
# Fondamenti di Internet

ACL and AAA Principles and Configuration



Prof.Gianluca Reali

# Foreword

- Security is a key issue in current networks and a basic related knowledge is essential
- Access control lists (ACLs) are closely related to network security and QoS. By accurately identifying packet flows on a network and working with other technologies, ACLs can control network access behaviors, prevent network attacks, and improve network bandwidth utilization, thereby ensuring network environment security and QoS reliability.
- Authentication, authorization, and accounting (AAA) is a management framework that provides a security mechanism for authorizing some users to access specified resources and recording the operations of these users. AAA is widely used because of its good scalability and easy implementation of centralized management of user information. AAA can be implemented through multiple protocols. Currently, the Remote Authentication Dial-In User Service (RADIUS) protocol is the most commonly used to implement AAA.

- Note:
  - The implementation of ACLs varies with vendors. This course describes the ACL technology implemented on Huawei devices.
  - A local area network (LAN) is a computer network that connects computers in a limited area, such as a residential area, a school, a lab, a college campus, or an office building.

# Objectives

- On completion of this course, you will be able to:
  - Understand basic principle on network security
  - Describe the basic principles and functions of ACLs and AAA
  - Understand the types and characteristics of ACLs.
  - Describe the basic composition of ACLs and ACL rule ID matching order.
  - Describe the application scenarios of AAA.
  - Understand the fundamentals of RADIUS.
  - Get familiar with the basic configurations of ACLs and AAA

# Contents


1. **Security Background**
2. ACL Overview
3. Basic Concepts and Working Mechanism of ACLs
4. Basic Configurations and Applications of ACLs
5. AAA Overview
6. AAA Configuration

# What is network security?

*confidentiality*: only sender, intended receiver should "understand" message contents

- sender encrypts message
- receiver decrypts message

*authentication:* sender, receiver want to confirm identity of each other

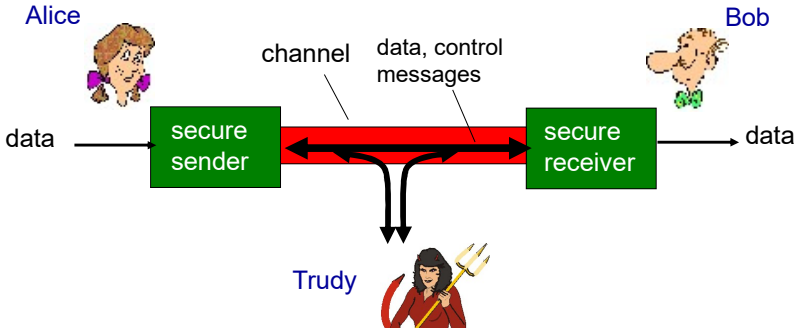
*message integrity:* sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

*access and availability*: services must be accessible and available to users

Page 4

# Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice want to communicate“securely”
- Trudy (intruder) may intercept, delete, add messages




Page 5

# There are bad guys out there!

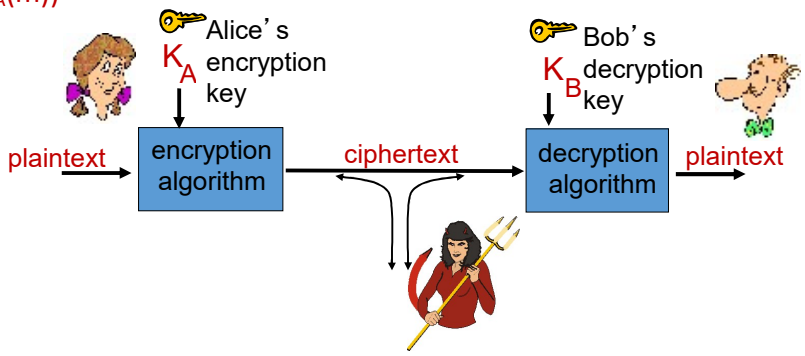
- *eavesdrop:* intercept messages
- actively *insert* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

