

P4 VPN Authentication

Authentication of VPN Traffic on a Network
Device with P4

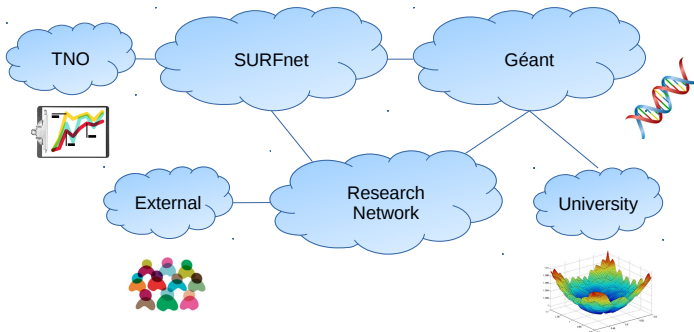
Jeroen Klomp

University of Amsterdam
System and Network Engineering
Research Project

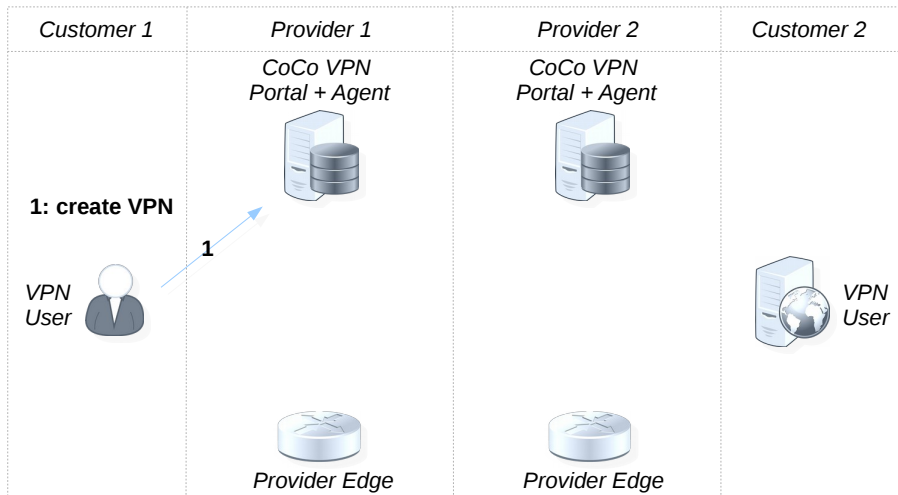
June 29, 2016

CoCo Introduction

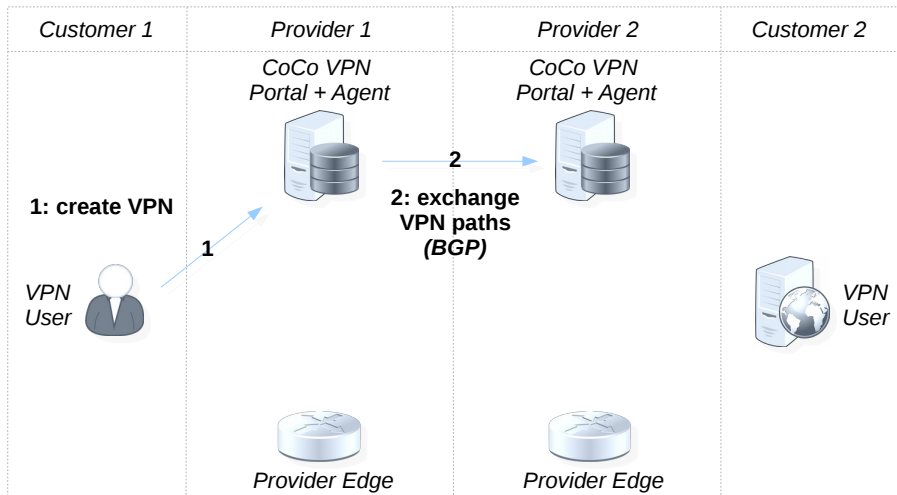
- Community Connection (CoCo)
 - User-initiated multi-domain VPN service
 - Support eScience
 - Prototype phase; no proper authentication



