Kerberos

Secure and efficient authentication and key distribution

Johannes Lötzsch and Meike Zehlike

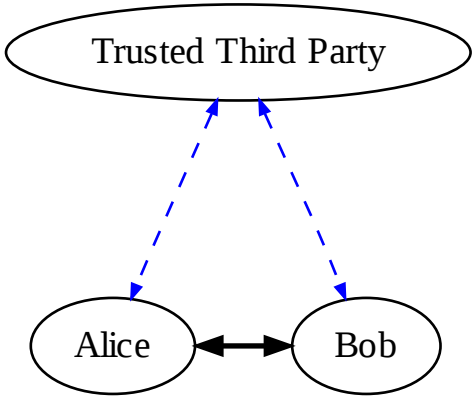
October 27, 2009

Table of content

Use cases for Kerberos / Alternatives

Trusted Instance for:

- Keydistribution
- Authentication
- SingleSignOn



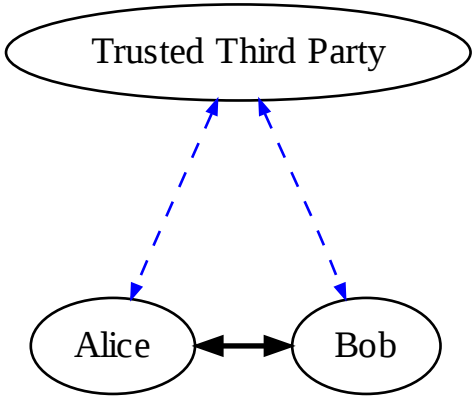
Well known alternative Trusted Third Parties:

- CA in hierarchical PKI
  - Asymmetric Cryptography $\Longrightarrow$ slow / expensive
- Members of Web of Trust (e.g. PGP)
  - Hard to say how trustworthy a „Trustpath“ is

Use cases for Kerberos / Alternatives

Trusted Instance for:

- Keydistribution
- Authentication
- SingleSignOn

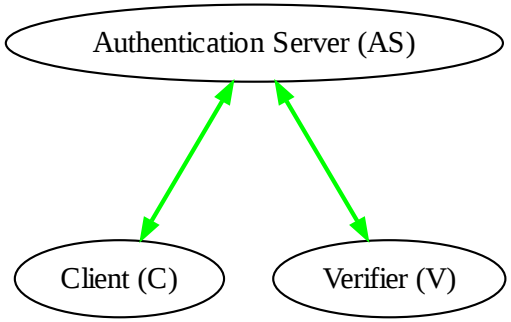


Well known alternative Trusted Third Parties:

- CA in hierarchical PKI
  - Asymmetric Cryptography $\Longrightarrow$ slow / expensive
- Members of Web of Trust (e.g. PGP)
  - Hard to say how trustworthy a „Trustpath“ is

Basic Protocol

- Based on symmetric Needham-Schroeder-Protocol
- Assumption: Each participant exchanged Key with AS on save channel $(K_{AS,C}, K_{AS,V})$

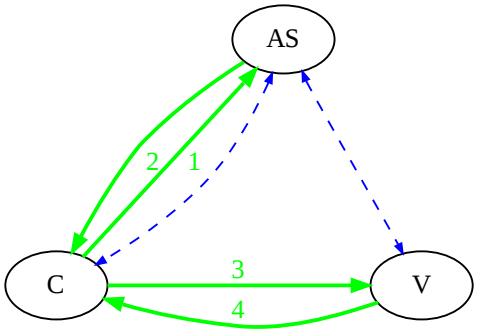


Procedure:

1. Authentication Request: $(C, V, time_{exp}, n)$
2. Authentication Reply: $E_{K_{AS,C}}(V, time_{exp}, n, K_{C,V}), E_{K_{AS,V}}(C, time_{exp}, K_{C,V})$
3. Application Request: $E_{K_{AS,V}}(C, time_{exp}, K_{C,V}), E_{K_{C,V}}(ts, K_{subSession}, ck)$
4. Application Reply: $E_{K_{C,V}}(ts)$

Basic Protocol

- Based on symmetric Needham-Schroeder-Protocol
- Assumption: Each participant exchanged Key with AS on save channel $(K_{AS,C}, K_{AS,V})$

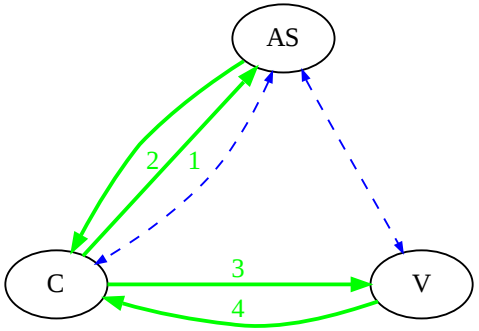


Procedure:

1. Authentication Request: $(C, V, time_{exp}, n)$
2. Authentication Reply: $E_{K_{AS,C}}(V, time_{exp}, n, K_{C,V}), E_{K_{AS,V}}(C, time_{exp}, K_{C,V})$
3. Application Request: $E_{K_{AS,V}}(C, time_{exp}, K_{C,V}), E_{K_{C,V}}(ts, K_{subSession}, ck)$
4. Application Reply: $E_{K_{C,V}}(ts)$

Basic Protocol

- Based on symmetric Needham-Schroeder-Protocol
- Assumption: Each participant exchanged Key with AS on save channel $(K_{AS,C}, K_{AS,V})$

