Paper SAS623-2017

Kerberos Cross-Realm Authentication: Unraveling the Mysteries

Stuart J. Rogers, SAS Institute Inc., Cary, NC

ABSTRACT

How do you enable strong authentication across different parts of your organization in a safe and secure way? We know that Kerberos provides us with a safe and secure strong authentication mechanism, but how does it work across different domains or realms? In this paper, we examine how Kerberos cross-realm authentication works and the different parts that you need ready in order to use Kerberos effectively. Understanding the principles and applying the ideas we present will make you successful at improving the security of your authentication system.

INTRODUCTION

Kerberos security is being implemented by a growing number of customers to enable strong authentication across different parts of an organization in a safe and secure way. SAS® is designed to work within this secure environment. I hope that this paper will help you to unravel the mysteries of Kerberos for SAS administrators, and enables you to work more closely with your IT and Security counterparts when SAS is deployed in an environment that uses Kerberos for basic authentication, and in the more complex case of cross-realm authentication.

Kerberos authentication provides us with a strong mechanism to authenticate users across a network, enabling us to have confidence in the authentication of end-users to our networked applications. However, as organizations grow in size and complexity, so does the network across which we need to authenticate users. In this paper we will examine how Kerberos authentication operates, the different components used to authenticate the users, and we will take a step-by-step view of how the Kerberos protocol messages enable strong authentication (see the section titled “Kerberos 101”).

Once we have detailed the basics of Kerberos authentication, we will extend this to look at what happens when the network infrastructure become more complex. We will examine what is meant by a Kerberos realm and how we can authenticate our end-users across different Kerberos realms. This will allow us to detail the requirements we need for Kerberos cross-realm authentication to correctly work (see the section titled “Kerberos Cross-Realm”).

Finally, we will look at how to configure a SAS deployment to leverage Kerberos authentication in multi-realm customer environments. We will split the SAS environment into the different tiers and look at each in turn. This paper highlights how Kerberos cross-realm authentication interoperates with the SAS Middle Tier, SAS Server Tier, and third-party data sources. SAS administrators can use these guidelines when configuring a SAS deployment. See the section titled “Cross-Realm Requirements” for an overview, and the section titled “Cross-Realm with SAS” for specific guidelines related to a SAS deployment.

KERBEROS 101

In this first section, we detail how basic Kerberos authentication operates. In most cases, Microsoft® Active Directory can be seen as just an implementation of the Kerberos protocol. However, in some cases Active Directory and other implementations, such as MIT Kerberos, do have some differences. Throughout the paper I will attempt to highlight any such differences.

INTRODUCTION TO KERBEROS COMPONENTS

Before we examine how Kerberos cross-realm authentication operates, we examine how basic Kerberos authentication operates. Version 5 of the Kerberos Authentication protocol is defined in RFC4120 (IETF 2005). Kerberos has three parties taking actions in the authentication process. The first party is the client where the end user is authenticated. The second party is the service the end user authenticates to. The third party is the trusted third party which establishes the authentication of the end-user and server process. The trusted third party is the Kerberos Key Distribution Center (KDC). These three parties are

shown in Figure 1. Within the Kerberos KDC, two processes run. These are the Authentication Service (AS) and Ticket-Granting Services (TGS). At different points in the authentication process the client will communicate with each of these services.

Authentication with Kerberos has two distinct phases. The two phases can occur one directly after the other, or the two phases can be separated by a period of time. In the first phase the end user authenticates to the Kerberos KDC. In the second phase the end user authenticates to a service running on a server using Kerberos.

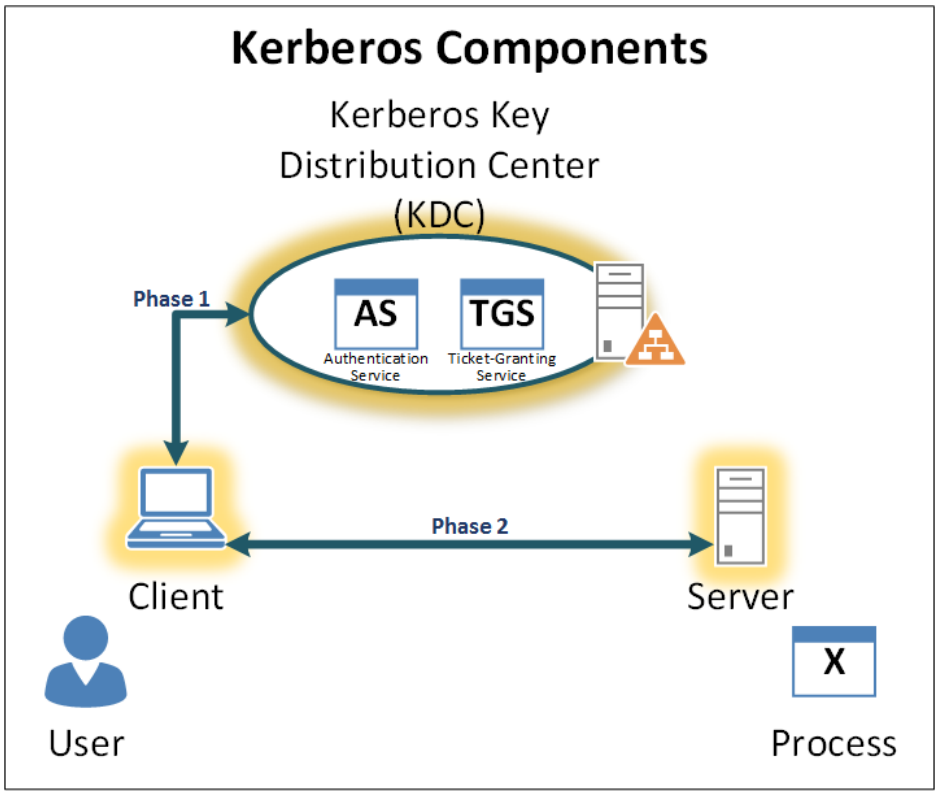

Figure 1. Kerberos Components

KERBEROS KEYS

Kerberos authentication relies upon the encryption of information using symmetric-key encryption. If one party is able to encrypt the information, then that party must have access to the key. Equally, for another party to be able to decrypt the information they must also have access to the same key. This enables both parties to prove they know the key without needing to send the key across the network. Kerberos leverages the following two types of keys:

- Long-term keys, which are normally valid for multiple days.
- Short-term session keys that are only valid for the length of a session.

The long-term keys are associated with the passwords of various objects. Here are the three long-term keys:

- A long-term key associated with the TGS.
- A long-term key for the service that end-users will authenticate to.
- A long-term key for the end user.

The key is created by taking the password, adding a salt, and then hashing the resulting string. Using different hashing algorithms results in different strength keys. When using Kerberos with Active Directory, each domain controller has a built-in user account with the name **krbtgt** and the password for this account is associated with the TGS Long-Term Key. A Kerberized service in an Active Directory domain will be attached to either a user object or computer object. The password of either the user or computer object is associated with the Service Long-Term key. Finally, the User Long-Term Key is associated with the password of the end user in Active Directory.

MIT Kerberos does not have the same concept of user and computer objects. MIT Kerberos just has principals; either user principals or service principals. With MIT Kerberos, the Long-Term Keys are associated with the passwords of different principals. Figure 2 illustrates the long-term keys used in Kerberos authentication.

Figure 2. Long-Term Keys

Short lived session keys are associated either with the TGS or the authenticated session the end user has with the service. These are shown in Figure 3. The short-term session keys are automatically generated by either the Kerberos AS or Kerberos TGS.