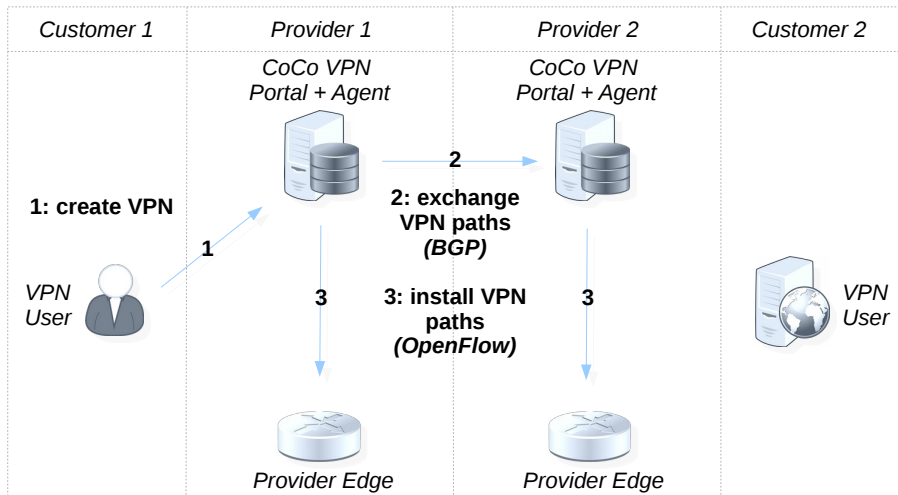
CoCo Overview



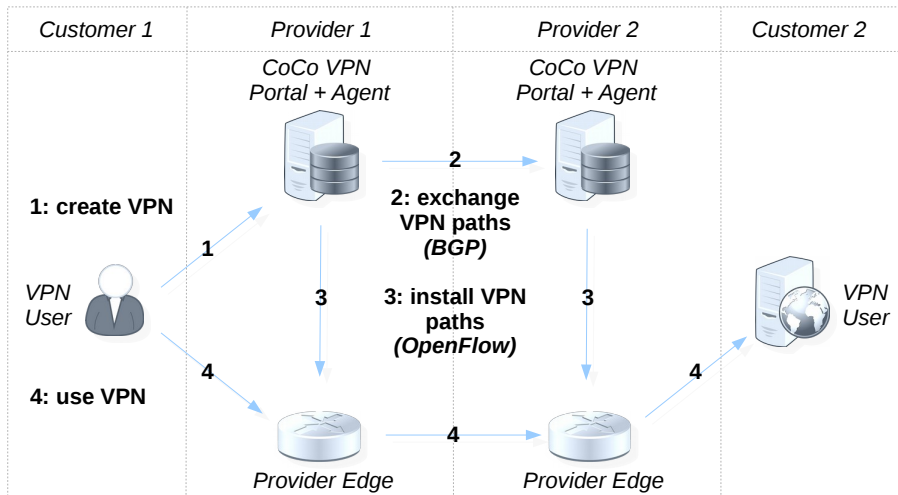
CoCo Overview



CoCo Overview



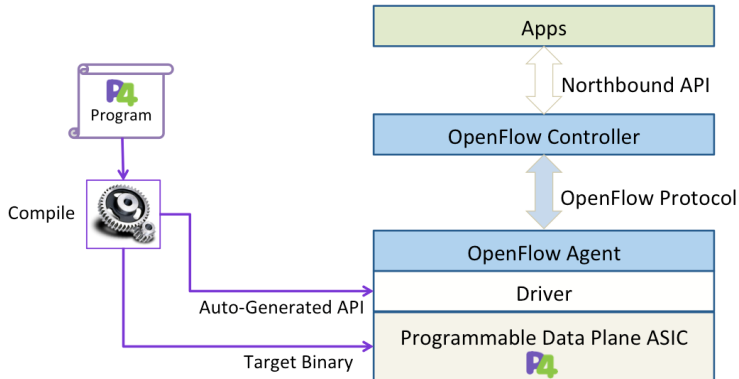