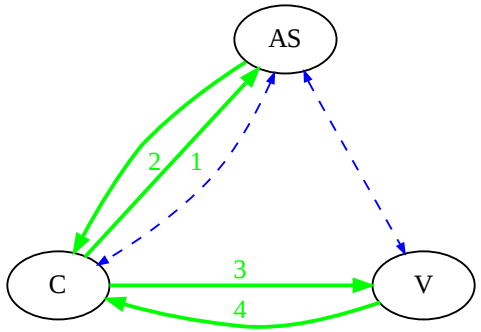


Procedure:

1. Authentication Request: $(C, V, time_{exp}, n)$
2. Authentication Reply: $E_{K_{AS,C}}(V, time_{exp}, n, K_{C,V}), E_{K_{AS,V}}(C, time_{exp}, K_{C,V})$
3. Application Request: $E_{K_{AS,V}}(C, time_{exp}, K_{C,V}), E_{K_{C,V}}(ts, K_{subSession}, ck)$
4. Application Reply: $E_{K_{C,V}}(ts)$

Basic Protocol

- Based on symmetric Needham-Schroeder-Protocol
- Assumption: Each participant exchanged Key with AS on save channel $(K_{AS,C}, K_{AS,V})$

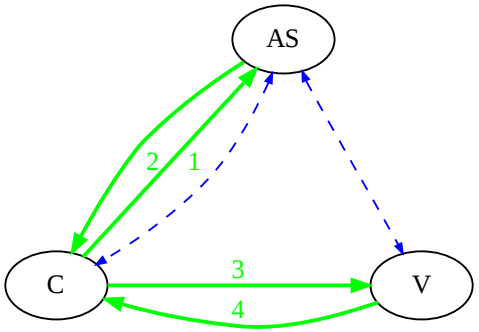


Procedure:

1. Authentication Request: $(C, V, time_{exp}, n)$
2. Authentication Reply: $E_{K_{AS,C}}(V, time_{exp}, n, K_{C,V}), E_{K_{AS,V}}(C, time_{exp}, K_{C,V})$
3. Application Request: $E_{K_{AS,V}}(C, time_{exp}, K_{C,V}), E_{K_{C,V}}(ts, K_{subSession}, ck)$
4. Application Reply: $E_{K_{C,V}}(ts)$

Basic Protocol

- Based on symmetric Needham-Schroeder-Protocol
- Assumption: Each participant exchanged Key with AS on save channel $(K_{AS,C}, K_{AS,V})$

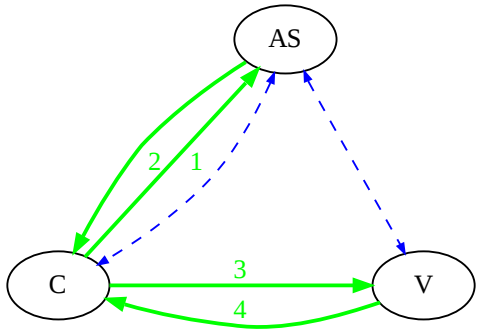


Procedure:

1. Authentication Request: $(C, V, time_{exp}, n)$
2. Authentication Reply: $E_{K_{AS,C}}(V, time_{exp}, n, K_{C,V}), E_{K_{AS,V}}(C, time_{exp}, K_{C,V})$
3. Application Request: $E_{K_{AS,V}}(C, time_{exp}, K_{C,V}), E_{K_{C,V}}(ts, K_{subSession}, ck)$
4. Application Reply: $E_{K_{C,V}}(ts)$

Basic Protocol

- Based on symmetric Needham-Schroeder-Protocol
- Assumption: Each participant exchanged Key with AS on save channel ($K_{AS,C}, K_{AS,V}$)

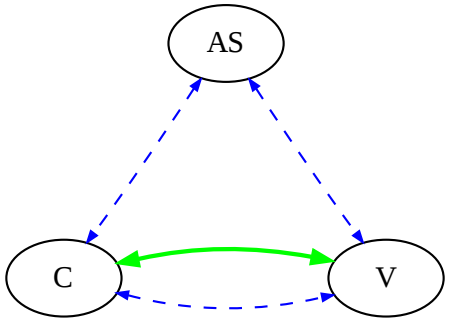


Procedure:

1. Authentication Request: $(C, V, time_{exp}, n)$
2. Authentication Reply: $E_{K_{AS,C}}(V, time_{exp}, n, K_{C,V}), E_{K_{AS,V}}(C, time_{exp}, K_{C,V})$
3. Application Request: $E_{K_{AS,V}}(C, time_{exp}, K_{C,V}), E_{K_{C,V}}(ts, K_{subSession}, ck)$
4. Application Reply: $E_{K_{C,V}}(ts)$

Basic Protocol

- Based on symmetric Needham-Schroeder-Protocol
- Assumption: Each participant exchanged Key with AS on save channel ($K_{AS,C}, K_{AS,V}$)



Procedure:

1. Authentication Request: $(C, V, time_{exp}, n)$
2. Authentication Reply: $E_{K_{AS,C}}(V, time_{exp}, n, K_{C,V}), E_{K_{AS,V}}(C, time_{exp}, K_{C,V})$
3. Application Request: $E_{K_{AS,V}}(C, time_{exp}, K_{C,V}), E_{K_{C,V}}(ts, K_{subSession}, ck)$
4. Application Reply: $E_{K_{C,V}}(ts)$