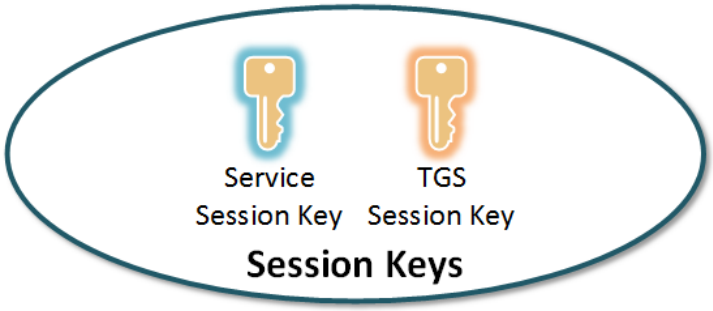

Figure 3. Short-Term Keys

PHASES OF AUTHENTICATION

As stated above, Kerberos authentication takes place in two distinct phases. Phase 1 is “initial authentication” and is where the end user authenticates to the Kerberos KDC. This process validates the Users Long-Term Key and issues the client software with the Ticket-Granting Ticket (TGT) and

associated TGS Session Key. These two Ticket-Granting objects are required for the second phase of Kerberos authentication.

Phase 2 is “user authentication to service” and is where the end user authenticates to the service they need to connect to. This phase uses the TGT and TGS Session Key to request a Service Ticket, and associated Service Session Key from the Kerberos KDC. The Service Ticket and associated Service Session Key are then used by the client software to authenticate the end user to the service.

The following two sections of the paper will examine each phase in more detail.

KERBEROS AUTHENTICATION: PHASE 1

The first phase of Kerberos authentication is the authentication of the end user to the Kerberos KDC. This phase involves communication solely between the client and the Kerberos KDC. This is depicted in Figure 4, with the following steps:

1. The end user provides the long-term key to the client, normally by entering their password.
2. The client sends a clear text message to the Kerberos KDC containing the user ID to the AS. This message is defined as the KRB_AS_REQ in the Kerberos specification (IETF 2005).
3. Optionally, according to the Kerberos specification, the AS responds with an error that pre-authentication is required. This message is defined as the KRB_AS_REP in the Kerberos specification (IETF 2005). This tells the client that it must use the User Long-Term Key to encrypt some information and send it to the AS.
4. The client encrypts a current timestamp with the User Long-Term Key and sends this to the AS, as shown as (A) in Figure 4. This is another KRB_AS_REQ.

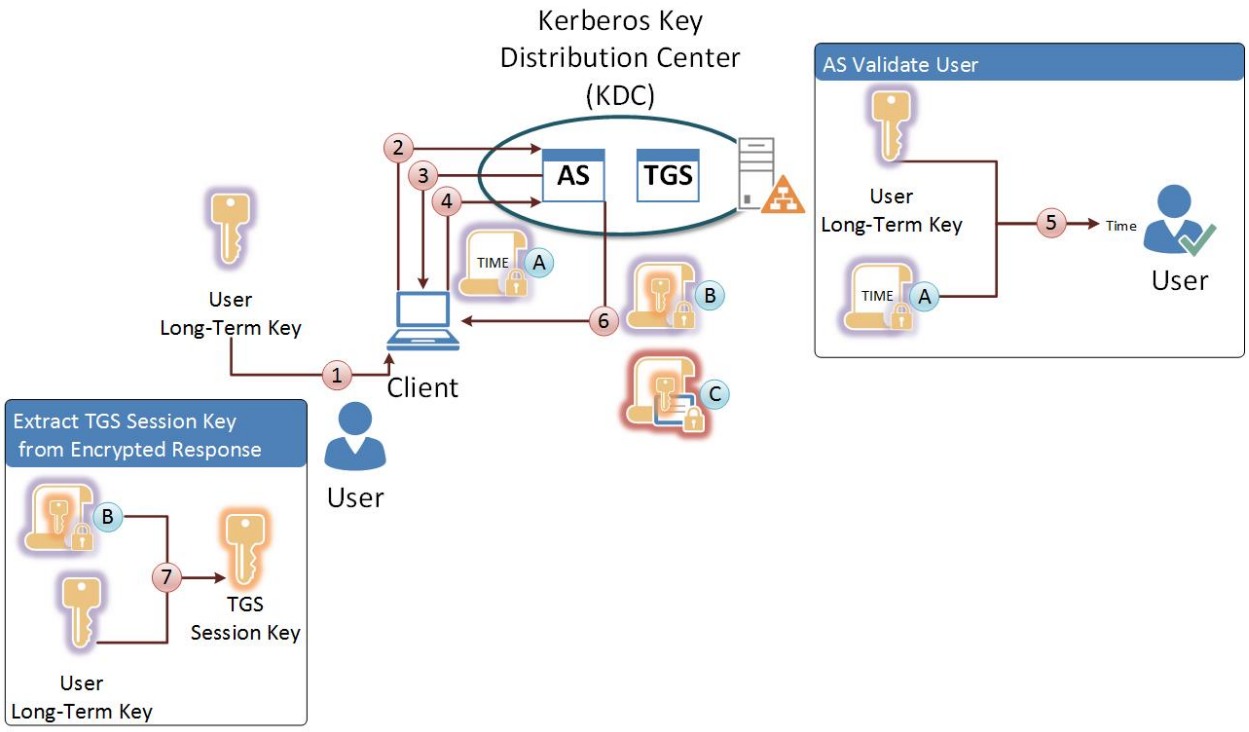

Figure 4. Kerberos Authentication: Phase 1

5. The Kerberos KDC uses the User Long-Term Key to decrypt the information sent in the

KRB_AS_REQ. If this is successful, the end user is validated by the Kerberos KDC.

6. The AS constructs a new KRB_AS_REP containing two objects, shown as (B) and (C) in Figure 4. Where (B) is the TGS Session Key encrypted using the User Long-Term Key and (C) is the TGT, which includes the client ID, client network address, ticket validity period, and TGS Session Key all encrypted using the TGS Long-Term Key. The KRB_AS_REP is sent to the client.
7. The client uses the User Long-Term Key to decrypt the first part of the KRB_AS_REP and obtains the TGS Session Key.

As a result of the first phase of Kerberos authentication, the client now has the following items:

- The User Long-Term Key
- The TGS Session Key
- The TGT

The TGT contains the TGS Session Key that the client already has, but since this is encrypted with the Long-Term Key for the TGS the client is unable to decrypt the content of the TGT. The purpose of including the TGS Session Key in the TGT is to enable any Kerberos KDC running within the domain/realm to process the second phase of Kerberos authentication. The multiple Kerberos KDCs do not need to maintain a list of all the TGS Session Keys since this will be sent as part of the second phase.

