Internet Engineering Task Force (IETF)
Request for Comments: 7831
Category: Informational
ISSN: 2070-1721

J. Howlett
Jisc
S. Hartman
Painless Security
H. Tschofenig
ARM Ltd.
J. Schaad
August Cellars

May 2016

Application Bridging for Federated Access Beyond Web (ABFAB) Architecture

Abstract

Over the last decade, a substantial amount of work has occurred in the space of federated access management. Most of this effort has focused on two use cases: network access and web-based access. However, the solutions to these use cases that have been proposed and deployed tend to have few building blocks in common.

This memo describes an architecture that makes use of extensions to the commonly used security mechanisms for both federated and non-federated access management, including the Remote Authentication Dial-In User Service (RADIUS), the Generic Security Service Application Program Interface (GSS-API), the Extensible Authentication Protocol (EAP), and the Security Assertion Markup Language (SAML). The architecture addresses the problem of federated access management to primarily non-web-based services, in a manner that will scale to large numbers of Identity Providers, Relying Parties, and federations.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7831.

Table of Contents

1. Introduction

Numerous security mechanisms have been deployed on the Internet to manage access to various resources. These mechanisms have been generalized and scaled over the last decade through mechanisms such as the Simple Authentication and Security Layer (SASL) with the Generic Security Server Application Program Interface (GSS-API) (known as the GS2 family) [RFC5801]; the Security Assertion Markup Language (SAML) [OASIS.saml-core-2.0-os]; and the Authentication, Authorization, and Accounting (AAA) architecture as embodied in RADIUS [RFC2865] and Diameter [RFC6733].

A Relying Party (RP) is the entity that manages access to some resource. The entity that is requesting access to that resource is often described as the client. Many security mechanisms are manifested as an exchange of information between these entities. The RP is therefore able to decide whether the client is authorized or not.

Some security mechanisms allow the RP to delegate aspects of the access management decision to an entity called the Identity Provider (IdP). This delegation requires technical signaling, trust, and a common understanding of semantics between the RP and IdP. These aspects are generally managed within a relationship known as a "federation". This style of access management is accordingly described as "federated access management".

Federated access management has evolved over the last decade through specifications like SAML [OASIS.saml-core-2.0-os], OpenID (http://www.openid.net), OAuth [RFC6749], and WS-Trust [WS-TRUST]. The benefits of federated access management include:

Single or simplified sign-on:

> An Internet service can delegate access management, and the associated responsibilities such as identity management and credentialing, to an organization that already has a long-term relationship with the client. This is often attractive, as RPs frequently do not want these responsibilities. The client also requires fewer credentials, which is also desirable.

Data minimization and user participation:

Often, an RP does not need to know the identity of a client to reach an access management decision. It is frequently only necessary for the RP to know specific attributes about the client -- for example, that the client is affiliated with a particular organization or has a certain role or entitlement. Sometimes, the RP only needs to know a pseudonym of the client.

Prior to the release of attributes to the RP from the IdP, the IdP will check configuration and policy to determine if the attributes are to be released. There is currently no direct client participation in this decision.

Provisioning:

Sometimes, an RP needs, or would like, to know more about a client than an affiliation or a pseudonym. For example, an RP may want the client's email address or name. Some federated access management technologies provide the ability for the IdP to supply this information, either on request by the RP or unsolicited.

This memo describes the Application Bridging for Federated Access Beyond web (ABFAB) architecture. This architecture addresses the problem of federated access management primarily for non-web-based services. This architecture makes use of extensions to the commonly used security mechanisms for both federated and non-federated access management, including RADIUS, the Generic Security Service (GSS), the Extensible Authentication Protocol (EAP), and SAML. The architecture should be extended to use Diameter in the future. It does so in a manner that is designed to scale to large numbers of IdPs, RPs, and federations.

1.1. Terminology

This document uses identity management and privacy terminology from [RFC6973]. In particular, this document uses the terms "identity provider", "relying party", "identifier", "pseudonymity", "unlinkability", and "anonymity".

In this architecture, the IdP consists of the following components: an EAP server, a RADIUS server, and, optionally, a SAML Assertion service.

This document uses the term "Network Access Identifier" (NAI) as defined in [RFC7542]. An NAI consists of a realm identifier, which is associated with a AAA server, and thus an IdP and a username, that are associated with a specific client of the IdP.

One of the problems some people have found with reading this document is that the terminology sometimes appears to be inconsistent. This is because the various standards that we refer to use different terms for the same concept. In general, this document uses either the ABFAB term or the term associated with the standard under discussion, as appropriate. For reference, we include Table 1 below, which provides a mapping for these different terms. (Note that items marked "N/A" (not applicable) indicate that there is no name that represents the entity.)

Protocol	Client	Relying Party	Identity Provider
ABFAB	N/A	Relying Party (RP)	Identity Provider (IdP)
	Initiator	Acceptor	N/A
	Client	Server	N/A
SAML	Subject	Service provider	Issuer
GSS-API	Initiator	Acceptor	N/A
EAP	EAP peer	EAP authenticator	EAP server
AAA	N/A	AAA client	AAA server
RADIUS	user	NAS	N/A
	N/A	RADIUS client	RADIUS server

Table 1: Terminology

1.1.1. Channel Binding

This document uses the term "channel binding" in two different contexts; this term has a different meaning in each of these contexts.

EAP channel binding is used to implement GSS-API naming semantics. EAP channel binding sends a set of attributes from the peer to the EAP server either as part of the EAP conversation or as part of a secure association protocol. In addition, attributes are sent in the back-end protocol from the EAP authenticator to the EAP server. The

EAP server confirms the consistency of these attributes and provides the confirmation back to the peer. In this document, channel binding without qualification refers to EAP channel binding.

GSS-API channel binding provides protection against man-in-the-middle attacks when GSS-API is used for authentication inside of some tunnel; it is similar to a facility called "cryptographic binding" in EAP. The binding works by each side deriving a cryptographic value from the tunnel itself and then using that cryptographic value to prove to the other side that it knows the value.

See [RFC5056] for a discussion of the differences between these two facilities. These differences can be summarized as follows:

- GSS-API channel binding specifies that there is nobody between the client and the EAP authenticator.

- EAP channel binding allows the client to have knowledge of such EAP authenticator attributes as the EAP authenticator’s name.

Typically, when considering both EAP and GSS-API channel binding, people think of channel binding in combination with mutual authentication. This is sufficiently common that, without additional qualification, channel binding should be assumed to imply mutual authentication. In GSS-API, without mutual authentication, only the acceptor has authenticated the initiator. Similarly, in EAP, only the EAP server has authenticated the peer. Sometimes, one-way authentication is useful. Consider, for example, a user who wishes to access a protected resource for a shared whiteboard in a conference room. The whiteboard is the acceptor; it knows that the initiator is authorized to give it a presentation, and the user can validate that the whiteboard got the correct presentation by visual means. (The presentation should not be confidential in this case.) If channel binding is used without mutual authentication, it is effectively a request to disclose the resource in the context of a particular channel. Such an authentication would be similar in concept to a holder-of-key SAML Assertion. However, note also that although it is not happening in the protocol, mutual authentication is happening in the overall system: the user is able to visually authenticate the content. This is consistent with all uses of channel binding without protocol-level mutual authentication found so far.

1.2. An Overview of Federation

In the previous section, we introduced the following entities:

- the client,
- the IdP, and
- the RP.

The final entity that needs to be introduced is the Individual. An Individual is a human being that is using the client. In any given situation, an Individual may or may not exist. Clients can act as front ends for Individuals, or clients may be independent entities that are set up and allowed to run autonomously. An example of such an independent entity can be found in the Trust Router Protocol (https://www.ietf.org/proceedings/86/slides/slides-86-rtgarea-0.pdf), where the routers use ABFAB to authenticate to each other.

These entities and their relationships are illustrated graphically in Figure 1.

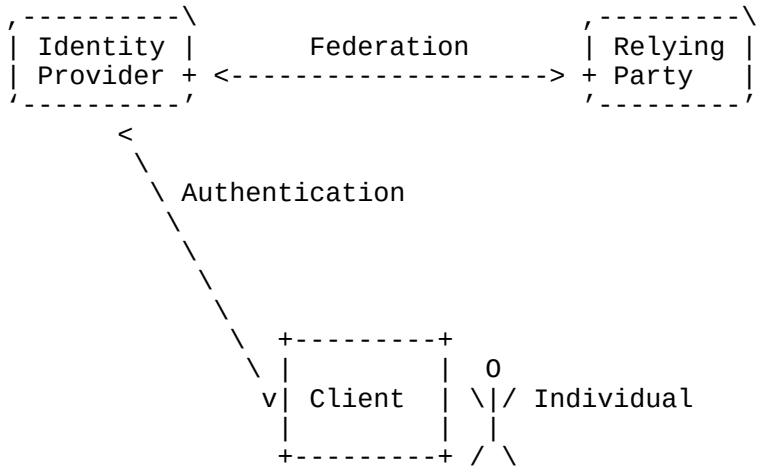


Figure 1: Entities and Their Relationships

The relationships between the entities in Figure 1 are as follows:

Federation

The IdP and the RPs are part of a federation. The relationship may be direct (they have an explicit trust relationship) or transitive (the trust relationship is mediated by one or more entities). The federation relationship is governed by a federation agreement. Within a single federation, there may be multiple IdPs as well as multiple RPs.