Ticket Granting Tickets

- $K_{AS,C}$ needs to be handled with caution
- usage from different machines
  - should not be stored on clients $\Longrightarrow$ always created from a password

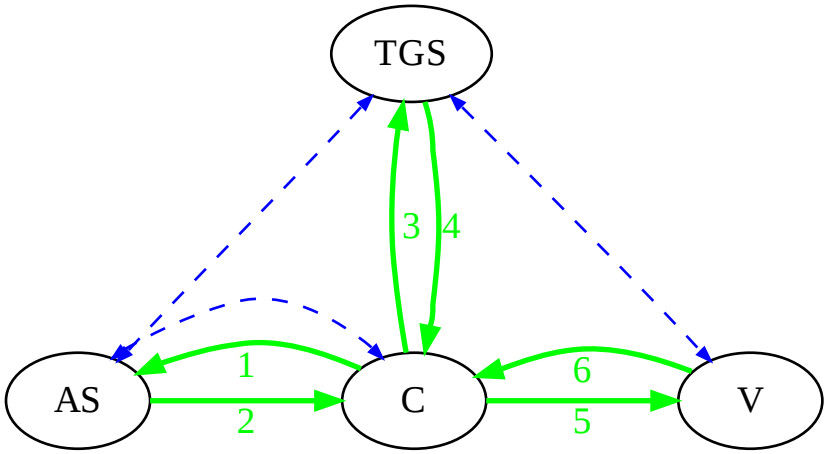
Howto cache credentials?

- 1-2: Authentication with AS using password $\Longrightarrow$ TGT
- 3-4: Authentication with TGS using TGT $\Longrightarrow$ SessionTicket for V
- 5-6: Authentication with V using SessionTicket
- 3-6 can be repeated until $time_{exp}$ of TGT

Ticket Granting Tickets

- $K_{AS,C}$ needs to be handled with caution
- usage from different machines
  - should not be stored on clients $\Longrightarrow$ always created from a password

Howto cache credentials?

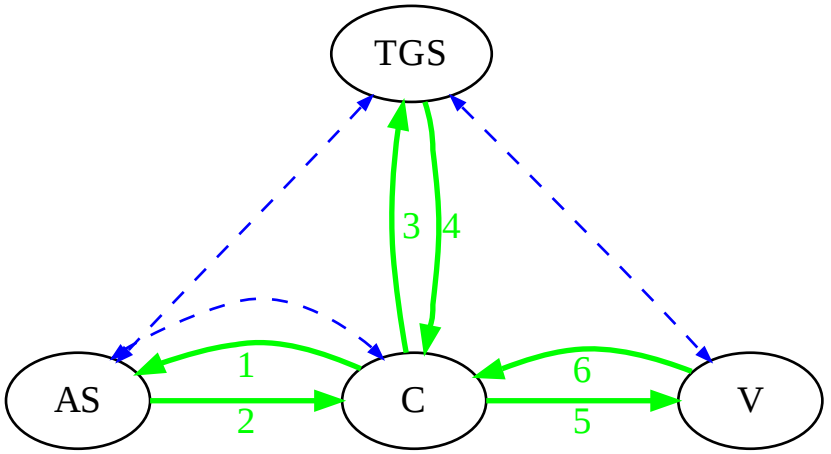


- 1-2: Authentication with AS using password $\Longrightarrow$ TGT
- 3-4: Authentication with TGS using TGT $\Longrightarrow$ SessionTicket for V
- 5-6: Authentication with V using SessionTicket
- 3-6 can be repeated until $time_{exp}$ of TGT

Ticket Granting Tickets

- $K_{AS,C}$ needs to be handled with caution
- usage from different machines
  - should not be stored on clients $\Longrightarrow$ always created from a password

Howto cache credentials?

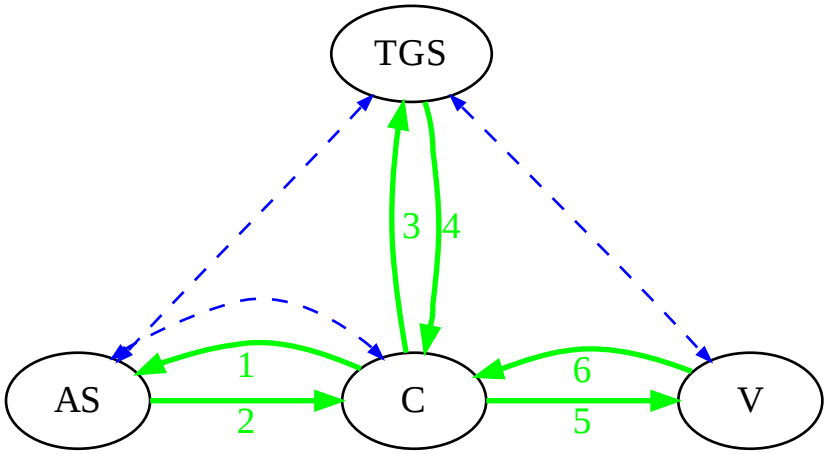


- 1-2: Authentication with AS using password $\Longrightarrow$ TGT
- 3-4: Authentication with TGS using TGT $\Longrightarrow$ SessionTicket for V
- 5-6: Authentication with V using SessionTicket
- 3-6 can be repeated until $time_{exp}$ of TGT

Ticket Granting Tickets

- $K_{AS,C}$ needs to be handled with caution
- usage from different machines
  - should not be stored on clients $\Longrightarrow$ always created from a password

Howto cache credentials?