# The language of cryptography

m plaintext message

$K_A(m)$ ciphertext, encrypted with key $K_A$

$m = K_B(K_A(m))$

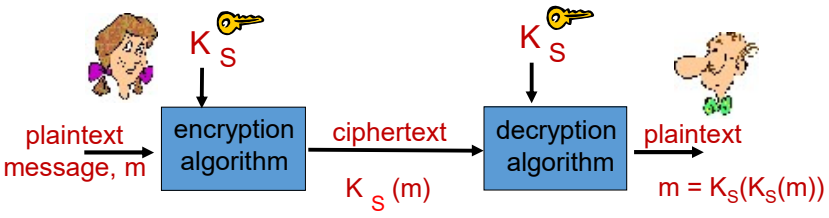


# Symmetric key cryptography

symmetric key crypto: Bob and Alice share same (symmetric) key: $K_S$

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

*Q:* how do Bob and Alice agree on key value?



# Simple encryption scheme

*substitution cipher:* substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:  abcdefghijklmnopqrstuvwxyz
                ↓                   ↓
ciphertext: mnbvcxzasdfghjklpoiuytrewq
```

e.g.:

```
Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

*Encryption key:* mapping from set of 26 letters to set of 26 letters

# Symmetric key crypto: DES

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- how secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase  decrypted (brute force) in less than a day
  - no known good analytic attack
- making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

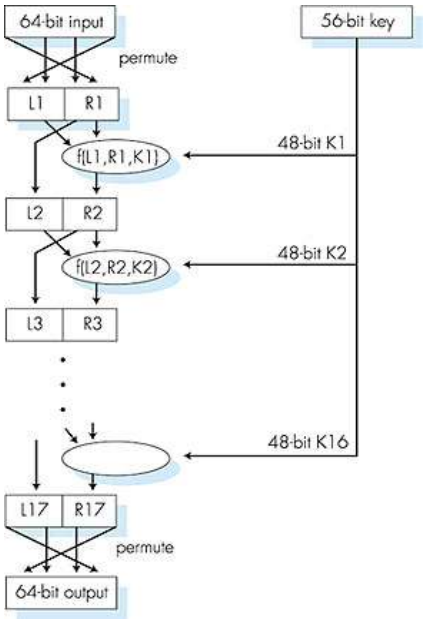
# Symmetric key crypto: DES

*DES operation*

initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

final permutation



# AES: Advanced Encryption Standard

- symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

# Public Key Cryptography

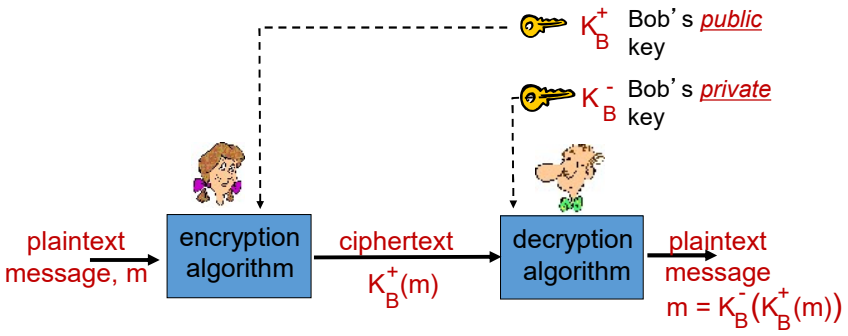
*symmetric key crypto*

- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

*public key crypto*

- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

# Public key cryptography



# Public key encryption algorithms

requirements:

1. need $K_B^+(\ )$ and $K_B^-(\ )$ such that

$$K_B^-(K_B^+(m)) = m$$

2. given public key $K_B^+$, it should be impossible to compute private key $K_B^-$

*RSA:* Rivest, Shamir, Adelson algorithm

# Prerequisite: modular arithmetic

- x mod n = remainder of x when divide by n
- facts:

  [(a mod n) + (b mod n)] mod n = (a+b) mod n

  proof: (a mod n)=a'; (b mod n)=b';

