W3C RTCPeerConnection API, and discusses how this relates to existing signaling protocols [30].
JSON JavaScript Object Notation is a lightweight, text-based, language-independent data interchange format derived from the ECMAScript (see RFC7159 [69]).
JWE JSON Web Token Encryption
JWS JSON Web Token Signature
JWT JSON Web Token is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted (see RFC7519 [68]).

LTE The Long Term Evolution mobile network is a standard for high-speed mobile networks designed by the 3GPP.

MitM A Man-in-the-Middle attack is a kind of cryptographic attack where the attacker is setup between two communicating parties and secretly intercepts and relays their messages.

NAT A Network Address Translator re-maps an IP address into another by modifying packet headers while in transit. It is usually used to hide an private IP address space into a single public IP address.

OIDC OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol [62].
OR Origin Request is a parameter of our ACOR extension to SDP.
OS Operating System
OSN An Online Social Network is a platform where users build social relations based on personal or professional interests.
OTT Over The Top services are provided on top of existing internet service providers networks.

P2P Peer-to-Peer
POSIX The Portable Operating System Interface is a set of standards of compatibility between operating systems.

QoS Quality of Service is the description or measurement of the performance of a service, in particular as seen by the user.

RFC Request For Comments are memorandum published by the Internet Engineering Task Force (IETF).
RO Resource Owner (OAuth 2 role)
RP A Relying Party is an OAuth 2 client using OpenID Connect.
RS Resource Server (OAuth 2 role)
RTCP The RTP Control Protocol is a protocol for controlling RTP sessions (see RFC3550 [50]).
RTP The Real-time Transport Protocol provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services (see RFC3550 [50]).

SAS Short Authentication String
SCTP The Stream Control Transmission Protocol is designed to transport Public Switched Telephone Network (PSTN) signalling messages over IP networks, but is capable of broader applications (see RFC4960 [56]).

SDES The Session Description Protocol Security Descriptions serves to configure security for a unicast media stream in either a single message or a roundtrip exchange (see RFC4568 [76]).
SDK A Software Development Kit is a set of tools and libraries facilitating the development of applications.
SDP Session Description Protocol is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation (see RFC4566 [**RFC4566**]).
SIP The Session Initiation Protocol is an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences (see RFC3261 [125]).
SPIT SPam over Internet Telephony
SRTP The Secure Real-time Transport Protocol is a secure profile for the RTP protocol (see RFC3711 [49]).
SRTP profile for DTLS is an extension of the Datagram Transport Layer Security (DTLS) protocol to establish keys for Secure Real-time Transport Protocol (SRTP) (see RFC5763 [53]).
SSO Single Sign-On systems permit users to log in with a single identifier and password to access a set of systems.
STUN Session Traversal Utilities for NAT is a protocol that serves as a tool for other protocols in dealing with Network Address Translator (NAT) traversal (see RFC5389 [33]).

TCB Trusted Computing Base
TCP The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks (see RFC793 [52].
TLS Transport Layer Security provides communications security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery (see RFC5246 [43]).
TURN Traversal Using Relays around NAT allows the traversal of NAT router through a TURN server relay (see RFC5766 [34]).

UA A User-Agent is an HTTP client.
UDP The User Datagram Protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed (see RFC768 [51]).
UE User Equipement (LTE role)
UI User Inteface
UICC The Universal Integrated Circuit Card is a smart card used for mobile phones authentication and security, often referred as the SIM card.
URI A Uniform Resource Identifier is a compact sequence of characters that identifies an abstract or physical resource (see RFC3986 [38]).
URL Uniform Resource Locator (see RFC3986 [38]).

VBR Variable Bit-Rate files vary the amount of output data per time segment, as opposed to constant bit-rate.
VoIP Voice over IP designates the techniques to communicate using voice or voice and video over any compatible IP networks.
VoIPSA The VoIP Security Alliance aims to fill the void of VoIP security related resources through a unique collaboration of VoIP and Information Security vendors, providers, and thought leaders.
VoLTE Voice over LTE is an architecture for VoIP over 4G mobile networks.
VPN A Virtual Private Network extends a private network through a public network, often over a secure tunnel established with TLS.

W3C The World Wide Web Consortium (W3C) is an international community where Member organizations, a full-time staff, and the public work together to develop Web standards.

XSS Cross-Site Scripting is a vulnerability of web applications allowing an attacker to inject malicious scripts from a domain into another.