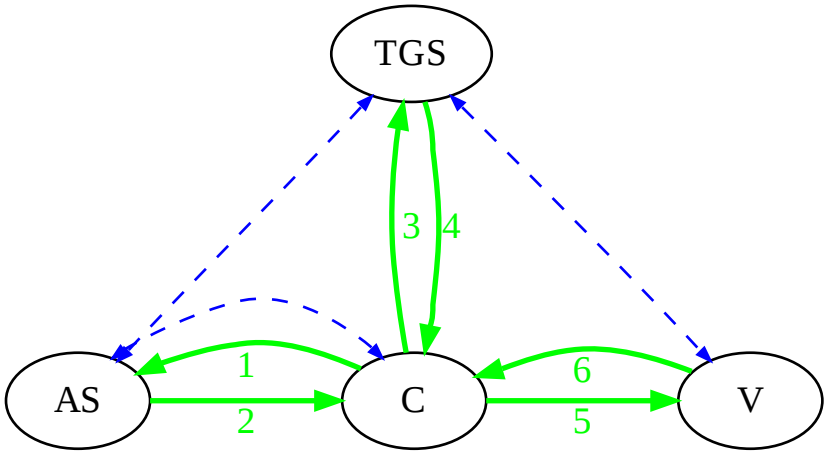


- 1-2: Authentication with AS using password $\Longrightarrow$ TGT
- 3-4: Authentication with TGS using TGT $\Longrightarrow$ SessionTicket for V
- 5-6: Authentication with V using SessionTicket
- 3-6 can be repeated until $time_{exp}$ of TGT

Ticket Granting Tickets

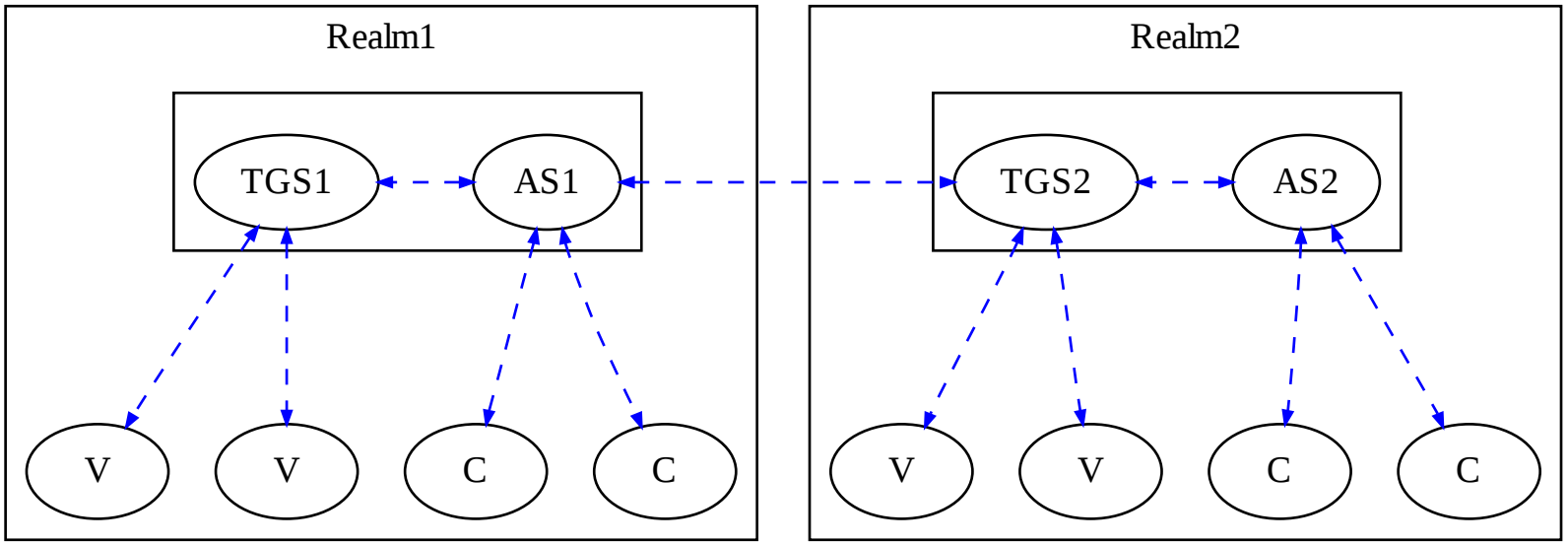
- $K_{AS,C}$ needs to be handled with caution
- usage from different machines
  - should not be stored on clients $\Longrightarrow$ always created from a password

Howto cache credentials?