Authentication

There is a direct relationship between the client and the IdP. This relationship provides the means by which they trust each other and can securely authenticate each other.

A federation agreement typically encompasses operational specifications and legal rules:

Operational Specifications:

The goal of operational specifications is to provide enough definition that the system works and interoperability is possible. These include the technical specifications (e.g., protocols used to communicate between the three parties), process standards, policies, identity proofing, credential and authentication algorithm requirements, performance requirements, assessment and audit criteria, etc.

Legal Rules:

The legal rules take the legal framework into consideration and provide contractual obligations for each entity. The rules define the responsibilities of each party and provide further clarification of the operational specifications. These legal rules regulate the operational specifications, make operational specifications legally binding to the participants, and define and govern the rights and responsibilities of the participants. The legal rules may, for example, describe liability for losses, termination rights, enforcement mechanisms, measures of damage, dispute resolution, warranties, etc.

The operational specifications can demand the usage of a specific technical infrastructure, including requirements on the message routing intermediaries, to offer the required technical functionality. In other environments, the operational specifications require fewer technical components in order to meet the required technical functionality.

The legal rules include many non-technical aspects of federation, such as business practices and legal arrangements, which are outside the scope of the IETF. The legal rules can still have an impact on the architectural setup or on how to ensure the dynamic establishment of trust.

While a federation agreement is often discussed within the context of formal relationships, such as between an enterprise and an employee or between a government and a citizen, a federation agreement does not have to require any particular level of formality. For an IdP and a client, it is sufficient for a relationship to be established by something as simple as using a web form and confirmation email. For an IdP and an RP, it is sufficient for the IdP to publish contact information along with a public key and for the RP to use that data. Within the framework of ABFAB, it will generally be required that a mechanism exist for the IdP to be able to trust the identity of the RP; if this is not present, then the IdP cannot provide the assurances to the client that the identity of the RP has been established.

The nature of federation dictates that there exists some form of relationship between the IdP and the RP. This is particularly important when the RP wants to use information obtained from the IdP for access management decisions and when the IdP does not want to release information to every RP (or only under certain conditions).

While it is possible to have a bilateral agreement between every IdP and every RP, on an Internet scale, this setup requires the introduction of the multilateral federation concept, as the management of such pair-wise relationships would otherwise prove burdensome.

The IdP will typically have a long-term relationship with the client. This relationship typically involves the IdP positively identifying and credentialing the client (for example, at the time of employment within an organization). When dealing with Individuals, this process is called "identity proofing" [NIST-SP.800-63-2]. The relationship will often be instantiated within an agreement between the IdP and the client (for example, within an employment contract or terms of use that stipulate the appropriate use of credentials and so forth).

The nature and quality of the relationship between the client and the IdP are important contributors to the level of trust that an RP may assign to an assertion describing a client made by an IdP. This is sometimes described as the level of assurance [NIST-SP.800-63-2].

Federation does not require an a priori relationship or a long-term relationship between the RP and the client; it is this property of federation that yields many of its benefits. However, federation does not preclude the possibility of a pre-existing relationship between the RP and the client or the possibility that the RP and client may use the introduction to create a new long-term relationship independent of the federation.

Finally, it is important to reiterate that in some scenarios there might indeed be an Individual behind the client and in other cases the client may be autonomous.

1.3. Challenges for Contemporary Federation

As federated IdPs and RPs (services) proliferate, the role of an Individual can become ambiguous in certain circumstances. For example, a school might provide online access for a student’s grades to their parents for review and to the student’s teacher for modification. A teacher who is also a parent must clearly distinguish their role upon access.

Similarly, as federations proliferate, it becomes increasingly difficult to discover which IdP(s) a user is associated with. This is true for both the web and non-web case but is particularly acute for the latter, as many non-web authentication systems are not semantically rich enough on their own to allow for such ambiguities. For instance, in the case of an email provider, SMTP and IMAP do not have the ability for the server to request information from the client, beyond the client NAI, that the server would then use to decide between the multiple federations it is associated with. However, the building blocks do exist to add this functionality.

1.4. An Overview of ABFAB-Based Federation

The previous section described the general model of federation and the application of access management within the federation. This section provides a brief overview of ABFAB in the context of this model.

In this example, a client is attempting to connect to a server in order to either get access to some data or perform some type of transaction. In order for the client to mutually authenticate with the server, the following steps are taken in an ABFAB architecture (a graphical view of the steps can be found in Figure 2):

1. Client configuration: The client is configured with an NAI assigned by the IdP. It is also configured with any keys, certificates, passwords, or other secret and public information needed to run the EAP protocols between it and the IdP.

2. Authentication mechanism selection: The client is configured to use the GSS-EAP GSS-API mechanism for authentication/ authorization.

3. Client provides an NAI to RP: The client sets up a transport to the RP and begins GSS-EAP authentication. In response, the RP sends an EAP request message (nested in GSS-EAP) asking for the client's name. The client sends an EAP response with an NAI name form that, at a minimum, contains the realm portion of its full NAI.

4. Discovery of federated IdP: The RP uses preconfigured information or a federation proxy to determine what IdP to use, based on policy and the realm portion of the provided client NAI. This is discussed in detail below (Section 2.1.2).

5. Request from RP to IdP: Once the RP knows who the IdP is, it (or its agent) will send a RADIUS request to the IdP. The RADIUS Access-Request encapsulates the EAP response. At this stage, the RP will likely have no idea who the client is. The RP sends its identity to the IdP in AAA attributes, and it may send a SAML request in a AAA attribute. The AAA network checks to see that the identity claimed by the RP is valid.

6. IdP begins EAP with the client: The IdP sends an EAP message to the client with an EAP method to be used. The IdP should not re-request the client's name in this message, but clients need to be able to handle it. In this case, the IdP must accept a realm only in order to protect the client's name from the RP. The available and appropriate methods are discussed below (Section 2.2.1).

7. EAP is run: A bunch of EAP messages are passed between the client (EAP peer) and the IdP (EAP server), until the result of the authentication protocol is determined. The number and content of those messages depend on the EAP method selected. If the IdP is unable to authenticate the client, the IdP sends an

EAP failure message to the RP. As part of the EAP method, the client sends an EAP channel-binding message to the IdP (Section 2.2.2). In the channel-binding message, the client identifies, among other things, the RP to which it is attempting to authenticate. The IdP checks the channel-binding data from the client against the data provided by the RP via the AAA protocol. If the bindings do not match, the IdP sends an EAP failure message to the RP.

8. Successful EAP authentication: At this point, the IdP (EAP server) and client (EAP peer) have mutually authenticated each other. As a result, the client and the IdP hold two cryptographic keys: a Master Session Key (MSK) and an Extended MSK (EMSK). At this point, the client has a level of assurance regarding the identity of the RP, based on the name checking the IdP has done, using the RP naming information from the AAA framework and from the client (by the channel-binding data).

9. Local IdP policy check: At this stage, the IdP checks local policy to determine whether the RP and client are authorized for a given transaction/service and, if so, what attributes, if any, will be released to the RP. If the IdP gets a policy failure, it sends an EAP failure message to the RP and client. (The RP will have done its policy checks during the discovery process.)

10. IdP provides the RP with the MSK: The IdP sends a success result EAP to the RP, along with an optional set of AAA attributes associated with the client (usually as one or more SAML Assertions). In addition, the EAP MSK is returned to the RP.

11. RP processes results: When the RP receives the result from the IdP, it should have enough information to either grant or refuse a resource Access-Request. It may have information that associates the client with specific authorization identities. If additional attributes are needed from the IdP, the RP may make a new SAML request to the IdP. It will apply these results in an application-specific way.

12. RP returns results to client: Once the RP has a response, it must inform the client of the result. If all has gone well, all are authenticated, and the application proceeds with appropriate authorization levels. The client can now complete the authentication of the RP by using the EAP MSK value.