KERBEROS AUTHENTICATION: PHASE 2

The second phase of Kerberos authentication is the authentication of the end user to the service. This phase of authentication can either take place immediately after phase 1 or at some time after the end user has successfully authenticated to the Kerberos KDC. In the second phase the client still communicates initially with the Kerberos KDC, but also the client directly communicates with the service. Figure 5 illustrates the process of authenticating to the service. This covers the following steps:

1. The client creates and sends the KRB_TGS_REQ message. The client requests a ticket for a given Service Principal Name (SPN). Included in the request is an authenticator shown as (A) in Figure 5; this may include the current timestamp and is encrypted using the TGS Session Key. Also included in the KRB_TGS_REQ is the TGT shown as (B) in Figure 5.
2. The TGS decrypts the TGT shown as (B) using the TGS Long-Term Key and extracts the TGS Session Key.
3. The TGS decrypts the authenticator (A) using the TGS Session Key to validate the user.
4. The TGS constructs the KRB_TGS_REP message containing the Service Session Key encrypted using the TGS Session Key, shown as (C) in Figure 5, and the Service Ticket (ST) shown as (D). The Service Ticket includes the client ID, client network address, ticket validity period, and Service Session Key all encrypted using the Service Long-Term Key.
5. The client receives the KRB_TGS_REP and decrypts (C) using the TGS Session Key and thus obtains the Service Session Key. Since the Service Ticket (D) is encrypted using the Service Long-Term Key, the client is not able to decrypt this.
6. The client constructs the KRB_AP_REQ message, this is quite often bundled inside another protocol such as Simple Protected Negotiate Protocol (SPNEGO) for web authentication. Within the KRB_AP_REQ message are an authenticator, shown as (E) in Figure 5, which is encrypted with the Service Session Key obtained in the previous step. Also included in the KRB_AP_REQ is the Service Ticket obtained from the Kerberos KDC.
7. When the service receives the KRB_AP_REQ it decrypts the Service Ticket, shown as (D) in Figure 5, using the Service Long-Term Key and obtains the Service Session Key.
8. The Service Session Key can then be used to decrypt the authenticator, shown as (E) in Figure 5, to validate the user.
9. Optionally, the service can send a response defined as the KRB_AP_REP message. This contains

an authenticator encrypted with the Service Session Key, shown as (F) in Figure 5.

10. If a KRB_AP_REP message is sent, the client can decrypt the authenticator, shown as (F) in Figure 5, to validate that the service was able to correctly obtain the Service Session Key from the Service Ticket.

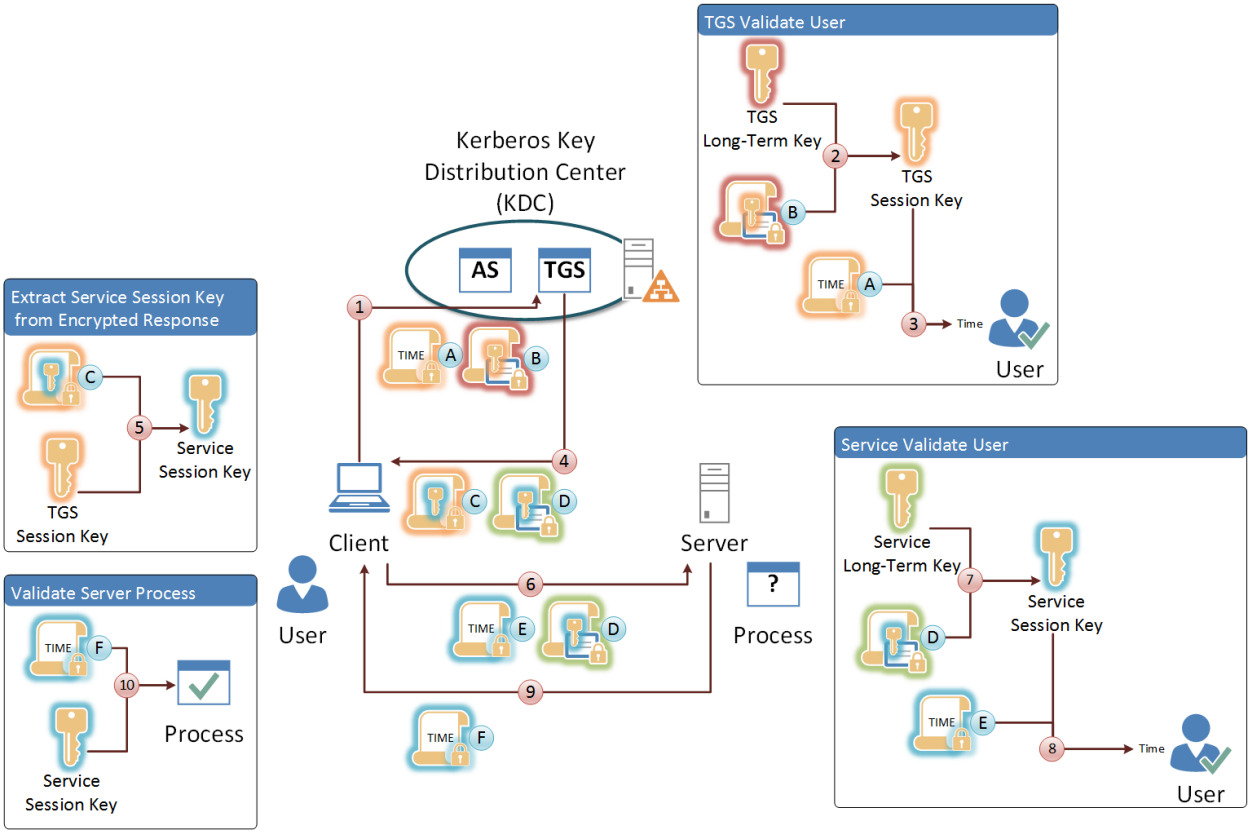

Figure 5. Kerberos Authentication: Phase 2

As a result of the second phase of Kerberos authentication, the client is now authenticated to the service and has the following items:

- The User Long-Term Key
- The TGS Session Key
- The TGT
- The Service Session Key
- The Service Ticket

The Service Ticket, like the TGT, is encrypted with a key not accessible to the client. Therefore, the client is not able to alter the Service Ticket at all. The Service Ticket can only have been constructed by the TGS running on the Kerberos KDC since only this has access to the Service Long-Term Key. This enables the service to be sure what issued the Service Ticket. Also, since the Service Ticket contains the Service Session Key there is no need for the service to directly communicate with the Kerberos KDC to authenticate the end user. The only communication to the Kerberos KDC is from the client.

CONSIDERATIONS FOR BASIC KERBEROS AUTHENTICATION

In order for basic Kerberos authentication to operate, several pieces need to be correctly configured. First, the service must be identifiable within the Kerberos KDC. The service is uniquely identified by its Service Principal Name. Without the Service Principal Name registered in the Kerberos KDC the Service Ticket cannot be generated. Active Directory allows Service Principal Names to be registered against either user objects or computer objects. MIT Kerberos does not have the same distinction and principal names are merely registered.

Equally important is the network connectivity. As illustrated in Figure 4 and Figure 5, the client must be able to connect to the Kerberos KDC and the machine hosting the service, which means that the client and the Kerberos KDC must be on interconnected networks. While the service being requested does not have to share a network connection with the Kerberos KDC, it only needs to share a connection with the client.

The client stores a number of key items to eliminate the need to complete both phases of Kerberos authentication each and every time the end user accesses a service. The client stores the following in a cache:

- The TGS Session Key
- The TGT
- The Service Session Key
- The Service Ticket

The type of cache used by different clients is dependent on the client. For example, Microsoft Windows uses a memory-based cache and also controls access to the TGS Session Key. By default, only processes using the Microsoft Windows API are allowed access to both the Ticket Cache and TGS Session Key. Most Linux systems use a file-based Ticket Cache, most often stored in the /tmp directory. Since this directory could in fact reside in-memory, the file-based Ticket Cache could still be stored in system memory. This file-based Ticket Cache uses standard Linux file permissions to control access. With newer versions of RedHat Enterprise Linux, version 7.x, the default location of the Ticket Cache moves to a Kernel protected memory location referred to as a keyring. The use of a keyring-based Ticket Cache makes RedHat Enterprise Linux behave more like Microsoft Windows.

KERBEROS CROSS-REALM

Now that we have an understanding of the basic process of Kerberos authentication, we can move on to examine what happens when there are multiple Kerberos Realms involved in the authentication process.

WHAT IS A REALM

First we need to understand what a Kerberos Realm is. A Kerberos Realm is a logical container of both multiple Kerberos KDCs and multiple Kerberos principals. The Kerberos principals are stored in a database; for Active Directory this is the main Active Directory database, while for MIT Kerberos this is a separate database. The database of principals is then shared across multiple Kerberos KDCs so that each can respond to requests. Running multiple Kerberos KDCs, each operating against the same database of principals, eliminates a possible single point of failure.