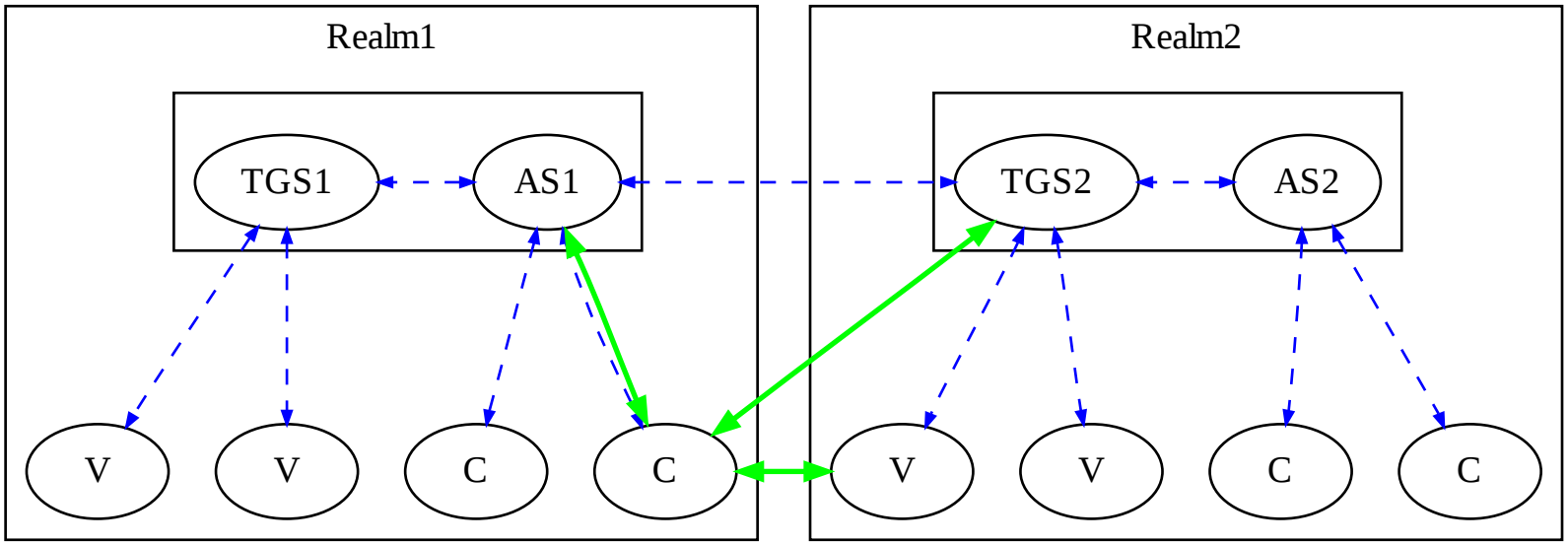


- 1-2: Authentication with AS using password $\Longrightarrow$ TGT
- 3-4: Authentication with TGS using TGT $\Longrightarrow$ SessionTicket for V
- 5-6: Authentication with V using SessionTicket
- 3-6 can be repeated until $time_{exp}$ of TGT