  (a+b mod n)=(kn+a'+hn+b')mod n=(a'+b')mod n

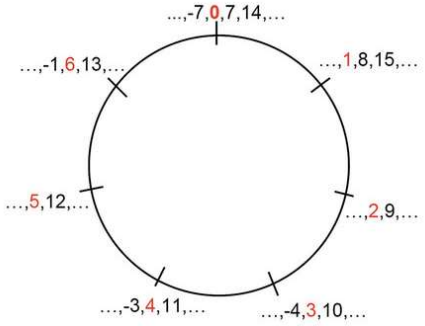
  [(a mod n) - (b mod n)] mod n = (a-b) mod n

  [(a mod n) * (b mod n)] mod n = (a*b) mod n
- thus

  $(a \bmod n)^d \bmod n = a^d \bmod n$
- example: x=14, n=10, d=2:

  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$

  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$




Modulo 7

# RSA: getting ready

- message: just a bit pattern
- bit pattern can be uniquely represented by an integer number
- thus, encrypting a message is equivalent to encrypting a number.

*example:*

- m= 10010001 . This message is uniquely represented by the decimal number 145.
- to encrypt m, we encrypt the corresponding number, which gives a new number (the ciphertext).

# RSA: Creating public/private key pair

1. choose two large prime numbers $p$, $q$. (e.g., 1024 bits each)
2. compute $n = pq$, $z = (p-1)(q-1)$
3. choose $e$ (with $e<n$) that has no common factors with z ($e$, $z$ are "relatively prime").
4. choose $d$ such that $ed-1$ is exactly divisible by $z$. (in other words: $ed$ mod $z = 1$ ).
5. *public* key is $(n,e)$. *private* key is $(n,d)$.

$K_B^+$ $K_B^-$

# RSA: encryption, decryption

0. given ($n$,$e$) and ($n$,$d$) as computed above

1. to encrypt message $m$ ($<n$), compute
   $c = m^e \bmod n$

2. to decrypt received bit pattern, $c$, compute
   $m = c^d \bmod n$

*magic happens!* $m = \underbrace{(m^e \bmod n)}_{c}{}^d \bmod n$

# Why does RSA work?

- must show that $c^d \bmod n = m$

  where $c = m^e \bmod n$

- fact: for any x and y: $x^y \bmod n = x^{(y \bmod z)} \bmod n$
  - Where n=pq, and p and q are prime numbers and z = (p-1)(q-1)
- thus,

  $c^d \bmod n = (m^e \bmod n)^d \bmod n$

  $= m^{ed} \bmod n$

  $= m^{(ed \bmod z)} \bmod n$

  $= m^1 \bmod n$

  $= m$

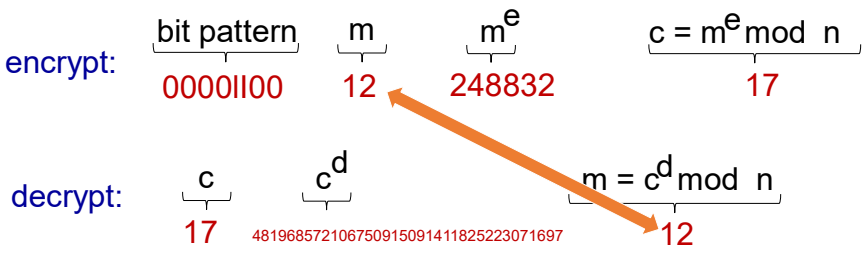
# RSA example:

Bob chooses *p=5, q=7*. Then *n=35, z=24*.

*e=5* (so *e, z* relatively prime).

*d=29* (so *ed-1* exactly divisible by z).

encrypting 8-bit messages.



# RSA: another important property

The following property will be *very* useful later:

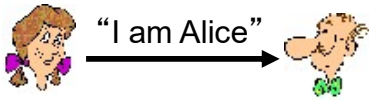
$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

*result is the same!*

# Authentication

*Goal:* Bob wants Alice to "prove" her identity to him

*Protocol ap1.0:* Alice says "I am Alice"

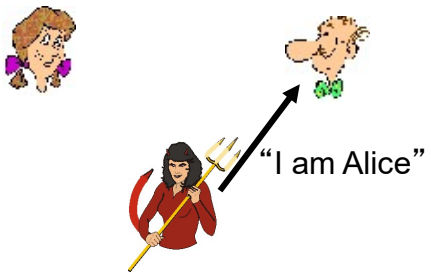



Failure scenario??