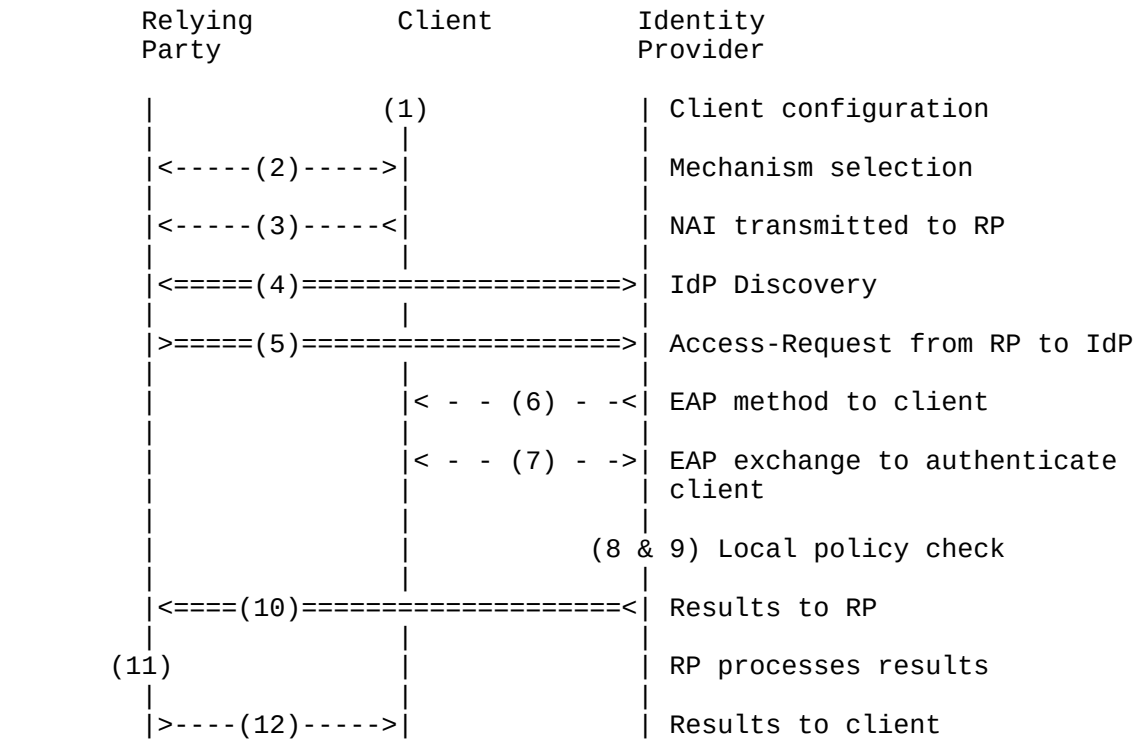


Figure 2: ABFAB Authentication Steps

1.5. Design Goals

Our key design goals are as follows:

- Each party in a transaction will be authenticated, although perhaps not identified, and the client will be authorized for access to a specific resource.
- The means of authentication is decoupled from the application protocol so as to allow for multiple authentication methods with minimal changes to the application.
- The architecture requires no sharing of long-term private keys between clients and RPs.

- The system will scale to large numbers of IdPs, RPs, and users.
- The system will be designed primarily for non-web-based authentication.
- The system will build upon existing standards, components, and operational practices.

Designing new three-party authentication and authorization protocols is difficult and fraught with the risk of cryptographic flaws. Achieving widespread deployment is even more difficult. A lot of attention on federated access has been devoted to the web. This document instead focuses on a non-web-based environment and focuses on those protocols where HTTP is not used. Despite the growing trend to layer every protocol on top of HTTP, there are still a number of protocols available that do not use HTTP-based transports. Many of these protocols are lacking a native authentication and authorization framework of the style shown in Figure 1.

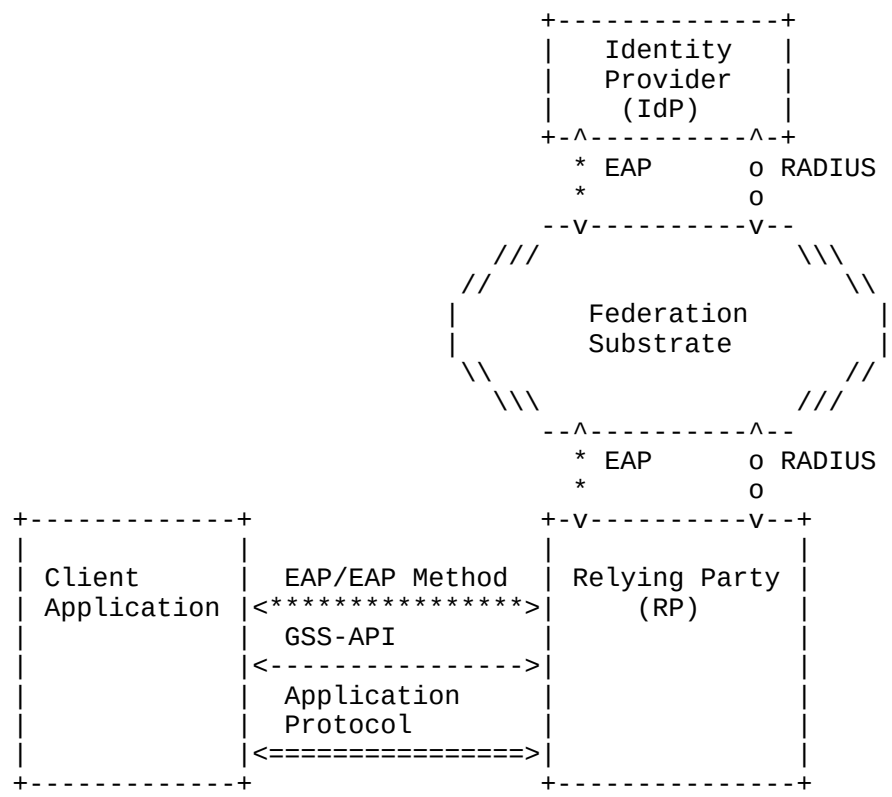
2. Architecture

We have already introduced the federated access architecture, with the illustration of the different actors that need to interact. This section expands on the specifics of providing support for non-web-based applications and provides motivations for design decisions. The main theme of the work described in this document is focused on reusing existing building blocks that have been deployed already and to rearrange them in a novel way.

Although this architecture assumes updates to the RP, the client, and the IdP, those changes are kept at a minimum. A mechanism that can demonstrate deployment benefits (based on ease of updates to existing software, low implementation effort, etc.) is preferred, and there may be a need to specify multiple mechanisms to support the range of different deployment scenarios.

There are a number of ways to encapsulate EAP into an application protocol. For ease of integration with a wide range of non-web-based application protocols, GSS-API was chosen. The technical specification of GSS-EAP can be found in [RFC7055].

The architecture consists of several building blocks, as shown graphically in Figure 3. In the following sections, we discuss the data flow between each of the entities, the protocols used for that data flow, and some of the trade-offs made in choosing the protocols.



Legend:

<****>: Client-to-IdP Exchange
<---->: Client-to-RP Exchange
<oooo>: RP-to-IdP Exchange
<====>: Protocol through which GSS-API/GS2 exchanges are tunneled

Figure 3: ABFAB Protocol Instantiation

2.1. Relying Party to Identity Provider

Communication between the RP and the IdP is done by the Federation Substrate. This communication channel is responsible for:

- Establishing the trust relationship between the RP and the IdP.
- Determining the rules governing the relationship.
- Conveying authentication packets from the client to the IdP and back.

- Providing the means of establishing a trust relationship between the RP and the client.

- Providing a means for the RP to obtain attributes about the client from the IdP.

The ABFAB working group has chosen the AAA framework for the messages transported between the RP and IdP. The AAA framework supports the requirements stated above, as follows:

- The AAA backbone supplies the trust relationship between the RP and the IdP.

- The agreements governing a specific AAA backbone contain the rules governing the relationships within the AAA federation.

- A method exists for carrying EAP packets within RADIUS [RFC3579] and Diameter [RFC4072].

- The use of EAP channel binding [RFC6677] along with the core ABFAB protocol provide the pieces necessary to establish the identities of the RP and the client, while EAP provides the cryptographic methods for the RP and the client to validate that they are talking to each other.

- A method exists for carrying SAML packets within RADIUS [RFC7833]; this method allows the RP to query attributes about the client from the IdP.

Protocols that support the same framework but do different routing are expected to be defined and used in the future. One such effort, called the Trust Router, is to set up a framework that creates a trusted point-to-point channel on the fly (https://www.ietf.org/proceedings/86/slides/slides-86-rtgarea-0.pdf).

2.1.1. AAA, RADIUS, and Diameter

The usage of the AAA framework with RADIUS [RFC2865] and Diameter [RFC6733] for network access authentication has been successful from a deployment point of view. To map the terminology used in Figure 1 to the AAA framework, the IdP corresponds to the AAA server; the RP corresponds to the AAA client; and the technical building blocks of a federation are AAA proxies, relays, and redirect agents (particularly if they are operated by third parties, such as AAA brokers and clearinghouses). In the case of network access authentication, the front end, i.e., the communication path between the end host and the AAA client, is offered by link-layer protocols that forward

authentication protocol exchanges back and forth. An example of a large-scale RADIUS-based federation is eduroam (https://www.eduroam.org).

By using the AAA framework, ABFAB can be built on the federation agreements that already exist; the agreements can then merely be expanded to cover the ABFAB architecture. The AAA framework has already addressed some of the problems outlined above. For example,

- It already has a method for routing requests based on a domain.
- It already has an extensible architecture allowing for new attributes to be defined and transported.
- Pre-existing relationships can be reused.

The astute reader will notice that RADIUS and Diameter have substantially similar characteristics. Why not pick one? RADIUS and Diameter are deployed in different environments. RADIUS can often be found in enterprise and university networks; RADIUS is also used by operators of fixed networks. Diameter, on the other hand, is deployed by operators of mobile networks. Another key difference is that today RADIUS is largely transported over UDP. The decision regarding which protocol will be appropriate to deploy is left to implementers. The protocol defines all the necessary new AAA attributes as RADIUS attributes. A future document could define the same AAA attributes for a Diameter environment. We also note that there exist proxies that convert from RADIUS to Diameter and back. This makes it possible for both to be deployed in a single Federation Substrate.

Through the integrity-protection mechanisms in the AAA framework, the IdP can establish technical trust that messages are being sent by the appropriate RP. Any given interaction will be associated with one federation at the policy level. The legal or business relationship defines what statements the IdP is trusted to make and how these statements are interpreted by the RP. The AAA framework also permits the RP or elements between the RP and IdP to make statements about the RP.