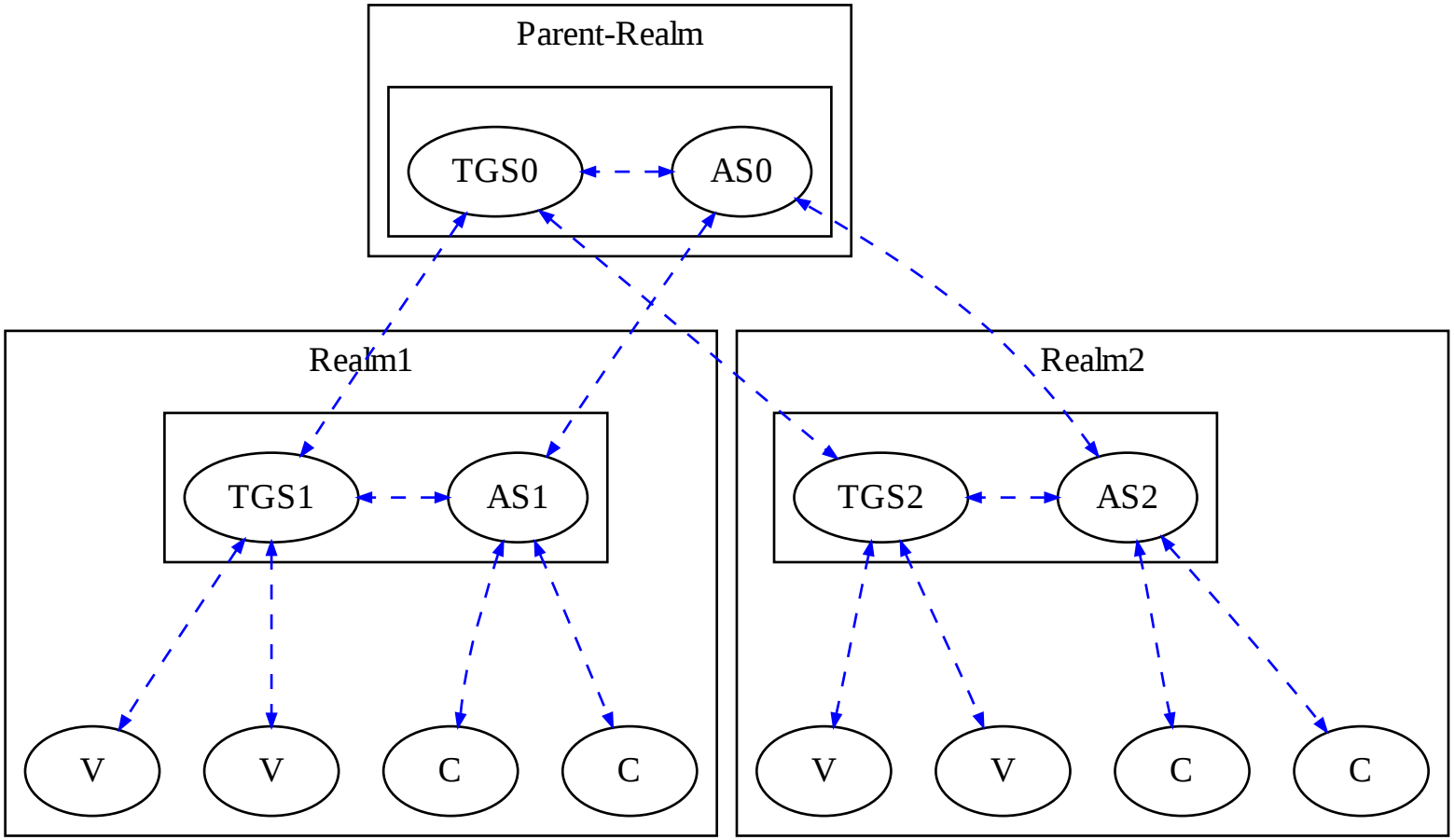
Cross-Realm-Authentication



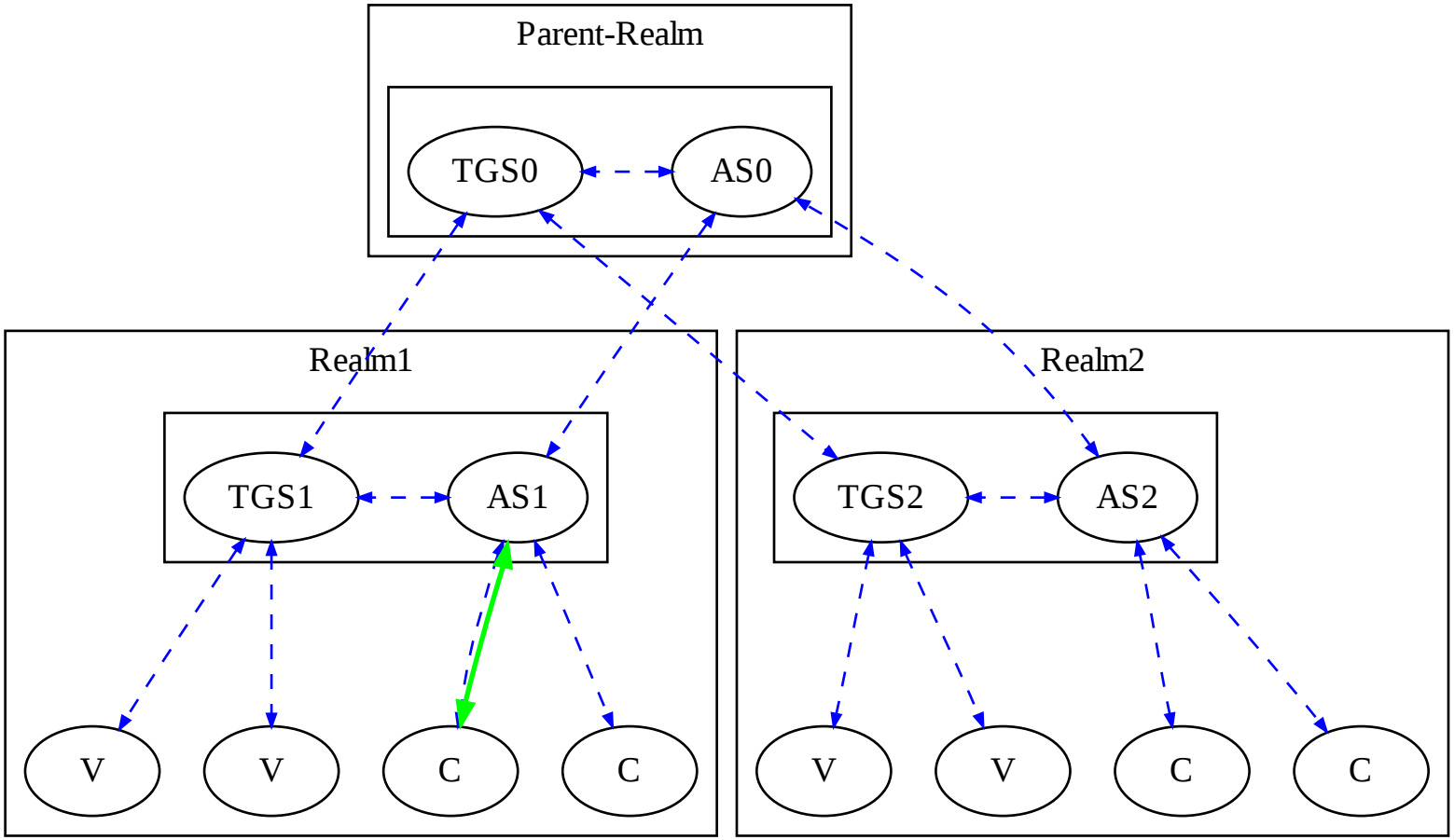
Cross-Realm-Authentication



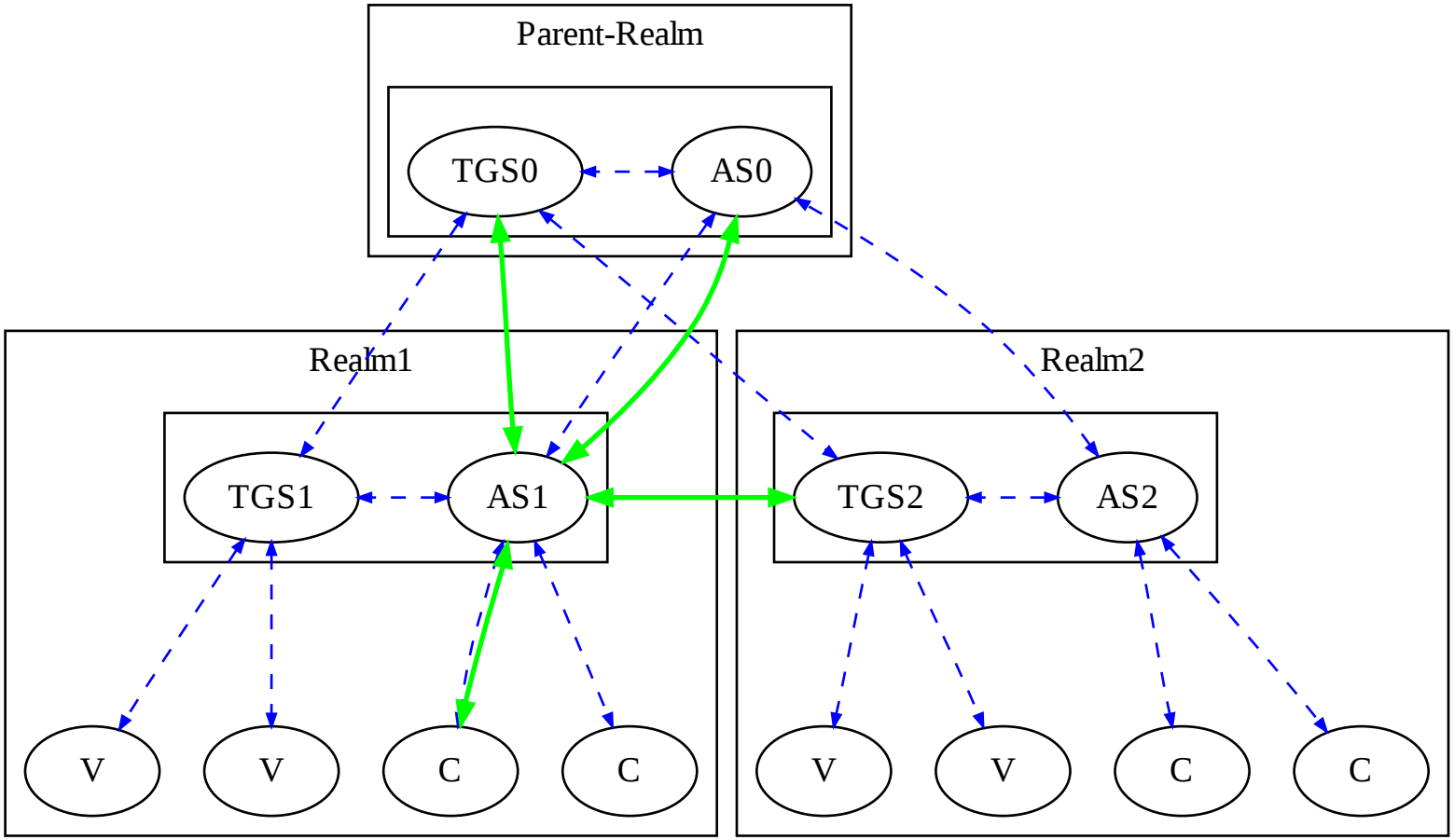
Authentication in hierarchic Realms



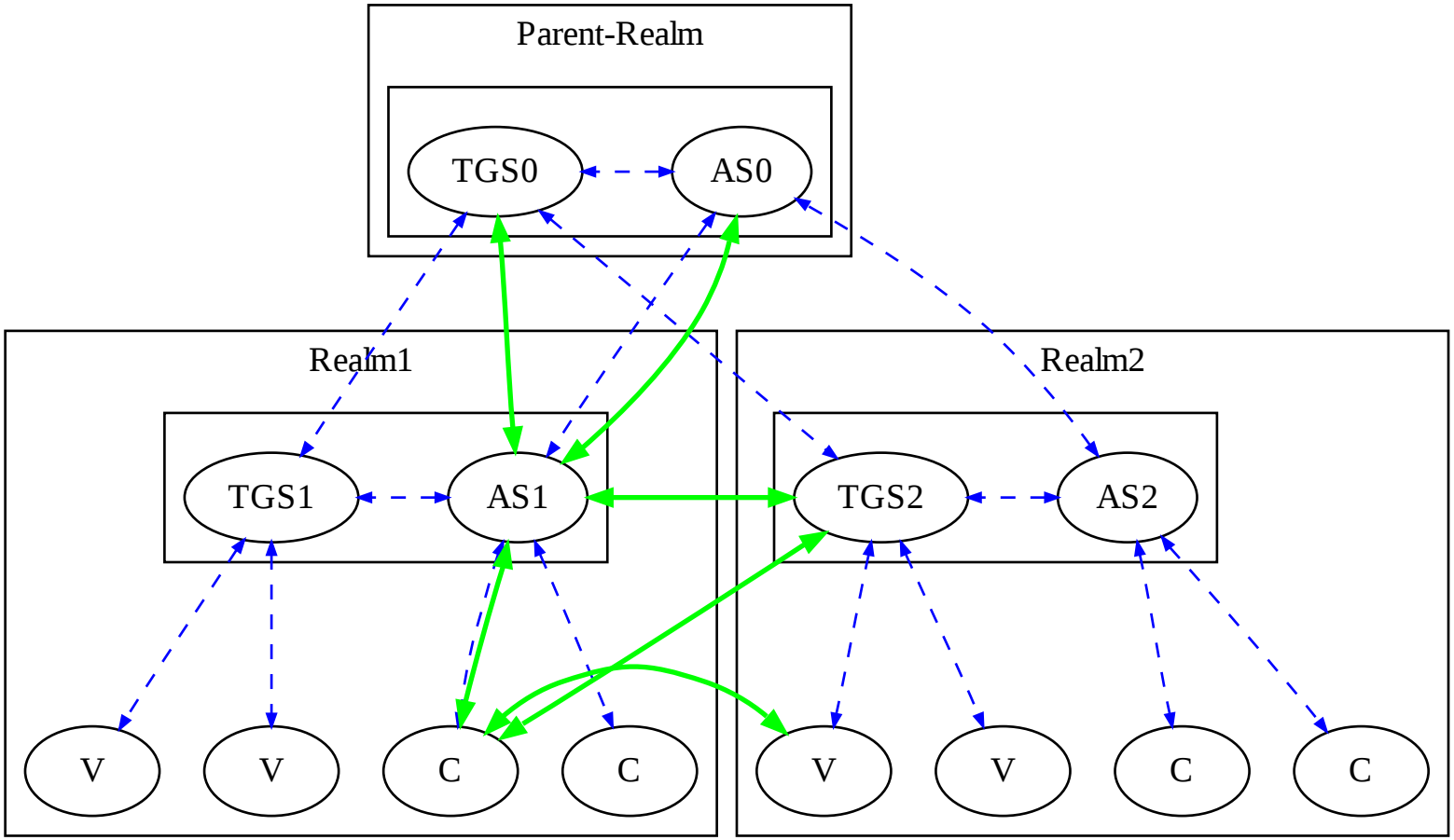
Authentication in hierarchic Realms



Authentication in hierarchic Realms



Authentication in hierarchic Realms



Installation on Debian / Ubuntu

$ wget http://johannesloetzsch.de/about/kerberos/setup_krb.sh

...
read, UNDERSTAND, modify
...

$ chmod +x setup_krb.sh
$ sudo ./setup_krb.sh

Add new user (create $K_{AS,C}$)

$ **sudo kadmin.local**

kadmin.local: **addprinc newuser**
Enter password for principal "newuser@EXAMPLE.COM": **$passwort**
Re-enter password for principal "newuser@EXAMPLE.COM": **$passwort**
Principal "newuser@EXAMPLE.COM" created.

kadmin.local: **list_principals**
...
newuser@EXAMPLE.COM
...

Add new server (create $K_{TGS,V}$)

kadmin.local: **addprinc -randkey host/servername.example.com**