A single Kerberos Realm will need to be either an implementation of Active Directory or an implementation of MIT Kerberos. Active Directory and MIT Kerberos KDCs cannot be mixed in a single Kerberos Realm.

The name of the Kerberos Realm is any ASCII string. The convention is to make the Realm name the same as your network domain name, but using upper-case letters. For example, hosts in the network domain example.com would be in the Kerberos Realm called EXAMPLE.COM. If you need multiple Kerberos realms, MIT recommends that you use descriptive names that end with your network domain name, such as BOSTON.EXAMPLE.COM or HOUSTON.EXAMPLE.COM. Equally, the convention with Active Directory domains is that the Kerberos Realm is the upper-case version of the Active Directory

domain. So if the Active Directory domain is na.example.com the Kerberos Realm would be NA.EXAMPLE.COM.

Active Directory automatically creates a Kerberos Realm as the Active Directory domain is created. For MIT Kerberos, a Kerberos Realm is created as part of the process of initializing the MIT Kerberos KDC.

WHY MULTIPLE REALMS

Now we understand what a Kerberos Realm is we can think about why we might need to have multiple Kerberos Realms. There are no specific rules for how your organization should logically divide objects into different Kerberos Realms or Active Directory domains. The main purpose behind such separation is to provide separation between users and computers in different units. This separation could be due to security concerns, administrative concerns, or based on the sheer number of objects to manage.

For environments using Active Directory, you will have a Kerberos Realm corresponding to each Active Directory domain. Within your organization you will probably have different sub-domains based around the geography of your organization. This could be based at the country, district, or city level depending on how large your organization is, as shown in Figure 6. All of the sub-domain "trees" are then amalgamated into a single Active Directory "forest". Equally, your organization might split customer facing parts of the organization from the back-end business processing by placing these into separate Active Directory sub-domains. So you might have a sales domain and factory domain.

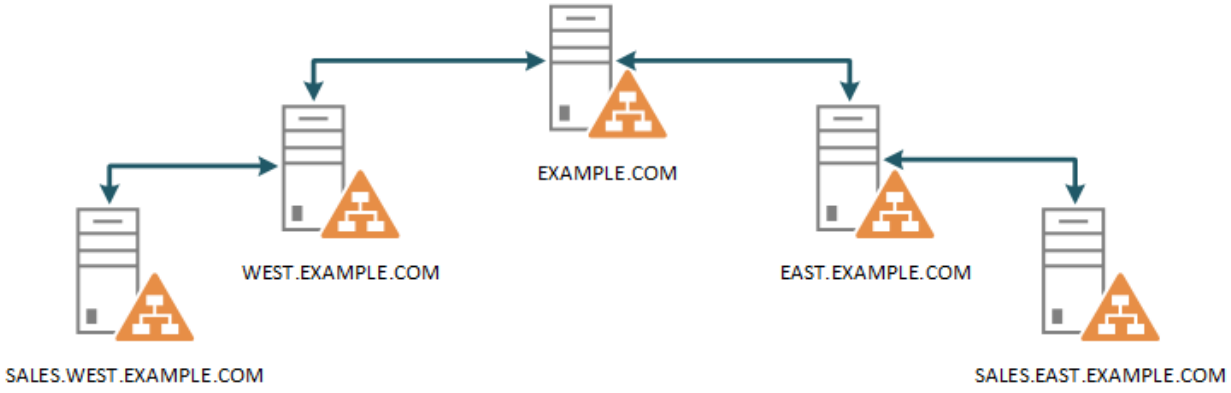

Figure 6. Example of Multiple Realms

CROSS-REALM AUTHENTICATION

The first phase of Kerberos authentication remains the same even when multiple Realms are involved. Differences are only apparent during the second phase of Kerberos authentication when the client requests a Service Ticket for a service contained in a different Kerberos Realm. For example, a user in SALES.WEST.EXAMPLE.COM connecting to a service in WEST.EXAMPLE.COM. The end result of the Cross-Realm authentication will be the same as discussed above, the user is authenticated to a service. The main difference in Cross-Realm authentication is the requirement for the client to send multiple requests to obtain a Service Ticket to each of the Kerberos Realms. Figure 7 illustrates the process of Cross-Realm authentication, covering the following steps:

1. The client creates and sends the KRB_TGS_REQ message. The client requests a ticket for a given SPN. Included in the request is an authenticator shown as (A) in Figure 7; this may include the current timestamp and is encrypted using the TGS Session Key. Also included in the KRB_TGS_REQ is the TGT shown as (B) in Figure 7.
2. The TGS decrypts the TGT shown as (B) using the TGS Long-Term Key and extracts the TGS Session Key.

3. The TGS decrypts the authenticator (A) using the TGS Session Key to validate the user.
4. The TGS examines the SPN for the KRB_TGS_REQ and notes that the SPN is for a different Kerberos Realm. The TGS constructs a KRB_TGS_REP containing two items. First, a Ticket-Granting Session Key for REALM 2 is encrypted using the existing TGS Session Key, shown as (C) in Figure 7. Second, a TGT for REALM 2, which includes the client ID, client network address, ticket validity period, and TGS Session Key are all encrypted using the shared Cross-Realm TGS Long-Term Key, shown as (D) in Figure 7.
5. The client receives the KRB_TGS_REP message and uses the current TGS Session Key to decrypt (C) and extract the Cross-Realm TGS Session Key.