The AAA framework provides transport for attributes. Statements made about the client by the IdP, statements made about the RP, and other information are transported as attributes.

One demand that the AAA substrate makes of the upper layers is that they must properly identify the endpoints of the communication. It must be possible for the AAA client at the RP to determine where to send each RADIUS or Diameter message. Without this requirement, it

would be the RP's responsibility to determine the identity of the client on its own, without the assistance of an IdP. This architecture makes use of the Network Access Identifier (NAI), where the IdP is indicated by the realm component [RFC7542]. The NAI is represented and consumed by the GSS-API layer as GSS_C_NT_USER_NAME, as specified in [RFC2743]. The GSS-API EAP mechanism includes the NAI in the EAP Response/Identity message.

At the time of this writing, no profiles for the use of Diameter have been created.

2.1.2. Discovery and Rules Determination

While we are using the AAA protocols to communicate with the IdP, the RP may have multiple Federation Substrates to select from. The RP has a number of criteria that it will use in selecting which of the different federations to use. The federation selected must

- be able to communicate with the IdP.
- match the business rules and technical policies required for the RP security requirements.

The RP needs to discover which federation will be used to contact the IdP. The first selection criterion used during discovery is going to be the name of the IdP to be contacted. The second selection criterion used during discovery is going to be the set of business rules and technical policies governing the relationship; this is called "rules determination". The RP also needs to establish technical trust in the communications with the IdP.

Rules determination covers a broad range of decisions about the exchange. One of these is whether the given RP is permitted to talk to the IdP using a given federation at all, so rules determination encompasses the basic authorization decision. Other factors are included, such as what policies govern release of information about the client to the RP and what policies govern the RP's use of this information. While rules determination is ultimately a business function, it has a significant impact on the technical exchanges. The protocols need to communicate the result of authorization. When multiple sets of rules are possible, the protocol must disambiguate which set of rules are in play. Some rules have technical enforcement mechanisms; for example, in some federations, intermediaries validate information that is being communicated within the federation.

At the time of this writing, no protocol mechanism has been specified to allow a AAA client to determine whether a AAA proxy will indeed be able to route AAA requests to a specific IdP. The AAA routing is impacted by business rules and technical policies that may be quite complex; at the present time, the route selection is based on manual configuration.

2.1.3. Routing and Technical Trust

Several approaches to having messages routed through the Federation Substrate are possible. These routing methods can most easily be classified based on the mechanism for technical trust that is used. The choice of technical trust mechanism constrains how rules determination is implemented. Regardless of what deployment strategy is chosen, it is important that the technical trust mechanism be able to validate the identities of both parties to the exchange. The trust mechanism must ensure that the entity acting as the IdP for a given NAI is permitted to be the IdP for that realm and that any service name claimed by the RP is permitted to be claimed by that entity. Here are the categories of technical trust determination:

AAA Proxy:
: The simplest model is that an RP is a AAA client and can send the request directly to a AAA proxy. The hop-by-hop integrity protection of the AAA fabric provides technical trust. An RP can submit a request directly to the correct federation. Alternatively, a federation disambiguation fabric can be used. Such a fabric takes information about what federations the RP is part of and what federations the IdP is part of, and it routes a message to the appropriate federation. The routing of messages across the fabric, plus attributes added to requests and responses, together provide rules determination. For example, when a disambiguation fabric routes a message to a given federation, that federation’s rules are chosen. Name validation is enforced as messages travel across the fabric. The entities near the RP confirm its identity and validate names it claims. The fabric routes the message towards the appropriate IdP, validating the name of the IdP in the process. The routing can be statically configured. Alternatively, a routing protocol could be developed to exchange reachability information about a given IdP and to apply policy across the AAA fabric. Such a routing protocol could flood naming constraints to the appropriate points in the fabric.

Trust Broker:
Instead of routing messages through AAA proxies, some trust broker could establish keys between entities near the RP and entities near the IdP. The advantage of this approach is efficiency of message handling. Fewer entities are needed to be involved for each message. Security may be improved by sending individual messages over fewer hops. Rules determination involves decisions made by trust brokers about what keys to grant. Also, associated with each credential is context about rules and about other aspects of technical trust, including names that may be claimed. A routing protocol similar to the one for AAA proxies is likely to be useful to trust brokers in flooding rules and naming constraints.

Global Credential:
A global credential such as a public key and certificate in a public key infrastructure can be used to establish technical trust. A directory or distributed database such as the Domain Name System is used by the RP to discover the endpoint to contact for a given NAI. Either the database or certificates can provide a place to store information about rules determination and naming constraints. Provided that no intermediates are required (or appear to be required) and that the RP and IdP are sufficient to enforce and determine rules, rules determination is reasonably simple. However, applying certain rules is likely to be quite complex. For example, if multiple sets of rules are possible between an IdP and RP, confirming that the correct set is used may be difficult. This is particularly true if intermediates are involved in making the decision. Also, to the extent that directory information needs to be trusted, rules determination may be more complex.

Real-world deployments are likely to be mixtures of these basic approaches. For example, it will be quite common for an RP to route traffic to a AAA proxy within an organization. That proxy could then use any of the above three methods to get closer to the IdP. It is also likely that, rather than being directly reachable, the IdP may have a proxy on the edge of its organization. Federations will likely provide a traditional AAA proxy interface even if they also provide another mechanism for increased efficiency or security.

2.1.4. AAA Security

For the AAA framework, there are two different places where security needs to be examined. The first is the security that is in place for the links in the AAA backbone being used. The second are the nodes that form the AAA backbone.

The default link security for RADIUS is showing its age, as it uses MD5 and a shared secret to both obfuscate passwords and provide integrity on the RADIUS messages. While some EAP methods include the ability to protect the client authentication credentials, the MSK returned from the IdP to the RP is protected only by RADIUS security. In many environments, this is considered to be insufficient, especially as not all attributes are obfuscated and can thus leak information to a passive eavesdropper. The use of RADIUS with Transport Layer Security (TLS) [RFC6614] and/or Datagram Transport Layer Security (DTLS) [RFC7360] addresses these attacks. The same level of security is included in the base Diameter specifications.

2.1.5. SAML Assertions

For the traditional use of AAA frameworks, i.e., granting access to a network, an affirmative response from the IdP is sufficient. In the ABFAB world, the RP may need to get significantly more additional information about the client before granting access. ABFAB therefore has a requirement that it can transport an arbitrary set of attributes about the client from the IdP to the RP.

The Security Assertion Markup Language (SAML) [OASIS.saml-core-2.0-os] was designed in order to carry an extensible set of attributes about a subject. Since SAML is extensible in the attribute space, ABFAB has no immediate needs to update the core SAML specifications for our work. It will be necessary to update IdPs that need to return SAML Assertions to RPs and for both the IdP and the RP to implement a new SAML profile designed to carry SAML Assertions in AAA. The new profile can be found in [RFC7833]. As SAML statements will frequently be large, RADIUS servers and clients that deal with SAML statements will need to implement [RFC7499].

There are several issues that need to be highlighted:

- The security of SAML Assertions.
- Namespaces and mapping of SAML attributes.
- Subject naming of entities.
- Making multiple queries about the subject(s).
- Level of assurance for authentication.

SAML Assertions have an optional signature that can be used to protect and provide the origination of the assertion. These signatures are normally based on asymmetric key operations and require that the verifier be able to check not only the cryptographic

operation but also the binding of the originator's name and the public key. In a federated environment, it will not always be possible for the RP to validate the binding; for this reason, the technical trust established in the federation is used as an alternate method of validating the origination and integrity of the SAML Assertion.

Attributes in a SAML Assertion are identified by a name string. The name string is either assigned by the SAML issuer context or scoped by a namespace (for example, a URI or object identifier (OID)). This means that the same attribute can have different name strings used to identify it. In many cases, but not all, the federation agreements will determine what attributes and names can be used in a SAML statement. This means that the RP needs to map from the SAML issuer or federation name, type, and semantic to the name, type, and semantics that the policies of the RP are written in. In other cases, the Federation Substrate, in the form of proxies, will modify the SAML Assertions in transit to do the necessary name, type, and value mappings as the assertion crosses boundaries in the federation. If the proxies are modifying the SAML Assertion, then they will remove any signatures on the SAML Assertion, as changing the content of the SAML Assertion would invalidate the signature. In this case, the technical trust is the required mechanism for validating the integrity of the assertion. (The proxy could re-sign the SAML Assertion, but the same issues of establishing trust in the proxy would still exist.) Finally, the attributes may still be in the namespace of the originating IdP. When this occurs, the RP will need to get the required mapping operations from the federation agreements and do the appropriate mappings itself.

[RFC7833] has defined a new SAML name format that corresponds to the NAI name form defined by [RFC7542]. This allows for easy name matching in many cases, as the name form in the SAML statement and the name form used in RADIUS or Diameter will be the same. In addition to the NAI name form, [RFC7833] also defines a pair of implicit name forms corresponding to the client and the client's machine. These implicit name forms are based on the Identity-Type enumeration defined in the Tunnel Extensible Authentication Protocol (TEAP) specification [RFC7170]. If the name form returned in a SAML statement is not based on the NAI, then it is a requirement on the EAP server that it validate that the subject of the SAML Assertion, if any, is equivalent to the subject identified by the NAI used in the RADIUS or Diameter session.