Principal "host/servername.example.com@EXAMPLE.COM" created.

kadmin.local: **ktadd -k /tmp/key host/servername.example.com**
Entry for principal host/servername.example.com with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/tmp/key.

$ **sudo scp /tmp/key root@servername.example.com:/etc/krb5.keytab**

Obtaining the TGT

$ **kinit newuser**
Password for newuser@EXAMPLE.COM: **$passwort**
$ **klist -5**
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: newuser@EXAMPLE.COM
Valid starting Expires Service principal
10/07/09 20:16:04 10/08/09 06:16:04 krbtgt/EXAMPLE.COM@EXAMPLE.COM
renew until 10/08/09 20:15:58

$ **kdestroy**
$ **klist -5**
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_1000)

Use the Kerberos-principal for SSH

$ **yes ” |sudo adduser --disabled-password newuser**

$ **kdestroy**
$ **ssh newuser@servername.example.com**
newuser@servername.example.com’s password: **passwort is disabled**
Permission denied, please try again.

$ **kinit newuser**
$ Password for newuser@EXAMPLE.COM: **$passwort**
$ **ssh newuser@servername.example.com**
newuser@servername.example.com: $

Done — SingleSignOn works

$ **klist -5**
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: newuser@EXAMPLE.COM
Valid starting Expires Service principal
10/07/09 20:27:09 10/08/09 06:27:09 krbtgt/EXAMPLE.COM@EXAMPLE.COM
renew until 10/08/09 20:27:03
10/07/09 20:27:14 10/08/09 06:27:09
host/servername.example.com@EXAMPLE.COM
renew until 10/08/09 20:27:03

$ **ssh newuser@servername.example.com**
newuser@servername.example.com: $ **exit**
$ **ssh newuser@servername.example.com**
newuser@servername.example.com: $

$ **kdestroy**

Further literature

DIY

- http://en.wikipedia.org/wiki/Kerberos_(protocol)
- http://gost.isi.edu/publications/kerberos-neuman-tso.html
- http://techpubs.spinlocksolutions.com/dklar/kerberos.html
- http://linux.die.net/man/1/kerberos
- http://tools.ietf.org/html/rfc4120