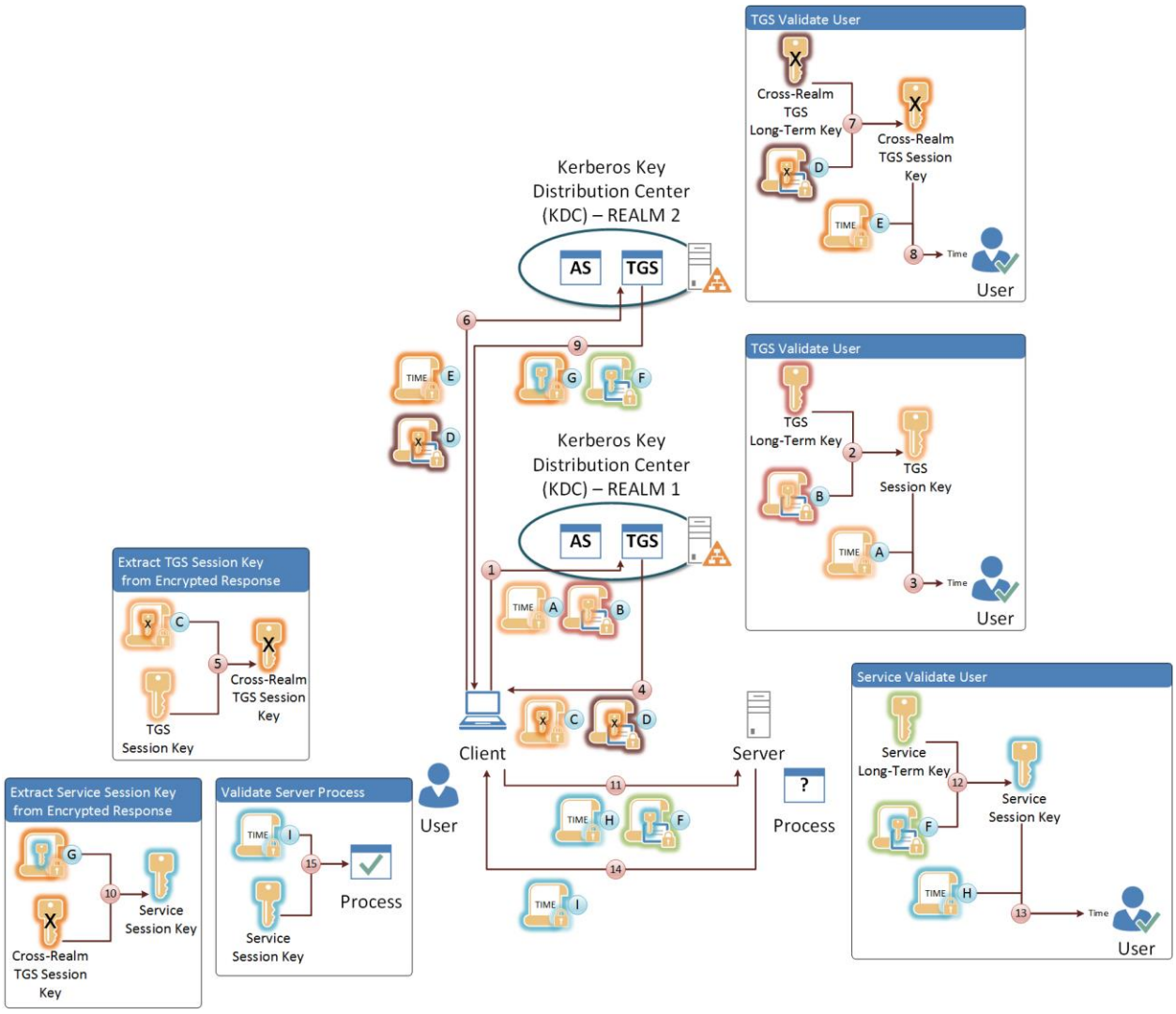

Figure 7. Kerberos Authentication Cross-Realm

6. The client sends a second TGS Request (KRB_TGS_REQ), this time to the KDC of REALM 2. Client requests a ticket for a given SPN. Included in the request is an authenticator, shown as (E) in Figure 7 and the Cross-Realm TGT (D). The authenticator (E) is this time encrypted using the Cross-Realm TGS Session Key.
7. The TGS for REALM 2 decrypts the TGT (D) using the Cross-Realm TGS Long-Term Key and

extracts the Cross-Realm TGS Session Key.

8. The TGS of REALM 2 then uses the Cross-Realm TGS Session Key to decrypt the authenticator (E) that was sent as part of the KRB_TGS_REQ and validates the end user. This end user does not need to be represented in the database of REALM 2 with a user principal since the act of decrypting the authenticator is sufficient to validate the end user.
9. The TGS of REALM 2 constructs the reply message (KRB_TGS_REP) containing two items. The first, shown as (F) in Figure 7, is the ST, which includes the client ID, client network address, ticket validity period, and Service Session Key all encrypted using the Service Long-Term Key. The second item, shown as (G) in Figure 7, is the Service Session Key encrypted using the Cross-Realm TGS Session Key.
10. The client receives the KRB_TGS_REP and uses the Cross-Realm TGS Session Key to decrypt (G) and extract the Service Session Key.
11. The client constructs the KRB_AP_REQ message. This is quite often bundled inside another protocol such as SPNEGO for web authentication. Within the KRB_AP_REQ message are an authenticator, shown as (H) in Figure 7, which is encrypted with the Service Session Key obtained in the previous step. Also included in the KRB_AP_REQ is the Service Ticket obtained from the Kerberos KDC of REALM 2.
12. When the service receives the KRB_AP_REQ, it decrypts the Service Ticket, shown as (F) in Figure 7, using the Service Long-Term Key and obtains the Service Session Key.
13. The Service Session Key can then be used to decrypt the authenticator, shown as (H) in Figure 7, to validate the user.
14. Optionally, the service can send a response defined as the KRB_AP_REP message. This contains an authenticator encrypted with the Service Session Key, shown as (I) in Figure 7.
15. If a KRB_AP_REP message is sent, the client can decrypt the authenticator, shown as (I) in Figure 7, to validate that the service was able to correctly obtain the Service Session Key from the Service Ticket.

As a result of the Kerberos Cross-Realm authentication, the client is now authenticated to the service and has the following items:

- The User Long-Term Key
- The TGS Session Key for REALM 1
- The Cross-Realm TGS Session Key for REALM 2
- The TGT
- The Cross-Realm TGT for REALM 2
- The Service Session Key
- The Service Ticket

Similar to the standard second phase of Kerberos authentication, with Cross-Realm Kerberos authentication, only the client communicates with the two Kerberos KDCs. There is no direct communication between the service and either of the KDCs. The client needs to be able to communicate with both KDCs and must be aware of where to send the second KRB_TGS_REQ in order to correctly request a Service Ticket.

In the example we have shown here, there are only two Kerberos Realms. However, the Kerberos specification does not limit the number of Realms it is possible to configure. As more Kerberos Realms are added, they can either have direct connections or a hierarchical structure. With a direct connection, the authentication follows the process we have already considered. For a hierarchical structure, the KRB_TGS_REQ and KRB_TGS_REP messages are passed through the structure until the KDC containing the SPN is reached.

The concept of working through a hierarchical structure can be difficult to understand and is important for both troubleshooting failures in authentication and performance concerns in the authentication process. Figure 8 illustrates a relatively simple structure with only two branches. We show the root Kerberos Realm for EXAMPLE.COM and then two branches for WEST.EXAMPLE.COM and EAST.EXAMPLE.COM. Below both EAST and WEST we have a further separation of SALES for each Realm. In our fictional organization we might then have other divisions alongside the SALES Realm.

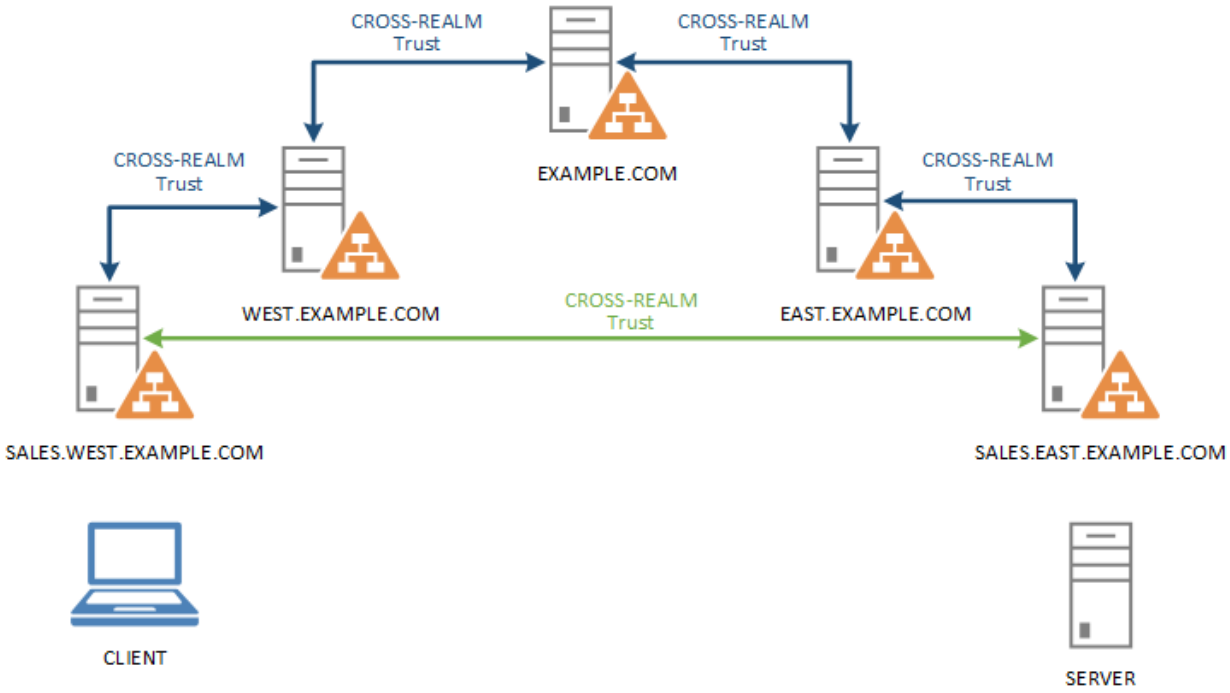

Figure 8. Cross-Realm Hierarchy

For the first case, we have a cross-realm trust in-place between each Realm and its parent, as shown by the blue lines in Figure 8. When a client in the SALES.WEST.EXAMPLE.COM attempts to connect to a resource on a server in SALES.EAST.EXAMPLE.COM, the path to retrieve a Service Ticket is:

1. SALES.WEST.EXAMPLE.COM
2. WEST.EXAMPLE.COM
3. EXAMPLE.COM
4. EAST.EXAMPLE.COM
5. SALES.EAST.EXAMPLE.COM

This means the client will need to send five KRB_TGS_REQ messages before it is able to obtain the Service Ticket from the Kerberos KDC for SALES.EAST.EXAMPLE.COM.

If the client, in SALES.WEST.EXAMPLE.COM, will need to regularly make a connection to the server in SALES.EAST.EXAMPLE.COM, then an option to simplify authentication is to create an additional cross-realm trust directly between the two SALES Realms. This is shown as the green line in Figure 8. With this additional cross-realm trust in-place the client will only need to send a KRB_TGS_REQ to the following Kerberos KDCs:

1. SALES.WEST.EXAMPLE.COM
2. SALES.EAST.EXAMPLE.COM

This type of additional cross-realm trust is sometimes referred to as a “shortcut trust”. However, if the client is located in WEST.EXAMPLE.COM rather than SALES.WEST.EXAMPLE.COM the additional shortcut trust does not impact the path taken. In this case the path taken is:

1. WEST.EXAMPLE.COM
2. EXAMPLE.COM
3. EAST.EXAMPLE.COM
4. SALES.EAST.EXAMPLE.COM

The algorithm that decides the shortest-path can only use explicit trusts that are at or above the current position in the tree.

CROSS-REALM REQUIREMENTS

Now that we have an understanding of the process of Kerberos authentication across realms, we can examine the requirements for making the authentication process work correctly. The following items are required for cross-realm authentication to operate in any given environment:

- Network access from the client host to at least one KDC for each Realm.
- Mapping of network names to Realm names.
- A set of shared cross-realm principals and associated long-term keys.

REQUIREMENTS FOR NETWORK CONNECTIVITY

From the detailed examination of the messages sent across the network during cross-realm authentication, it is clear that the client must be able to access a KDC in each Kerberos Realm the authentication process needs to traverse. In the simple example shown in Figure 7, this means connecting to two different KDCs. However, for the more complex case, shown in Figure 8, this is five different KDCs. If these different KDCs are geographically spread around the world, or even a country, this can lead to delays in the network traffic which will lead to delays in the authentication process.

In addition, depending on the software components being used, a server could become a client. By this we mean that once authenticated to a service we can then use Kerberos authentication from this service to connect to another service. One example of such a connection is to use SSH to connect to one host and then once successfully connected to the first host make a second SSH connection from that host to another. If we use Kerberos authentication for both connections, what was the server in the first connection, becomes the client in the second connection. So now the server needs to be able to access the KDCs to successfully request a Service Ticket. This concept is called “delegation” within the Kerberos Protocol and this itself is an important topic in using Kerberos throughout a given environment.

The Kerberos authentication network messages are transported using either UDP or TCP network packets. By default, Kerberos will attempt to use UDP first and if the messages are too large will fallback to using TCP. For many clients the size of the message that will trigger the fallback to TCP can be defined. For the Microsoft Windows operating system, the size of the message is controlled by the MaxPacketSize registry DWORD value (Microsoft 2010). This is defined in the following registry subkey:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\
Kerberos\Parameters
```

For other clients, the message size can be defined in the Kerberos configuration file with the udp_preference_limit setting (MIT 2015). For both cases, setting the value to 1 will ensure that the message is sent via TCP rather than UDP. Using TCP rather than UDP packets can help when the KDCs are in different network segments than clients since often organizations will block UDP packets crossing the network boundary.

For both TCP and UDP packets, the default port used by the KDC processes is port 88. This is the same port used for both Active Directory and other implementations of Kerberos.

REQUIREMENTS FOR MAPPING NAMES OF NETWORKS AND REALMS

When creating the KRB_TGS_REQ message to request the Service Ticket, the client needs to be able to map the network domain name of the server hosting the service to a Kerberos Realm. The client can request a referral from its own Realm by sending the KRB_TGS_REQ message to a KDC in the Realm the client authenticated to. Alternatively, a specific mapping can be defined in the Kerberos configuration file (MIT 2015).

The [domain_realm] section of the Kerberos configuration file maps either a computer hostname or network domain name to a Kerberos Realm. The value in the configuration file can be a hostname or domain name, where domain names are indicated by a prefix of a period (.). For example:

```
[domain_realm]
    server1.sales.local = SALES.EAST.EXAMPLE.COM
    .dev.sales.local = SALES.WEST.EXAMPLE.COM
    sales.local = WEST.EXAMPLE.COM
```

This maps the host with the name server1.sales.local into the SALES.EAST.EXAMPLE.COM realm. The second entry maps all hosts under the network domain dev.sales.local into the SALES.WEST.EXAMPLE.COM realm, but not the host with the name dev.sales.local. That host is matched by the third entry, which maps the host sales.local and all hosts under the network domain sales.local that do not match a preceding rule into the realm WEST.EXAMPLE.COM.

We can see that using the Kerberos configuration file enables us to easily specify network domain to Kerberos Realm mappings that do not necessarily directly follow the network domain name structure. This can be very useful if you have a limited number of hosts within a network domain that need to be linked to a different Kerberos Realm without impacting the network domain names. For example, if you have two test servers with the network domain names test1.west.example.com and test2.west.example.com and these need to be linked to a Kerberos Realm TEST.EXAMPLE.COM. You can use the following in the [domain_realm] section:

```
[domain_realm]
    .west.example.com = WEST.EXAMPLE.COM
    test1.west.example.com = TEST.EXAMPLE.COM
    test2.west.example.com = TEST.EXAMPLE.COM
```

In this case, we need to ensure that the first entry is made since a hostname mapping has an associated implicit network domain name mapping. Without the first entry, all hosts in the network domain of west.example.com would be mapped to the TEST.EXAMPLE.COM realm.

In order to perform direct non-hierarchical cross-realm authentication, as shown by the green line in Figure 8, information is required to determine the authentication paths between realms. For Active Directory, because information about all trusts is stored in Active Directory, all domains in an Active Directory forest have knowledge of the trust relationships that are in place throughout the forest. Similarly, when two or more forests are joined together through forest trusts, the forest root domains in each forest have knowledge of the trust relationships that are in place throughout all of the domains in a trusted Active Directory forest (Microsoft 2014). This means that Microsoft Windows systems are aware of any direct cross-realm trusts without needing to provide additional configuration.

For other clients, the information to perform direct non-hierarchical cross-realm authentication can be provided in the Kerberos configuration file (MIT 2015). This information is provided in the [capaths] section of the Kerberos configuration file. A client will use this section to find the authentication path between its realm and the realm of the server. The server will use this section to verify the authentication path used by the client, by checking the transited field of the received ticket.

The format of the [capaths] section is relatively straightforward: each entry in the section is named after a realm in which a client might exist. Inside of that subsection, the set of intermediate realms from which

the client must obtain credentials is listed as values of the key, which corresponds to the realm in which a service might reside. If there are no intermediate realms, the value "." is used. For example:

```
[capaths]
SALES.WEST.EXAMPLE.COM = {
SALES.EAST.EXAMPLE.COM = .
EAST.EXAMPLE.COM = SALES.EAST.EXAMPLE.COM
}
```

Clients in the SALES.WEST.EXAMPLE.COM realm can obtain cross-realm credentials for SALES.EAST.EXAMPLE.COM directly from the SALES.WEST.EXAMPLE.COM KDC. If the clients wish to connect to the service in the EAST.EXAMPLE.COM realm, they will first need to obtain necessary credentials from the SALES.EAST.EXAMPLE.COM realm and then use those credentials to obtain credentials for use in the EAST.EXAMPLE.COM realm. This corresponds to the cross-realm trust shown with the green line in Figure 8.

REQUIREMENTS FOR MAINTAINING CROSS-REALM PRINCIPALS

The cross-realm principals and associated long-term keys are crucial to the cross-realm authentication working correctly. For our simple example for a client in REALM1 to access a service in REALM2, both the Kerberos realms must share a long-term key for a principal named **krbtgt/REALM2@REALM1** and both long-term keys must have the same key version number associated with them. The key version number is a tag associated with encrypted data that identifies which key was used for encryption when a long-term key associated with a principal changes over time. The key version number is incremented each time the password for the principal is changed.

Creating the principal **krbtgt/REALM2@REALM1** only allows for cross-realm authentication in one direction. This means only clients in REALM1 can access resources in REALM2. For clients in REALM2 to access resources in REALM1 we need to create the principal **krbtgt/REALM1@REALM2** in both realms; and again ensure the long-term key and key version number are the same in both realms. Adding the second cross-realm principal and long-term key enables two-way authentication. Active Directory manages the cross-realm principals when trusts are configured. MIT Kerberos requires that the principals be managed manually.

CROSS-REALM WITH SAS

In this final section, we will examine the impact and requirements for Kerberos cross-realm authentication on the SAS environment. We will split the SAS environment into its different tiers and consider each in turn.

MIDDLE-TIER AUTHENTICATION

The SAS Middle Tier can be configured for Kerberos authentication. Details on this configuration can be found in multiple SAS Global Forum papers (Rogers 2016), (Rogers 2013), and (Li and Roda, 2014) as well as the SAS® 9.4 Intelligence Platform: Middle-Tier Administration Guide, Fourth Edition. Kerberos cross-realm impacts the SAS Middle Tier when the SPN for the middle tier is located in a separate realm from the clients. Using Kerberos cross-realm authentication does not present any major challenges for the configuration of the middle tier. It is important to ensure the Kerberos configuration file used by the SAS Web Application Server instances has the correct [domain_realm] mappings. Without the correct mappings the authentication can fail.

If you are configuring Kerberos delegation, as detailed in Kerberos Delegation with SAS® 9.4 (Rogers, 2016), there are further considerations for Kerberos cross-realm authentication. Since with Kerberos Delegation the SAS Middle Tier will operate as both a Kerberized server and client, more thought needs to be given to the network constraints of cross-realm authentication. The SAS Middle Tier will attempt to obtain a Service Ticket for the SAS Server Tier using the TGT forwarded by the client.

Since the forwarded TGT is only valid for the client's realm, the SAS Web Application Server instance must be able to connect to this realm as well as its own realm. Also, the Kerberos configuration file must correctly identify the hosts running the KDC process for both the client's realm and the SAS Web

Application Server's realm. Finally, it might be necessary to update the Service Principal Name registered in SAS® Metadata Server for SAS® Workspace Server if SAS Workspace Server is in a different realm than the client. Further details can be found in (Li and Roda, 2014). With this in place is it possible to perform Kerberos Delegation across different realms.

When we discuss delegation in relation to the SAS Middle Tier, because of limitations in the Java Runtime used by the SAS Web Application Server, it is important to callout that only full delegation is supported. Microsoft has extended the Kerberos specification to support constrained delegation. This constrained delegation is not supported with the SAS Middle Tier. Also, constrained delegation is not supported by Microsoft across different realms. So cross-realm authentication delegation must be to any service using Kerberos. Figure 9 shows this setting for an account object in Active Directory.

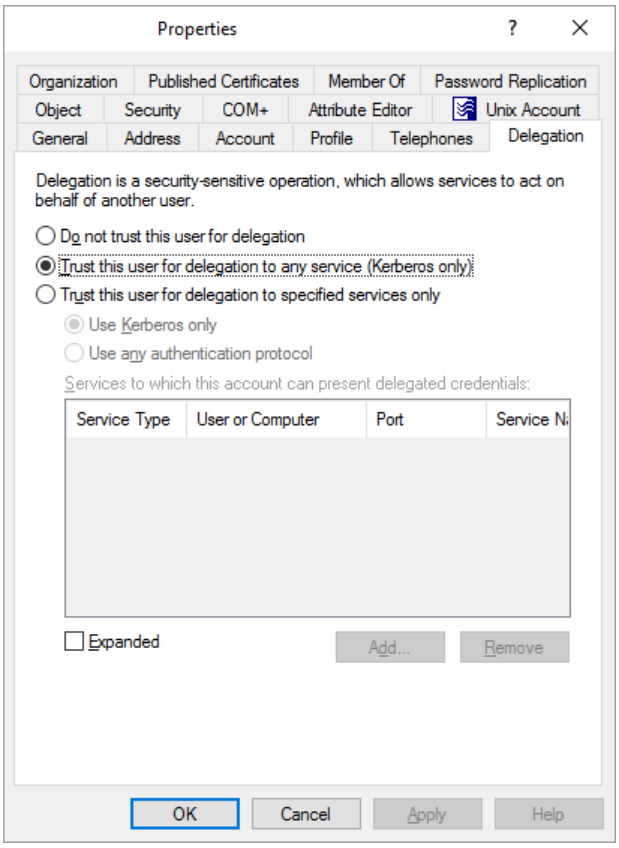

Figure 9. Active Directory Delegation Settings

SERVER TIER AUTHENTICATION

Kerberos authentication to the SAS Server Tier is most often to either SAS Metadata Server or SAS® Object Spawner. Both of these support Kerberos authentication from either a SAS desktop client such as SAS® Enterprise Guide or SAS® Management Console and from the SAS Middle Tier when using delegated Kerberos authentication. The SAS Server Tier supports cross-realm authentication. When the SAS Server Tier is running on Microsoft Windows this is straight forward. When the SAS Server Tier is running on a UNIX platform, such as Linux, this requires more thought.

On the UNIX platform, as with any other Kerberos server process, ensuring that the Kerberos configuration file is correctly configured to enable cross-realm authentication is key. The system administrator needs to ensure that the underlying Kerberos environment is correctly configured to support cross-realm authentication. There is nothing additional required by SAS.

Another SAS Server Tier component that requires more planning is the distributed SAS® High-Performance Analytics environment. The distributed SAS High-Performance Analytics environment provides both the distributed SAS® LASR Analytic Server and distributed SAS® High-Performance

Analytics Server. The distributed SAS High-Performance Analytics environment runs across multiple Linux hosts. Kerberos authentication can be used to authenticate the end user to the distributed SAS High-Performance Analytics environment. However, in this case, the Kerberos client will be SAS® Foundation session running the code that is used to connect to the SAS High-Performance Analytics environment. This SAS Foundation session can be either run standalone or could be launched by the SAS Object Spawner.

SAS Foundation session connects to SAS High-Performance Analytics environment using Secure Shell (SSH). An initial SSH connection is made from SAS Foundation session to the root node of the distributed SAS High-Performance Analytics environment. Then further SSH sessions are initialized from the root node to the worker nodes. Therefore, to use Kerberos authentication for these SSH sessions we must have Kerberos delegation configured for all SAS High-Performance Analytics environment nodes. This requires the system administrator to correctly configure SSH for delegation on all SAS High-Performance Analytics environments nodes.

Kerberos cross-realm authentication can further complicate this configuration of the SAS High-Performance Analytics environment nodes since we must ensure the correct network configuration to enable the cross-realm authentication to take place. Remember from our earlier sections we need to ensure that the Kerberos client has access to the client's realm. For this distributed SSH case each node must be able to connect to both its own realm and the client realm for authentication to be successful. This can be validated outside of SAS by being able to SSH using Kerberos from any node to any node.

THIRD-PARTY AUTHENTICATION

Kerberos authentication from the SAS environment out to a third-party data source, such as Hadoop, is the most common area where we see cross-realm authentication. The SAS environment and the third-party data source are typically located in different realms resulting in the need for cross-realm authentication for SAS to access the data source. This is common for accessing third-party Hadoop environments, where the Hadoop environment has been configured for Kerberos authentication using a separate Kerberos realm.

The SAS environment can use Kerberos delegation from the SAS Server Tier to access the third-party Hadoop environment. Alternatively, the SAS environment can be configured to initialize the end user's Kerberos credentials as the SAS session is launched. In this second case SAS does not perform the first phase of Kerberos authentication. The process to obtain the TGT for the end user must be performed by the operating system hosting the SAS session. Once the TGT is available, SAS can make use of this to access the third-party Hadoop environment.

The use of cross-realm authentication when the operating system initializes the end user's Kerberos credentials is separated from the SAS environment configuration. Since the TGT is obtained by the operating system, the operating system Kerberos configuration needs to be correctly configured to be able to obtain the Service Tickets for Hadoop using cross-realm authentication. There is nothing additional that needs to be configured in the SAS environment.

Within the third-party Hadoop environment it is crucial to ensure that the configuration is correctly setup to obtain a user ID from the Kerberos principal. The Kerberos principal is included in the Service Ticket used to connect to the Hadoop environment. The Hadoop configuration includes a set of rules to map a Kerberos principal to a short name. These Hadoop specific rules are defined in the Hadoop configuration property, **hadoop.security.auth_to_local,** within the core-site.xml file.

To specify a mapping rule, use the prefix string RULE: followed by three sections—principal translation, acceptance filter, and short name substitution—described in more detail below. The syntax of a mapping rule is:

```
RULE:[<principal translation>](<acceptance filter>)<short name
substitution>
```

The first section of a rule, <principal translation>, performs the matching of the principal name to the rule. If there is a match, the principal translation also does the initial translation of the principal name to a short

name. In the <principal translation> section, you specify the number of components in the principal name and the pattern you want to use to translate those principal components and realm into a short name.

The second section of a rule, <acceptance filter>, matches the translated short name from the principal translation (that is, the output from the first section). The acceptance filter is specified in "( )" characters and is a standard regular expression. A rule matches only if the specified regular expression matches the entire translated short name from the principal translation.

The third and final section of a rule is the <short name substitution>. If there is a match in the second section, the acceptance filter, the <short name substitution> section does a final translation of the short name from the first section. This translation is a sed replacement expression (s/.../.../g) that translates the short name from the first section into the final short name string. The short name substitution section is optional. In many cases, it is sufficient to use the first two sections only.

The following rule set converts user accounts in the SALES.EXAMPLE.COM domain to lowercase.

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
RULE:[1:$1@$0](.*@\QSALES.EXAMPLE.COM\E$)s/@\QSALES.EXAMPLE.COM\E$///L
RULE:[2:$1@$0](.*@\QSALES.EXAMPLE.COM\E$)s/@\QSALES.EXAMPLE.COM\E$///L
DEFAULT</value>
</property>
```

The first rule applies to principals with a single component, USER@REALM, while the second rule applies to principals with two components, for example, HIVE/HOSTNAME@REALM. In this example, the JohnDoe@SALES.EXAMPLE.COM principal becomes the johndoe HDFS user. The last rule in the example above is the DEFAULT rule. The default rule reduces a principal name down to its first component only. The default rule applies only if the principal is in the default realm. If a principal name does not match any of the specified rules, the mapping for that principal name will fail.

The mapping of Kerberos principals to short names is the most common area for errors to arise. Without the correct mappings the security applied to HDFS will not work and users will not be able to access the files and directories in HDFS that they expect to be able to access.

CONCLUSION

Kerberos security is being implemented by a growing number of customers to enable strong authentication across different parts of an organization in a safe and secure way. SAS is designed to work within this secure environment. I hope that this paper helps unravel the mysteries of Kerberos for SAS administrators, and enables them to work more closely with their IT and Security counterparts when SAS is deployed in an environment that uses Kerberos for basic authentication, and in the more complex case of cross-realm authentication.

Kerberos authentication can be a confusing topic. I hope that this paper has helped to explain how Kerberos authentication operates in both a basic case and in the more complex case of cross-realm authentication. This paper has aimed to make you aware of the intricacies of TGTs, long-term keys, and cross-realm trusts. Being aware of how the Kerberos authentication process functions will help you to see how it can be applied to your environments. Knowing about the requirements and considerations will allow you to better plan for enabling strong authentication for your end-users and making effective use of Kerberos authentication.

REFERENCES

IETF. 2005. “The Kerberos Network Authentication Service (V5).” Available https://www.ietf.org/rfc/rfc4120.txt. Accessed on January 6, 2017.

Microsoft. 2010. “How to force Kerberos to use TCP instead of UDP in Windows.” Available https://support.microsoft.com/en-gb/kb/244474. Accessed on January 6, 2017.

MIT. 2015. “MIT Kerberos Documentation.” Available http://web.mit.edu/Kerberos/krb5-1.12/doc/admin/conf_files/krb5_conf.html. Accessed on January 6, 2017.

Microsoft. 2014. “How Domain and Forest Trusts Work.” Available https://technet.microsoft.com/en-us/library/cc773178(v=ws.10).aspx. Accessed on January 6, 2017.

Li, Zhiyong and Mike Roda. 2014. “An Advanced Fallback Authentication Framework for SAS® 9.4 and SAS® Visual Analytics”. *Proceedings of the SAS Global Forum 2014 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings14/SAS102-2014.pdf.

Rogers, Stuart J. 2013. “Kerberos and SAS® 9.4: A Three-Headed Solution for Authentication”. *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings13/476-2013.pdf.

Rogers, Stuart J. 2016. “Kerberos Delegation with SAS® 9.4.” *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings16/SAS3443-2016.pdf.

Roda, Mike. 2016. “Tips and Best Practices for Configuring Integrated Windows Authentication.” *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings16/SAS3720-2016.pdf.

ACKNOWLEDGMENTS

I would like to thank the following people for taking the time to review and contribute to this paper:

- Larry Noe
- Donna Bennett
- Mike Roda
- Grant Whiteheart

RECOMMENDED READING

- *SAS® 9.4 Intelligence Platform: Installation and Configuration Guide, Third Edition*
- *SAS® 9.4 Intelligence Platform: Security Administration Guide, Third Edition*
- *SAS® 9.4 Intelligence Platform: Middle-Tier Administration Guide, Fourth Edition*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Stuart J Rogers
SAS Institute Inc.
stuart.rogers@sas.com
http://www.sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.