RADIUS has the ability to deal with multiple SAML queries for those EAP servers that follow [RFC5080]. In this case, a State attribute will always be returned with the Access-Accept. The EAP client can then send a new Access-Request with the State attribute and the new

SAML request. Multiple SAML queries can then be done by making a new Access-Request, using the State attribute returned in the last Access-Accept to link together the different RADIUS sessions.

Some RPs need to ensure that specific criteria are met during the authentication process. This need is met by using levels of assurance. A level of assurance is communicated to the RP from the EAP server by using a SAML Authentication Request, using the Authentication Profile described in [RFC7833]. When crossing boundaries between different federations, (1) the policy specified will need to be shared between the two federations, (2) the policy will need to be mapped by the proxy server on the boundary, or (3) the proxy server on the boundary will need to supply information to the EAP server so that the EAP server can do the required mapping. If this mapping is not done, then the EAP server will not be able to enforce the desired level of assurance, as it will not understand the policy requirements.

2.2. Client to Identity Provider

Looking at the communications between the client and the IdP, the following items need to be dealt with:

- The client and the IdP need to mutually authenticate each other.
- The client and the IdP need to mutually agree on the identity of the RP.

ABFAB selected EAP for the purposes of mutual authentication and assisted in creating some new EAP channel-binding documents for dealing with determining the identity of the RP. A framework for the channel-binding mechanism has been defined in [RFC6677] that allows the IdP to check the identity of the RP provided by the AAA framework against the identity provided by the client.

2.2.1. Extensible Authentication Protocol (EAP)

Traditional web federation does not describe how a client interacts with an IdP for authentication. As a result, this communication is not standardized. There are several disadvantages to this approach. Since the communication is not standardized, it is difficult for machines to recognize which entity is going to do the authentication, and thus which credentials to use and where in the authentication form the credentials are to be entered. It is much easier for humans to correctly deal with these problems. The use of browsers for authentication restricts the deployment of more secure forms of authentication beyond plaintext usernames and passwords known by the server. In a number of cases, the authentication interface may be

presented before the client has adequately validated that they are talking to the intended server. By giving control of the authentication interface to a potential attacker, the security of the system may be reduced, and opportunities for phishing may be introduced.

As a result, it is desirable to choose some standardized approach for communication between the client's end host and the IdP. There are a number of requirements this approach must meet, as noted below.

Experience has taught us one key security and scalability requirement: it is important that the RP not get possession of the long-term secret of the client. Aside from a valuable secret being exposed, a synchronization problem can develop when the client changes keys with the IdP.

Since there is no single authentication mechanism that will be used everywhere, another associated requirement is that the authentication framework must allow for the flexible integration of authentication mechanisms. For instance, some IdPs require hardware tokens, while others use passwords. A service provider wants to provide support for both authentication methods and also for other methods from IdPs not yet seen.

These requirements can be met by utilizing standardized and successfully deployed technology, namely the EAP framework [RFC3748]. Figure 3 illustrates the integration graphically.

EAP is an end-to-end framework; it provides for two-way communication between a peer (i.e., client or Individual) through the EAP authenticator (i.e., RP) to the back end (i.e., IdP). This is precisely -- and conveniently -- the communication path that is needed for federated identity. Although EAP support is already integrated in AAA systems (see [RFC3579] and [RFC4072]), several challenges remain:

- The first is how to carry EAP payloads from the end host to the RP.
- Another is to verify statements the RP has made to the client, confirm that these statements are consistent with statements made to the IdP, and confirm that all of the above are consistent with the federation and any federation-specific policy or configuration.
- Another challenge is choosing which IdP to use for which service.

The EAP method used for ABFAB needs to meet the following requirements:

- It needs to provide mutual authentication of the client and IdP.
- It needs to support channel binding.

As of this writing, the only EAP method that meets these criteria is TEAP [RFC7170], either alone (if client certificates are used) or with an inner EAP method that does mutual authentication.

2.2.2. EAP Channel Binding

EAP channel binding is easily confused with a facility in GSS-API that is also called "channel binding". GSS-API channel binding provides protection against man-in-the-middle attacks when GSS-API is used for authentication inside of some tunnel; it is similar to a facility called "cryptographic binding" in EAP. See [RFC5056] for a discussion of the differences between these two facilities.

The client knows, in theory, the name of the RP that it attempted to connect to; however, in the event that an attacker has intercepted the protocol, the client and the IdP need to be able to detect this situation. A general overview of the problem, along with a recommended way to deal with the channel-binding issues, can be found in [RFC6677].

Since the time that [RFC6677] was published, a number of possible attacks were found. Methods to address these attacks have been outlined in [RFC7029].

2.3. Client to Relying Party

The final set of interactions between the parties to consider are those between the client and the RP. In some ways, this is the most complex set, since at least part of it is outside the scope of the ABFAB work. The interactions between these parties include:

- Running the protocol that implements the service that is provided by the RP and desired by the client.
- Authenticating the client to the RP and the RP to the client.
- Providing the necessary security services to the service protocol that it needs, beyond authentication.
- Dealing with client re-authentication where desired.

2.3.1. GSS-API

One of the remaining layers is responsible for integration of federated authentication with the application. Applications have adopted a number of approaches for providing security, so multiple strategies for integration of federated authentication with applications may be needed. To this end, we start with a strategy that provides integration with a large number of application protocols.

Many applications, such as Secure Shell (SSH) [RFC4462], NFS [RFC7530], DNS [RFC3645], and several non-IETF applications, support GSS-API [RFC2743]. Many applications, such as IMAP, SMTP, the Extensible Messaging and Presence Protocol (XMPP), and the Lightweight Directory Access Protocol (LDAP), support the Simple Authentication and Security Layer (SASL) [RFC4422] framework. These two approaches work together nicely: by creating a GSS-API mechanism, SASL integration is also addressed. In effect, using a GSS-API mechanism with SASL simply requires placing some headers before the mechanism's messages and constraining certain GSS-API options.

GSS-API is specified in terms of an abstract set of operations that can be mapped into a programming language to form an API. When people are first introduced to GSS-API, they focus on it as an API. However, from the perspective of authentication for non-web applications, GSS-API should be thought of as a protocol as well as an API. When looked at as a protocol, it consists of abstract operations such as the initial context exchange, which includes two sub-operations (GSS_Init_sec_context and GSS_Accept_sec_context) [RFC2743]. An application defines which abstract operations it is going to use and where messages produced by these operations fit into the application architecture. A GSS-API mechanism will define what actual protocol messages result from that abstract message for a given abstract operation. So, since this work is focusing on a particular GSS-API mechanism, we generally focus on protocol elements rather than the API view of GSS-API.

The API view of GSS-API does have significant value as well; since the abstract operations are well defined, the information that a mechanism gets from the application is well defined. Also, the set of assumptions the application is permitted to make is generally well defined. As a result, an application protocol that supports GSS-API or SASL is very likely to be usable with a new approach to authentication, including the authentication mechanism defined in this document, with no required modifications. In some cases, support for a new authentication mechanism has been added using plugin interfaces to applications without the application being modified at all. Even when modifications are required, they can

often be limited to supporting a new naming and authorization model. For example, this work focuses on privacy; an application that assumes that it will always obtain an identifier for the client will need to be modified to support anonymity, unlinkability, or pseudonymity.

So, we use GSS-API and SASL because a number of the application protocols we wish to federate support these strategies for security integration. What does this mean from a protocol standpoint, and how does this relate to other layers? This means that we need to design a concrete GSS-API mechanism. We have chosen to use a GSS-API mechanism that encapsulates EAP authentication. So, GSS-API (and SASL) encapsulates EAP between the end host and the service. The AAA framework encapsulates EAP between the RP and the IdP. The GSS-API mechanism includes rules about how initiators and services are named as well as per-message security and other facilities required by the applications we wish to support.

2.3.2. Protocol Transport

The transport of data between the client and the RP is not provided by GSS-API. GSS-API creates and consumes messages, but it does not provide the transport itself; instead, the protocol using GSS-API needs to provide the transport. In many cases, HTTP or HTTPS is used for this transport, but other transports are perfectly acceptable. The core GSS-API document [RFC2743] provides some details on what requirements exist.

In addition, we highlight the following:

- The transport does not need to provide either confidentiality or integrity. After GSS-EAP has finished negotiation, GSS-API can be used to provide both services. If the negotiation process itself needs protection from eavesdroppers, then the transport would need to provide the necessary services.

- The transport needs to provide reliable transport of the messages.

- The transport needs to ensure that tokens are delivered in order during the negotiation process.

- GSS-API messages need to be delivered atomically. If the transport breaks up a message, it must also reassemble the message before delivery.

2.3.3. Re-authentication

There are circumstances where the RP will want to have the client re-authenticate itself. These include very long sessions, where the original authentication is time limited or cases where in order to complete an operation a different authentication is required. GSS-EAP does not have any mechanism for the server to initiate a re-authentication, as all authentication operations start from the client. If a protocol using GSS-EAP needs to support re-authentication that is initiated by the server, then a request from the server to the client for the re-authentication to start needs to be placed in the protocol.

Clients can reuse the existing secure connection established by GSS-API, and run the new authentication in that connection, by calling GSS_Init_sec_context. At this point, a full re-authentication will be done.