List of Figures

6 A Scytale 1
7 Browser Security Indications 2
8 Overview of our Contributions 5

1.1 WebRTC deployment with two browser endpoints and two signalling servers [27]. 12
1.2 WebRTC browser endpoint model [27]. . 12
1.3 Session Description Protocol syntax . . 13
1.4 Example of a SDP message (answer) . . 14
1.6 Cross-site Request Example 15
1.5 Percentage of recorded browsers visiting `w3schools.com` and `amiunique.org` in June 2017. 15
1.7 Web Browser Security Indications 17
1.9 Public-key encryption of m from Alice to Bob [46]. 17
1.8 X.509 Certificate 18
1.10 Public-key signature of a message m from Alice to Bob [46]. 18
1.12 WebRTC Protocol Stack [48]. 19
1.11 Cypher Suit Example 19
1.13 WebRTC Man-in-the-Middle Attack . . 22
1.14 WebRTC Identity Architecture 23
1.15 Interface Exposed by Identity Providers in WebIDL. 23
1.16 Evolution of standards and technologies for user identity management, updated from Jøsang's 2014 survey [67]. 24
1.17 A JWT Example: OIDC ID Token . . . 25
1.18 OAuth 2 Process Example 26
1.19 Abstract Oauth 2 code flow taken from RFC 6749. 27
1.20 Phone-X application displaying the ZRTP SAS. 29
1.21 Trust representation 30
a Trust relationship 30
b Trust transitivity 30
1.22 Tor Onion Encryption 34
1.23 Silo Architecture 35
1.24 Simplified VoLTE Architecture 36
1.25 E2E and E2AE IMS Encryption 37
1.26 Matrix Messaging Federation 38
1.27 Simplified reThink architecture 39
1.28 reThink Identity Discovery 40
1.29 Spray Architecture 40

2.1 Classification tree [98] 44
2.2 VBR Transcripts Reconstruction 46
2.3 Denial of Service Flooding Attack 47
2.4 Alia *et al.* [124] Utility Function 49
2.5 Paper collection search strings used on Scholar. 50
2.6 Javed *et al.* [146] WebRTC trust model. 55
2.7 Service Mode decision process based on context and QUSA profiles. 56
2.8 Overview of our Scientific Methodology 63

3.1 Overview of our Contributions: Study of the WebRTC Identity Architecture. . . . 65
3.2 Local Authentication Architecture . . . 66
3.3 RTCIdentityAssertionResult specification in WebIDL 67
3.4 RTCIdentityValidationResult specification in WebIDL. 67
3.5 Local Identity Assertion Generation . . 68
3.6 Local Identity Assertion Verification . . 69
3.7 OIDC Assertion Generation 71
3.8 Google Identity Selection Page 73
3.9 JS Code for WebRTC Identity Collection 74
3.10 Example of an OAuth 2 request collected by our extension. 75
3.11 Classification of RP-IdP relationships. . 77
3.12 BreizhCamp Developer Survey 82
3.13 Mean, minimum, and standard deviation of the trust survey results. 83

4.1 Overview of our Contributions: Controlling the WebRTC Identity Parameters. . 85
4.2 SDP Offer 89
4.3 SDP Answer 90
4.4 Identity negotiation interface on a WebRTC service. 91

4.5 CardSpace User Interface 93
4.6 WebConnect Architecture 94
4.7 WebConnect identity assertion management sequence diagram. 95
4.8 WebConnect interface specification in WebIDL. 96
4.9 Prototype user interface for the Authentication web API. 98

5.1 Overview of our Contributions: Modelling the WebRTC Trust and Security. . 101
5.2 Javed *et al.* TrustCall architecture [148]. 102
5.3 Alia *et al.* [124] Overall Utility Function 103
5.4 Attack Tree Example 104
5.5 Alice's trust in the confidentiality of her WebRTC session. 105
5.6 Alice's trust in Bob's authenticity resulting from the signalling process. 106
5.7 Alice trust in Bob's identity path. . . . 107
5.8 Alice's trust in the confidentiality of the peer-to-peer media streams. 108
5.10 Comparative Security Utility Function . 108
5.9 Overall trust of Alice in the confidentiality of her WebRTC session. 109
5.11 Overall computational formula for Alice trust in her WebRTC session. 111
5.12 Survey: Trust in Communication Scenarios. 113
5.13 WebRTC Trust and Security Model implementation in D3.js. 115
5.14 Survey: Interest in the WebRTC Trust and Security Model. 116
5.15 This JavaScript code prints `RTCTransportStats`. However, on Firefox it returns no element, and on Chrome the returned stat does not include `dtlsCipher` or `srtpCipher` elements. . . 118

7.1 One-Shot Protocols Architecture 126
7.2 ID Loopback Sequence 127
7.3 Payer Makes a Purchase. 129

List of Tables

2.1 Classification of VoIP security papers returned by our search. 51
2.2 User privacy properties in identity provision model [142]. 54
2.3 Reviewed WebRTC Papers 57

3.1 JS code lines for local IdP Proxy 72
3.2 Observed Relying Parties' Classes 78
3.3 Observed OIDC and discovery features Implementations 80

4.1 Comparison of communication setup actors to act as an identity recommendation source. 87
4.2 Capability to Implement the Proposed Solution 91
4.3 Code lines written for the prototype implementation 99

5.1 Alia *et al.* [124] Confidentiality Utility Function 103
5.2 Security element instantiation based on ANSSI recommendations [122]. 110
5.3 Survey results: Trust in Audio and Video Communications 114
5.4 Survey results: Interest of the Trust and Security Model 117