CoCo Overview



P4 Overview

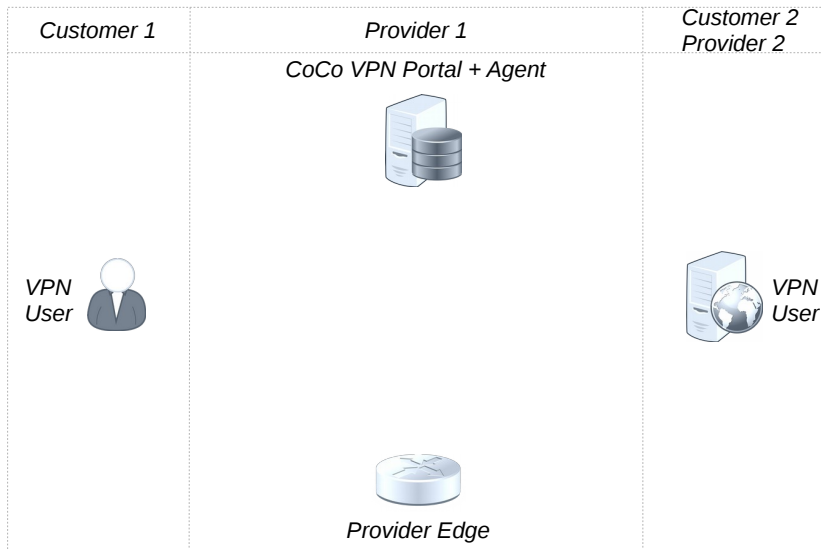


P4 & OpenFlow

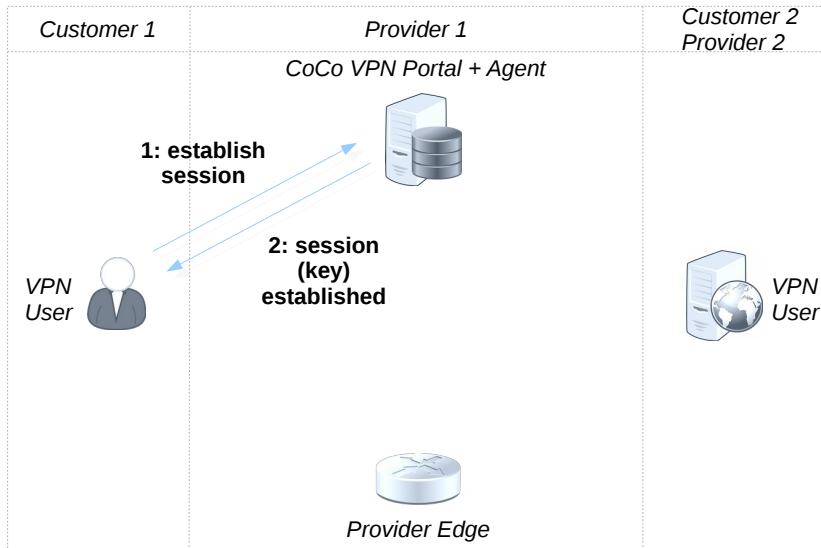


Copyright © 2016 P4 Language Consortium.