# Authentication

*Goal:* Bob wants Alice to "prove" her identity to him

*Protocol ap1.0:* Alice says "I am Alice"




in a network,
Bob can not "see" Alice,
so Trudy simply declares
herself to be Alice

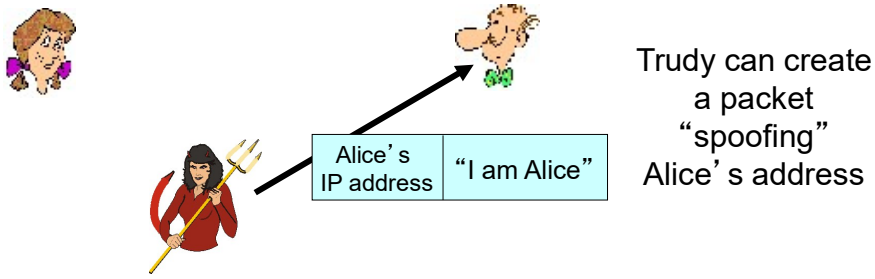
# Authentication: another try

*Protocol ap2.0:* Alice says "I am Alice" in an IP packet containing her source IP address


Failure scenario??

# Authentication: another try

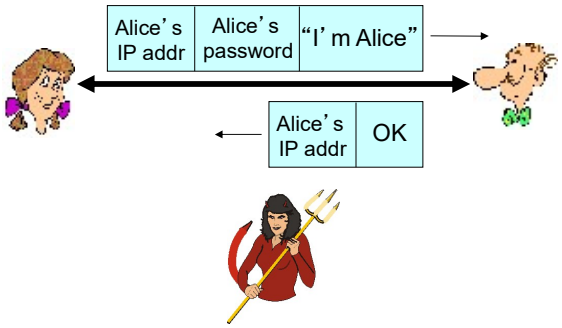
*Protocol ap2.0:* Alice says "I am Alice" in an IP packet containing her source IP address




Trudy can create a packet "spoofing" Alice's address

# Authentication: another try

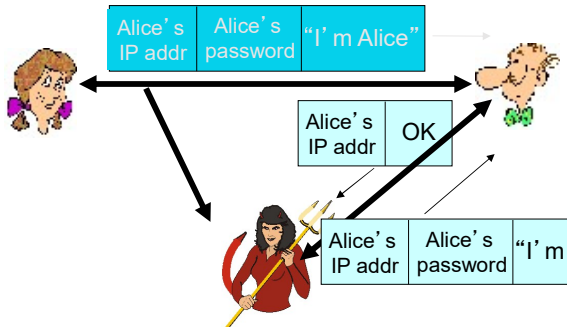
*Protocol ap3.0:* Alice says "I am Alice" and sends her secret password to "prove" it.




Failure scenario??

# Authentication: another try

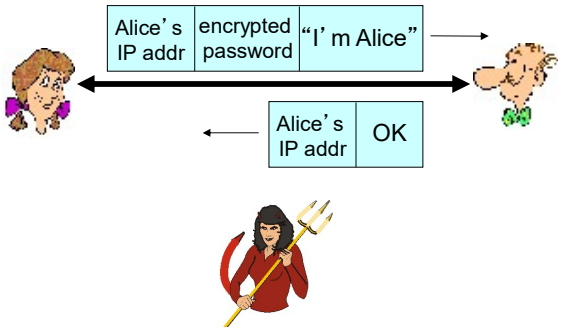
*Protocol ap3.0:* Alice says "I am Alice" and sends her secret password to "prove" it.




*playback attack:* Trudy records Alice's packet and later plays it back to Bob

# Authentication: yet another try

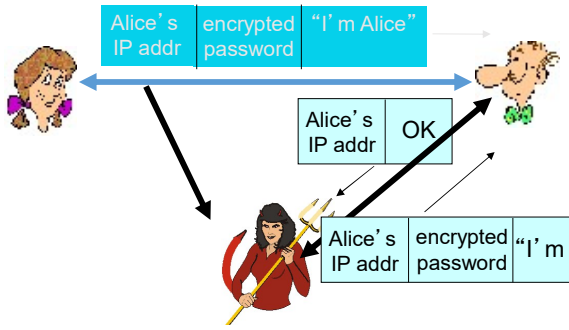
*Protocol ap3.1:* Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.