3. Application Security Services

One of the key goals is to integrate federated authentication with existing application protocols and, where possible, existing implementations of these protocols. Another goal is to perform this integration while meeting the best security practices of the technologies used to perform the integration. This section describes security services and properties required by the EAP GSS-API mechanism in order to meet these goals. This information could be viewed as specific to that mechanism. However, other future application integration strategies are very likely to need similar services. So, it is likely that these services will be expanded across application integration strategies if new application integration strategies are adopted.

3.1. Authentication

GSS-API provides an optional security service called "mutual authentication". This service means that in addition to the initiator providing (potentially anonymous or pseudonymous) identity to the acceptor, the acceptor confirms its identity to the initiator. In the context of ABFAB in particular, the naming of this service is confusing. We still say that mutual authentication is provided when the identity of an acceptor is strongly authenticated to an anonymous initiator.

Unfortunately, [RFC2743] does not explicitly talk about what mutual authentication means. Within this document, we therefore define mutual authentication as follows:

- If a target name is configured for the initiator, then the initiator trusts that the supplied target name describes the acceptor. This implies that (1) appropriate cryptographic exchanges took place for the initiator to make such a trust decision and (2) after evaluating the results of these exchanges, the initiator's policy trusts that the target name is accurate.

- If no target name is configured for the initiator, then the initiator trusts that the acceptor name, supplied by the acceptor, correctly names the entity it is communicating with.

- Both the initiator and acceptor have the same key material for per-message keys, and both parties have confirmed that they actually have the key material. In EAP terms, there is a protected indication of success.

Mutual authentication is an important defense against certain aspects of phishing. Intuitively, clients would like to assume that if some party asks for their credentials as part of authentication, successfully gaining access to the resource means that they are talking to the expected party. Without mutual authentication, the server could "grant access" regardless of what credentials are supplied. Mutual authentication better matches this user intuition.

It is important, therefore, that the GSS-EAP mechanism implement mutual authentication. That is, an initiator needs to be able to request mutual authentication. When mutual authentication is requested, only EAP methods capable of providing the necessary service can be used, and appropriate steps need to be taken to provide mutual authentication. While a broader set of EAP methods could be supported by not requiring mutual authentication, it was decided that the client needs to always have the ability to request it. In some cases, the IdP and the RP will not support mutual authentication; however, the client will always be able to detect this and make an appropriate security decision.

The AAA infrastructure may hide the initiator's identity from the GSS-API acceptor, providing anonymity between the initiator and the acceptor. At this time, whether the identity is disclosed is determined by EAP server policy rather than by an indication from the initiator. Also, initiators are unlikely to be able to determine whether anonymous communication will be provided. For this reason, initiators are unlikely to set the anonymous return flag from GSS_Init_sec_context (Section 2.2.1 of [RFC2743]).

3.2. GSS-API Channel Binding

[RFC5056] defines a concept of channel binding that is used to prevent man-in-the-middle attacks. This type of channel binding works by taking a cryptographic value from the transport security layer and checks to see that both sides of the GSS-API conversation know this value. Transport Layer Security (TLS) [RFC5246] is the most common transport security layer used for this purpose.

It needs to be stressed that channel binding as described in [RFC5056] (also called "GSS-API channel binding" when GSS-API is involved) is not the same thing as EAP channel binding. GSS-API channel binding is used for detecting man-in-the-middle attacks. EAP channel binding is used for mutual authentication and acceptor naming checks. See [RFC7055] for details. A more detailed description of the differences between the facilities can be found in [RFC5056].

The use of TLS can provide both encryption and integrity on the channel. It is common to provide SASL and GSS-API with these other security services.

One of the benefits that the use of TLS provides is that a client has the ability to validate the name of the server. However, this validation is predicated on a couple of things. The TLS session needs to be using certificates and not be an anonymous session. The client and the TLS server need to share a common trust point for the certificate used in validating the server. TLS provides its own server authentication. However, there are a variety of situations where, for policy or usability reasons, this authentication is not checked. When the TLS authentication is checked, if the trust infrastructure behind the TLS authentication is different from the trust infrastructure behind the GSS-API mutual authentication, then confirming the endpoints using both trust infrastructures is likely to enhance security. If the endpoints of the GSS-API authentication are different than the endpoints of the lower layer, this is a strong indication of a problem, such as a man-in-the-middle attack. Channel binding provides a facility to determine whether these endpoints are the same.

The GSS-EAP mechanism needs to support channel binding. When an application provides channel-binding data, the mechanism needs to confirm that this is the same on both sides, consistent with the GSS-API specification.

3.3. Host-Based Service Names

IETF security mechanisms typically take a host name and perhaps a service, entered by a user, and make some trust decision about whether the remote party in the interaction is the intended party. This decision can be made via the use of certificates, preconfigured key information, or a previous leap of trust. GSS-API has defined a relatively flexible naming convention; however, most of the IETF applications that use GSS-API (including SSH, NFS, IMAP, LDAP, and XMPP) have chosen to use a more restricted naming convention based on the host name. The GSS-EAP mechanism needs to support host-based service names in order to work with existing IETF protocols.

The use of host-based service names leads to a challenging trust delegation problem. Who is allowed to decide whether a particular host name maps to a specific entity? Possible solutions to this problem have been looked at.

- The Public Key Infrastructure (PKI) used by the web has chosen to have a number of trust anchors (root certificate authorities), each of which can map any host name to a public key.

- A number of GSS-API mechanisms, such as Kerberos [RFC1964], have split the problem into two parts. [RFC1964] introduced a new concept called a realm; the realm is responsible for host mapping within itself. The mechanism then decides what realm is responsible for a given name. This is the approach adopted by ABFAB.

GSS-EAP defines a host naming convention that takes into account the host name, the realm, the service, and the service parameters. An example of a GSS-API service name is "xmpp/foo@example.com". This identifies the XMPP service on the host foo in the realm example.com. Any of the components, except for the service name, may be omitted from a name. When omitted, a local default would be used for that component of the name.

While there is no requirement that realm names map to Fully Qualified Domain Names (FQDNs) within DNS, in practice this is normally true. Doing so allows the realm portion of service names and the portion of NAIs to be the same. It also allows for the use of DNS in locating the host of a service while establishing the transport channel between the client and the RP.

It is the responsibility of the application to determine the server that it is going to communicate with; GSS-API has the ability to help confirm that the server is the desired server but not to determine the name of the server to use. It is also the responsibility of the

application to determine how much of the information identifying the service needs to be validated by the ABFAB system. The information that needs to be validated is used to construct the service name passed into the GSS-EAP mechanism. What information is to be validated will depend on (1) what information was provided by the client and (2) what information is considered significant. If the client only cares about getting a specific service, then it does not need to validate the host and realm that provides the service.

Applications may retrieve information about providers of services from DNS. Service Records (SRVs) [RFC2782] and Naming Authority Pointer (NAPTR) [RFC3401] records are used to help find a host that provides a service; however, the necessity of having DNSSEC on the queries depends on how the information is going to be used. If the host name returned is not going to be validated by EAP channel binding because only the service is being validated, then DNSSEC [RFC4033] is not required. However, if the host name is going to be validated by EAP channel binding, then DNSSEC needs to be used to ensure that the correct host name is validated. In general, if the information that is returned from the DNS query is to be validated, then it needs to be obtained in a secure manner.

Another issue that needs to be addressed for host-based service names is that they do not work ideally when different instances of a service are running on different ports. If the services are equivalent, then it does not matter. However, if there are substantial differences in the quality of the service, that information needs to be part of the validation process. If one has just a host name and not a port in the information being validated, then this is not going to be a successful strategy.

3.4. Additional GSS-API Services

GSS-API provides per-message security services that can provide confidentiality and/or integrity. Some IETF protocols, such as NFS and SSH, take advantage of these services. As a result, GSS-EAP needs to support these services. As with mutual authentication, per-message security services will limit the set of EAP methods that can be used to those that generate a Master Session Key (MSK). Any EAP method that produces an MSK is able to support per-message security services as described in [RFC2743].

GSS-API provides a pseudorandom function. This function generates a pseudorandom sequence using the shared session key as the seed for the bytes generated. This provides an algorithm that both the initiator and acceptor can run in order to arrive at the same key value. The use of this feature allows an application to generate keys or other shared secrets for use in other places in the protocol.

In this regard, it is similar in concept to the mechanism (formerly known as "TLS Extractors") described in [RFC5705]. While no current IETF protocols require this feature, non-IETF protocols are expected to take advantage of it in the near future. Additionally, a number of protocols have found the mechanism described in [RFC5705] to be useful in this regard, so it is highly probable that IETF protocols may also start using this feature.

4. Privacy Considerations

As an architecture designed to enable federated authentication and allow for the secure transmission of identity information between entities, ABFAB obviously requires careful consideration regarding privacy and the potential for privacy violations.

This section examines the privacy-related information presented in this document, summarizing the entities that are involved in ABFAB communications and what exposure they have to identity information. In discussing these privacy considerations in this section, we use terminology and ideas from [RFC6973].

Note that the ABFAB architecture uses at its core several existing technologies and protocols; detailed privacy discussion regarding these topics is not examined. This section instead focuses on privacy considerations specifically related to the overall architecture and usage of ABFAB.