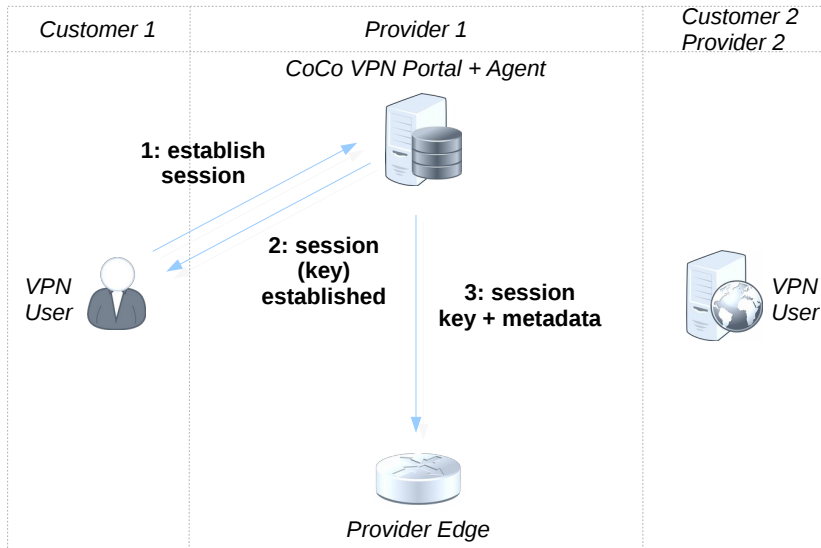
Authentication Use Case



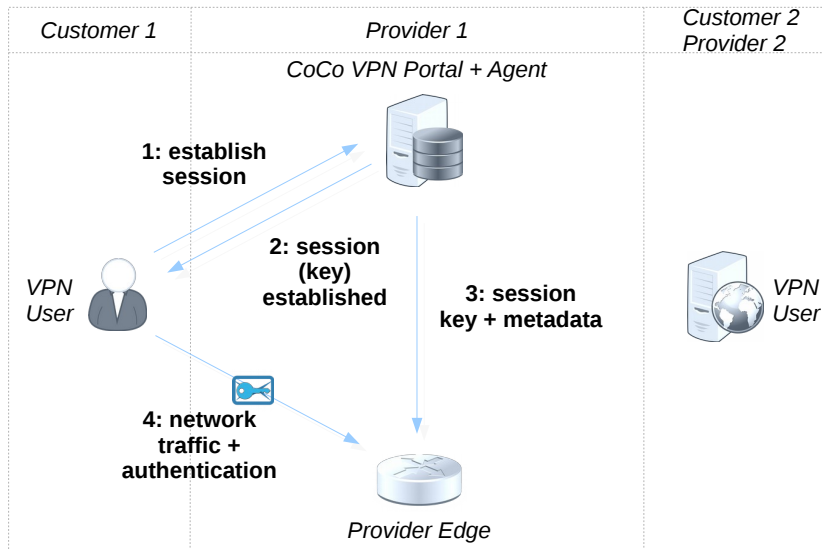
Authentication Use Case



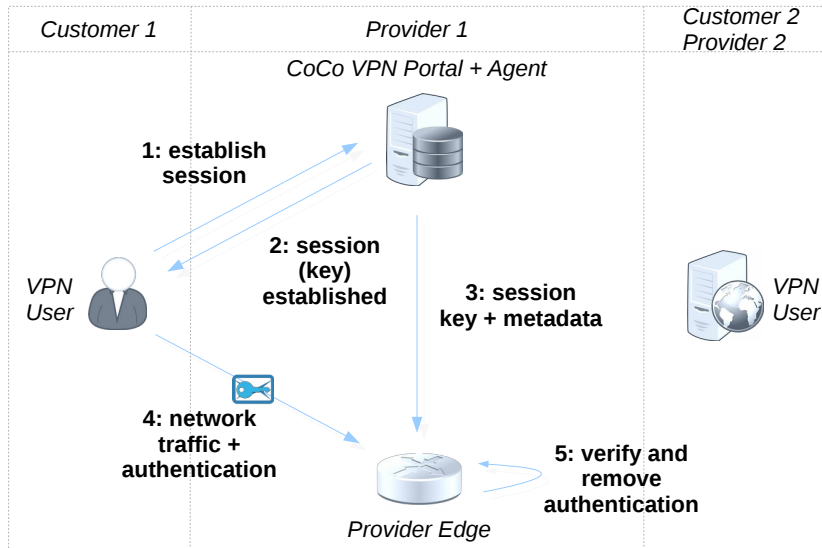
Authentication Use Case



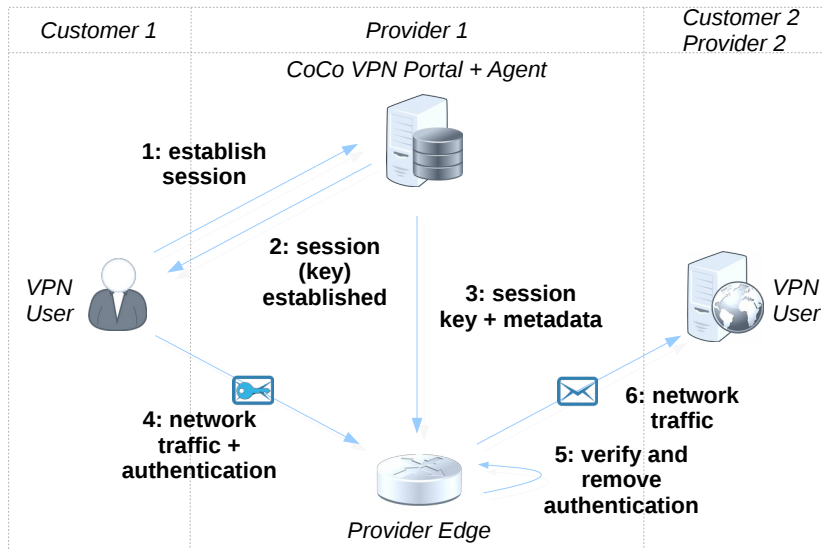
Authentication Use Case



Authentication Use Case



Authentication Use Case



Authentication Scheme

- Requirements
 - Secure
 - Support research

Authentication Scheme

- Requirements
 - Secure
 - Support research
- Authentication method
 - Message authentication code (MAC)

Authentication Scheme

- Requirements
 - Secure
 - Support research
- Authentication method
 - Message authentication code (MAC)
 - Like checksums, MAC algorithms cannot be implemented in P4 itself
 - Use external function provided by the target
 - Possibly a standard library with MAC algorithms?
 - Language needs facility for key input

Authentication Scheme

- Requirements
 - Secure
 - Support research
- Authentication method
 - Message authentication code (MAC)
 - Like checksums, MAC algorithms cannot be implemented in P4 itself
 - Use external function provided by the target
 - Possibly a standard library with MAC algorithms?
 - Language needs facility for key input
- Authentication protocol

Authentication Protocol

- IPSEC Authentication Header?
 - Contains all necessary fields
 - Security Parameters Index (SPI): Security Association → session ID
 - Sequence number → replay protection
 - Integrity Check Value (ICV) → variable length MAC

Offset		Type	0							1							2							3									
Bit	Octet		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
0	0	AH	Next Header							Payload Length							Reserved																
32	4		Security Parameters Index (SPI): <i>session identifier</i>																														
64	8		Sequence Number: <i>replay protection</i>																														
96	12		Integrity Check Value (ICV) (<i>variable MAC</i>)																														
128	16		-----																														
160	20		-----																														
192	24		-----																														
224	28		-----																														
256	32		<i>[~ padding]</i>																														
288	36	ICMP/ UDP/ TCP/...	Transport protocol and payload																														

Implementation in P4

- Distinguish sessions
 - Session identifier table containing session IDs and session keys

Implementation in P4

- Distinguish sessions
 - Session identifier table containing session IDs and session keys
- Sequence number
 - Register per session

Implementation in P4

- Distinguish sessions
 - Session identifier table containing session IDs and session keys
- Sequence number
 - Register per session
- MAC
 - Session key mixed with message
 - Hash¹ calculated and stored as metadata via primitive action:
`MODIFY_FIELD_WITH_HASH_BASED_OFFSET()`

¹'simulated' via checksum

Simplified Authentication Protocol

- GRE (Generic Routing Encapsulation)
 - Has necessary fields
 - Key: session ID
 - Sequence number
 - Checksum: (mis)used for MAC simulation
 - Easily craft packets e.g., via Scapy

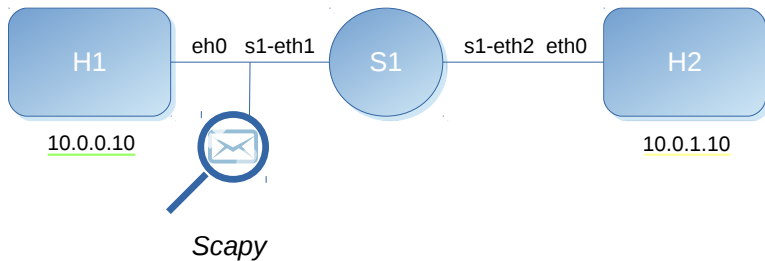
Offset		Type	0				1				2				3																			
Bit	Octet		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	GRE	Flags: <i>CKS</i>				Reserved0				Version				Protocol Type: <i>0000 (possibly GRE keepalive)</i>																			
32	4		Checksum: <i>MAC (CRC16)</i>								Offset: <i>key (not on wire)</i>																							
64	8		Key: <i>session identifier</i>																															
96	12		Sequence Number																															
128	16	ICMP	ICMP echo request with random payload																															