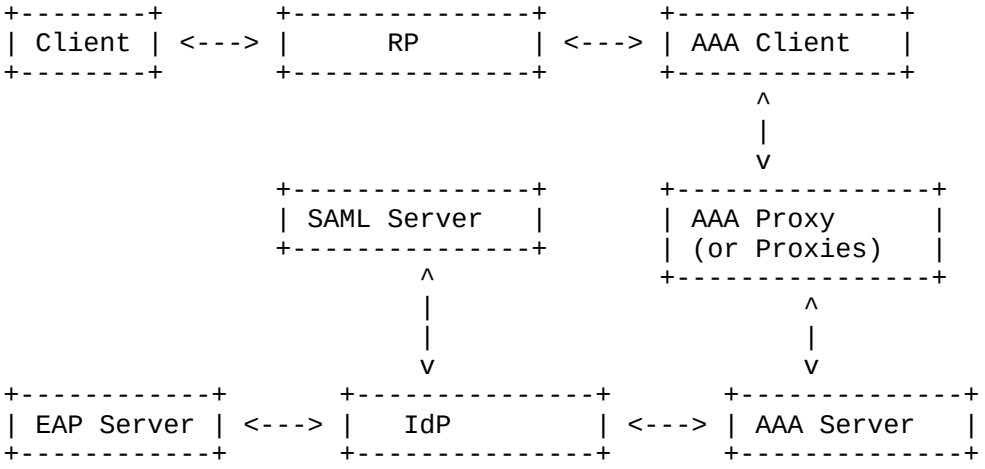


Figure 4: Entities and Data Flow

4.1. Entities and Their Roles

Categorizing the ABFAB entities shown in Figure 4 according to the taxonomy of terms from [RFC6973] is somewhat complicated, as the roles of each entity will change during the various phases of ABFAB communications. The three main phases of relevance are the client-to-RP communication phase, the client-to-IdP (via the Federation Substrate) communication phase, and the IdP-to-RP (via the Federation Substrate) communication phase.

In the client-to-RP communication phase, we have:

Initiator: Client.

Observers: Client, RP.

Recipient: RP.

In the client-to-IdP (via the Federation Substrate) communication phase, we have:

Initiator: Client.

Observers: Client, RP, AAA Client, AAA Proxy (or Proxies), AAA Server, IdP.

Recipient: IdP

In the IdP-to-RP (via the Federation Substrate) communication phase, we have:

Initiator: RP.

Observers: IdP, AAA Server, AAA Proxy (or Proxies), AAA Client, RP.

Recipient: IdP

Eavesdroppers and attackers can reside on any or all communication links between the entities shown in Figure 4.

The various entities in the system might also collude or be coerced into colluding. Some of the significant collusions to look at are as follows:

- If two RPs are colluding, they have the information available to both nodes. This can be analyzed as if a single RP were offering multiple services.

- If an RP and a AAA proxy are colluding, then the trust of the system is broken, as the RP would be able to lie about its own identity to the IdP. There is no known way to deal with this situation.

- If multiple AAA proxies are colluding, they can be treated as a single node for analysis.

The Federation Substrate consists of all of the AAA entities. In some cases, the AAA proxies may not exist, as the AAA client can talk directly to the AAA server. Specifications such as the Trust Router Protocol (https://www.ietf.org/proceedings/86/slides/slides-86-rtgarea-0.pdf) and RADIUS dynamic discovery [RFC7585] can be used to shorten the path between the AAA client and the AAA server (and thus stop these AAA proxies from being observers); however, even in these circumstances, there may be AAA proxies in the path.

In Figure 4, the IdP has been divided into multiple logical pieces; in actual implementations, these pieces will frequently be tightly coupled. The links between these pieces provide the greatest opportunity for attackers and eavesdroppers to acquire information; however, as they are all under the control of a single entity, they are also the easiest to have tightly secured.

4.2. Privacy Aspects of ABFAB Communication Flows

In the ABFAB architecture, there are a few different types of data and identifiers in use. The best way to understand them, and their potential privacy impacts, is to look at each phase of communication in ABFAB.

4.2.1. Client to RP

The flow of data between the client and the RP is divided into two parts. The first part consists of all of the data exchanged as part of the ABFAB authentication process. The second part consists of all of the data exchanged after the authentication process has been finished.

During the initial communication phase, the client sends an NAI (see [RFC7542]) to the RP. Many EAP methods (but not all) allow the client to disclose an NAI to the RP in a form that includes only a realm component during this communication phase. This is the minimum amount of identity information necessary for ABFAB to work -- it indicates an IdP that the principal has a relationship with. EAP methods that do not allow this will necessarily also reveal an identifier for the principal in the IdP realm (e.g., a username).

The data shared during the initial communication phase may be protected by a channel protocol such as TLS. This will prevent the leakage of information to passive eavesdroppers; however, an active attacker may still be able to set itself up as a man-in-the-middle. The client may not be able to validate the certificates (if any) provided by the service, deferring the check of the identity of the RP until the completion of the ABFAB authentication protocol (using EAP channel binding rather than certificates).

The data exchanged after the authentication process can have privacy and authentication using the GSS-API services. If the overall application protocol allows for the process of re-authentication, then the same privacy implications as those discussed in previous paragraphs apply.

4.2.2. Client to IdP (via Federation Substrate)

This phase includes a secure TLS tunnel set up between the client and the IdP via the RP and Federation Substrate. The process is initiated by the RP using the realm information given to it by the client. Once set up, the tunnel is used to send credentials to the IdP to authenticate.

Various operational information is transported between the RP and the IdP over the AAA infrastructure -- for example, using RADIUS headers. As no end-to-end security is provided by AAA, all AAA entities on the path between the RP and IdP have the ability to eavesdrop on this information. Some of this information may form identifiers or explicit identity information:

- The RP knows the IP address of the client. It is possible that the RP could choose to expose this IP address by including it in a RADIUS header (e.g., using the Calling-Station-Id). This is a privacy consideration to take into account for the application protocol.

- The EAP MSK is transported between the IdP and the RP over the AAA infrastructure -- for example, through RADIUS headers. This is a particularly important privacy consideration, as any AAA proxy

that has access to the EAP MSK is able to decrypt and eavesdrop on any traffic encrypted using that EAP MSK (i.e., all communications between the client and RP). This problem can be mitigated if the application protocol sets up a secure tunnel between the client and the RP and performs a cryptographic binding between the tunnel and EAP MSK.

- o Related to the bullet point above, the AAA server has access to the material necessary to derive the session key; thus, the AAA server can observe any traffic encrypted between the client and RP. This "feature" was chosen as a simplification and to make performance faster; if it was decided that this trade-off was not desirable for privacy and security reasons, then extensions to ABFAB that make use of techniques such as Diffie-Hellman key exchange would mitigate this.

The choice of EAP method used has other potential privacy implications. For example, if the EAP method in use does not support mutual authentication, then there are no guarantees that the IdP is who it claims to be, and thus the full NAI, including a username and a realm, might be sent to any entity masquerading as a particular IdP.

Note that ABFAB has not specified any AAA accounting requirements. Implementations that use the accounting portion of AAA should consider privacy appropriately when designing this aspect.

4.2.3. IdP to RP (via Federation Substrate)

In this phase, the IdP communicates with the RP, informing it as to the success or failure of authentication of the user and, optionally, the sending of identity information about the principal.

As in the previous flow (client to IdP), various operation information is transported between the IdP and RP over the AAA infrastructure, and the same privacy considerations apply. However, in this flow, explicit identity information about the authenticated principal can be sent from the IdP to the RP. This information can be sent through RADIUS headers, or using SAML [RFC7833]. This can include protocol-specific identifiers, such as SAML NameIDs, as well as arbitrary attribute information about the principal. What information will be released is controlled by policy on the IdP. As before, when sending this information through RADIUS headers, all AAA entities on the path between the RP and IdP have the ability to eavesdrop, unless additional security measures are taken (such as the use of TLS for RADIUS [RFC6614]). However, when sending this

information using SAML as specified in [RFC7833], confidentiality of the information should be guaranteed, as [RFC7833] requires the use of TLS for RADIUS.

4.3. Relationship between User and Entities

- Between user and IdP - The IdP is an entity the user will have a direct relationship with, created when the organization that operates the entity provisioned and exchanged the user's credentials. Privacy and data protection guarantees may form a part of this relationship.

- Between user and RP - The RP is an entity the user may or may not have a direct relationship with, depending on the service in question. Some services may only be offered to those users where such a direct relationship exists (for particularly sensitive services, for example), while some may not require this and would instead be satisfied with basic federation trust guarantees between themselves and the IdP. This may well include the option that the user stays anonymous with respect to the RP (though, obviously, never anonymous to the IdP). If attempting to preserve privacy via data minimization (Section 1), then the only attribute information about Individuals exposed to the RP should be attribute information that is strictly necessary for the operation of the service.

- Between user and Federation Substrate - The user is highly likely to have no knowledge of, or relationship with, any entities involved with the Federation Substrate (not that the IdP and/or RP may, however). Knowledge of attribute information about Individuals for these entities is not necessary, and thus such information should be protected in such a way as to prevent the possibility of access to this information.

4.4. Accounting Information

Alongside the core authentication and authorization that occur in AAA communications, accounting information about resource consumption may be delivered as part of the accounting exchange during the lifetime of the granted application session.