Test Setup


Mininet


P4


Mininet



 GRE(key)/ICMP(rqst)/'123' →

 ICMP(rqst)/'123'

 ICMP(rply)/'123' ←

 ICMP(rply)/'123'

Demonstration

```
####
sending packet: Identifier: 123456789 (0x75bcd15), hash key: 0xabc
d, sequence number: 123, checksum: 0xcfc45, payload: 538
.
Sent 1 packets.
response:
### Ethernet ###
dst = 00:04:00:00:00:00
src = 00:0a1bb:00:00:00
type = IPv4
### IP ###
version = 4L
ttl = 3L
tos = 0x0
len = 31
id = 13339
flags =
frag = 0L
ttl = 63
proto = icmp
checksum = 0x360c
src = 10.0.1.10
dst = 10.0.0.10
options \
### ICMP ###
type = echo-reply
code = 0
checksum = 0x02cc
id = 0x0
seq = 0x0
### Raw ###
load = '538'
> |

mininet> h1 ping -c1 h2
PING 10.0.1.10 (10.0.1.10) 56(64) bytes of data:
64 bytes from 10.0.1.10: icmp_seq=1 ttl=63 time=2.70 ms

--- 10.0.1.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.700/2.700/2.700/0.000 ms
mininet>
```

```
H1 - S1-ETH1 - S1:
# Src => Dst Prot Type Data ID Seq Chksum
1 10.0.1.10 => 10.0.1.10 ICMP RQST 1234567 1 123 0xcfc45
2 10.0.1.10 => 10.0.0.10 ICMP RPLY 1234567
3 10.0.1.10 => 10.0.1.10 GRE ICMP 538 0x75bcd15 123 0xcfc45
4 10.0.1.10 => 10.0.0.10 ICMP RPLY 538

S1 - S1-ETH2 - H2:
# Src => Dst Prot Type Data ID Seq Chksum
1 10.0.1.10 => 10.0.1.10 ICMP RQST 1234567
2 10.0.1.10 => 10.0.0.10 ICMP RPLY 1234567
3 10.0.1.10 => 10.0.1.10 ICMP RQST 538
4 10.0.1.10 => 10.0.0.10 ICMP RPLY 538
```


Results

- Concepts work²
 - Packets accepted only with correct key
 - Sequence number correctly checked & updated
 - Multiple session IDs and keys supported simultaneously

²although using an extremely weak form of MAC

Results

- Concepts work²
 - Packets accepted only with correct key
 - Sequence number correctly checked & updated
 - Multiple session IDs and keys supported simultaneously
- P4 language and software targets still work in progress
 - Problems with dropping traffic
 - Register operations not yet in specification
 - Key length supported?

²although using an extremely weak form of MAC

Conclusion

- Authentication with P4 is feasible
 - But requires new P4 features and target support
 - Keep authentication scheme & P4 program simple

Conclusion

- Authentication with P4 is feasible
 - But requires new P4 features and target support
 - Keep authentication scheme & P4 program simple
- Lots of caveats
 - Target limitations; cryptographic algorithms; NAT; IP fragmentation; packet forwarding mode (cut-through); sequence number synchronisation; asymmetric flows

Conclusion

- Authentication with P4 is feasible
 - But requires new P4 features and target support
 - Keep authentication scheme & P4 program simple
- Lots of caveats
 - Target limitations; cryptographic algorithms; NAT; IP fragmentation; packet forwarding mode (cut-through); sequence number synchronisation; asymmetric flows
- Where to go from here?
 - Add cryptographic means to P4
 - Further design CoCo architecture & authentication scheme
 - Implement in P4, controller & client
 - End-to-end authentication & encryption?

Questions?