4.5. Collection and Retention of Data and Identifiers

In cases where RPs are not required to identify a particular Individual when an Individual wishes to make use of their service, the ABFAB architecture enables anonymous or pseudonymous access. Thus, data and identifiers other than pseudonyms and unlinkable attribute information need not be stored and retained.

However, in cases where RPs require the ability to identify a particular Individual (e.g., so they can link this identity information to a particular account in their service, or where identity information is required for audit purposes), the service will need to collect and store such information, and to retain it for as long as they require. The de-provisioning of such accounts and information is out of scope for ABFAB, but for privacy protection, it is obvious that any identifiers collected should be deleted when they are no longer needed.

4.6. User Participation

In the ABFAB architecture, by its very nature users are active participants in the sharing of their identifiers, as they initiate the communications exchange every time they wish to access a server. They are, however, not involved in the control of information related to them that is transmitted from the IdP to the RP for authorization purposes; rather, this is under the control of policy on the IdP. Due to the nature of the AAA communication flows, with the current ABFAB architecture there is no place for a process of gaining user consent for the information to be released from the IdP to the RP.

5. Security Considerations

This document describes the architecture for Application Bridging for Federated Access Beyond web (ABFAB), and security is therefore the main focus. Many of the items that are security considerations have already been discussed in Section 4 ("Privacy Considerations"). Readers should be sure to read that section as well.

There are many places in this document where TLS is used. While in some places (e.g., client to RP) anonymous connections can be used, it is very important that TLS connections within the AAA infrastructure and between the client and the IdP be fully authenticated and, if using certificates, that revocation be checked as well. When using anonymous connections between the client and the RP, all messages and data exchanged between those two entities will be visible to an active attacker. In situations where the client is not yet on the network, the status_request extension [RFC6066] can be used to obtain revocation-checking data inside of the TLS protocol. Clients also need to get the trust anchor for the IdP configured correctly in order to prevent attacks; this is a difficult problem in general and is going to be even more difficult for kiosk environments.

Selection of the EAP methods to be permitted by clients and IdPs is important. The use of a tunneling method such as TEAP [RFC7170] allows other EAP methods to be used while hiding the contents of

those EAP exchanges from the RP and the AAA framework. When considering inner EAP methods, the considerations outlined in [RFC7029] about binding the inner and outer EAP methods need to be taken into account. Finally, one wants to have the ability to support channel binding in those cases where the client needs to validate that it is talking to the correct RP.

In those places where SAML statements are used, RPs will generally be unable to validate signatures on the SAML statement, either because the signature has been stripped off by the IdP or because the RP is unable to validate the binding between the signer, the key used to sign, and the realm represented by the IdP. For these reasons, it is required that IdPs do the necessary trust checking on the SAML statements and that RPs can trust the AAA infrastructure to keep the SAML statements valid.

When a pseudonym is generated as a unique long-term identifier for a client by an IdP, care must be taken in the algorithm that it cannot easily be reverse-engineered by the service provider. If it can be reverse-engineered, then the service provider can consult an oracle to determine if a given unique long-term identifier is associated with a different known identifier.

6. References

6.1. Normative References

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, DOI 10.17487/RFC2743, January 2000, <http://www.rfc-editor.org/info/rfc2743>.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <http://www.rfc-editor.org/info/rfc2865>.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, DOI 10.17487/RFC3579, September 2003, <http://www.rfc-editor.org/info/rfc3579>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <http://www.rfc-editor.org/info/rfc3748>.

[RFC4072] Eronen, P., Ed., Hiller, T., and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, DOI 10.17487/RFC4072, August 2005, <http://www.rfc-editor.org/info/rfc4072>.

[RFC6677] Hartman, S., Ed., Clancy, T., and K. Hoeper, "Channel-Binding Support for Extensible Authentication Protocol (EAP) Methods", RFC 6677, DOI 10.17487/RFC6677, July 2012, <http://www.rfc-editor.org/info/rfc6677>.

[RFC7055] Hartman, S., Ed., and J. Howlett, "A GSS-API Mechanism for the Extensible Authentication Protocol", RFC 7055, DOI 10.17487/RFC7055, December 2013, <http://www.rfc-editor.org/info/rfc7055>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <http://www.rfc-editor.org/info/rfc7542>.

[RFC7833] Howlett, J., Hartman, S., and A. Perez-Mendez, Ed., "A RADIUS Attribute, Binding, Profiles, Name Identifier Format, and Confirmation Methods for the Security Assertion Markup Language (SAML)", RFC 7833, DOI 10.17487/RFC7833, May 2016, <http://www.rfc-editor.org/info/rfc7833>.

6.2. Informative References

[NIST-SP.800-63-2] Burr, W., Dodson, D., Newton, E., Perlner, R., Polk, W., Gupta, S., and E. Nabbus, "Electronic Authentication Guideline", NIST Special Publication 800-63-2, August 2013, <http://dx.doi.org/10.6028/NIST.SP.800-63-2>.

[OASIS.saml-core-2.0-os] Cantor, S., Kemp, J., Philpott, R., and E. Maler, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-core-2.0-os, March 2005, <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>.

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, DOI 10.17487/RFC1964, June 1996, <http://www.rfc-editor.org/info/rfc1964>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

[RFC3401] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS", RFC 3401, DOI 10.17487/RFC3401, October 2002, <http://www.rfc-editor.org/info/rfc3401>.

[RFC3645] Kwan, S., Garg, P., Gilroy, J., Esibov, L., Westhead, J., and R. Hall, "Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG)", RFC 3645, DOI 10.17487/RFC3645, October 2003, <http://www.rfc-editor.org/info/rfc3645>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <http://www.rfc-editor.org/info/rfc4033>.

[RFC4422] Melnikov, A., Ed., and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, DOI 10.17487/RFC4422, June 2006, <http://www.rfc-editor.org/info/rfc4422>.

[RFC4462] Hutzelman, J., Salowey, J., Galbraith, J., and V. Welch, "Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol", RFC 4462, DOI 10.17487/RFC4462, May 2006, <http://www.rfc-editor.org/info/rfc4462>.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, DOI 10.17487/RFC5056, November 2007, <http://www.rfc-editor.org/info/rfc5056>.

[RFC5080] Nelson, D. and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", RFC 5080, DOI 10.17487/RFC5080, December 2007, <http://www.rfc-editor.org/info/rfc5080>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <http://www.rfc-editor.org/info/rfc5705>.

[RFC5801] Josefsson, S. and N. Williams, "Using Generic Security Service Application Program Interface (GSS-API) Mechanisms in Simple Authentication and Security Layer (SASL): The GS2 Mechanism Family", RFC 5801, DOI 10.17487/RFC5801, July 2010, <http://www.rfc-editor.org/info/rfc5801>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <http://www.rfc-editor.org/info/rfc6066>.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, DOI 10.17487/RFC6614, May 2012, <http://www.rfc-editor.org/info/rfc6614>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <http://www.rfc-editor.org/info/rfc6733>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <http://www.rfc-editor.org/info/rfc6749>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <http://www.rfc-editor.org/info/rfc6973>.

[RFC7029] Hartman, S., Wasserman, M., and D. Zhang, "Extensible Authentication Protocol (EAP) Mutual Cryptographic Binding", RFC 7029, DOI 10.17487/RFC7029, October 2013, <http://www.rfc-editor.org/info/rfc7029>.

[RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <http://www.rfc-editor.org/info/rfc7170>.

[RFC7360] DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, DOI 10.17487/RFC7360, September 2014, <http://www.rfc-editor.org/info/rfc7360>.

[RFC7499] Perez-Mendez, A., Ed., Marin-Lopez, R., Pereniguez-Garcia, F., Lopez-Millan, G., Lopez, D., and A. DeKok, "Support of Fragmentation of RADIUS Packets", RFC 7499, DOI 10.17487/RFC7499, April 2015, <http://www.rfc-editor.org/info/rfc7499>.

[RFC7530] Haynes, T., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", RFC 7530, DOI 10.17487/RFC7530, March 2015, <http://www.rfc-editor.org/info/rfc7530>.

[RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October 2015, <http://www.rfc-editor.org/info/rfc7585>.

[WS-TRUST] Lawrence, K., Kaler, C., Nadalin, A., Goodner, M., Gudgin, M., Turner, D., Barbir, A., and H. Granqvist, "WS-Trust 1.4", OASIS Standard ws-trust-2012-04, April 2012, <http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html>.

Acknowledgments

We would like to thank Mayutan Arumaithurai, Klaas Wierenga, and Rhys Smith for their feedback. Additionally, we would like to thank Eve Maler, Nicolas Williams, Bob Morgan, Scott Cantor, Jim Fenton, Paul Leach, and Luke Howard for their feedback on the federation terminology question.

Furthermore, we would like to thank Klaas Wierenga for his review of the first draft version of this document. We also thank Eliot Lear for his work on early draft versions of this document.

Authors' Addresses

Josh Howlett
Jisc
Lumen House, Library Avenue, Harwell
Oxford OX11 0SG
United Kingdom

Phone: +44 1235 822363
Email: Josh.Howlett@ja.net

Sam Hartman
Painless Security

Email: hartmans-ietf@mit.edu

Hannes Tschofenig
ARM Ltd.
110 Fulbourn Road
Cambridge CB1 9NJ
United Kingdom

Email: Hannes.tschofenig@gmx.net
URI: http://www.tschofenig.priv.at

Jim Schaad
August Cellars

Email: ietf@augustcellars.com