Failure scenario??

# Authentication: yet another try

*Protocol ap3.1:* Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



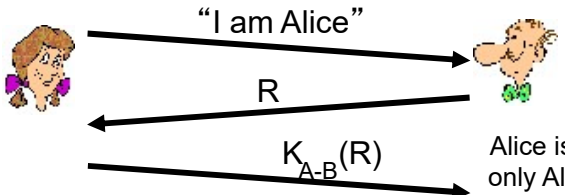

record and playback *still* works!

# Authentication: yet another try

*Goal:* avoid playback attack

*nonce:* number (R) used only *once-in-a-lifetime*

*ap4.0:* to prove Alice "live", Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key




Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

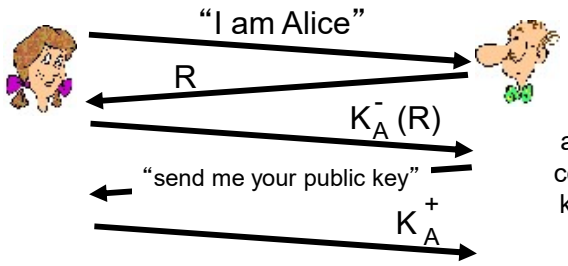
Failures, drawbacks?

# Authentication: ap5.0

ap4.0 requires shared symmetric key

- can we authenticate using public key techniques?

*ap5.0:* use nonce, public key cryptography

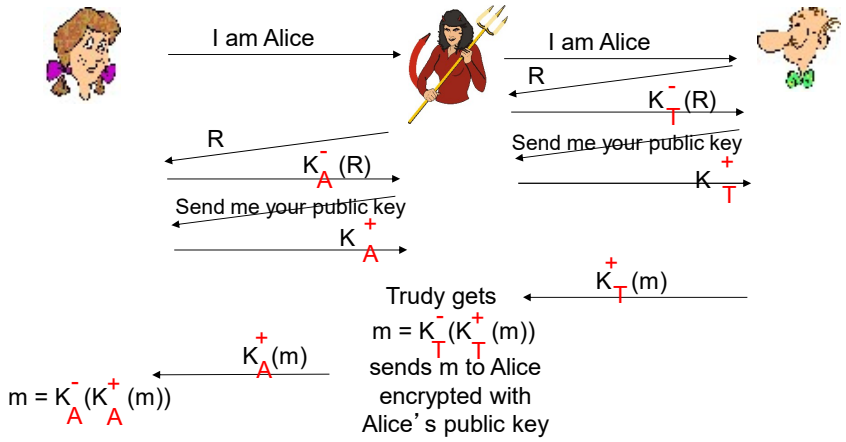



Bob computes
$K_A^+(K_A^-(R)) = R$
and knows only Alice could have the private key, that encrypted R such that
$K_A^+(K_A^-(R)) = R$

# ap5.0: security hole

*man (or woman) in the middle attack:* Trudy poses as Alice

(to Bob) and as Bob (to Alice)



# Digital signatures

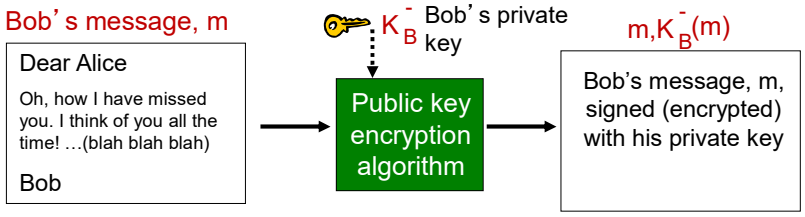
cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- *verifiable, nonforgeable:* recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

# Digital signatures

simple digital signature for message m:

- Bob signs m by encrypting with his private key $K_B^-$, creating "signed" message, $K_B^-(m)$



# Digital signatures

- suppose Alice receives msg m, with signature: m, $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key $K_B^+$ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

Alice thus verifies that:

- ✓ Bob signed m
- ✓ no one else signed m
- ✓ Bob signed m and not m'

non-repudiation:

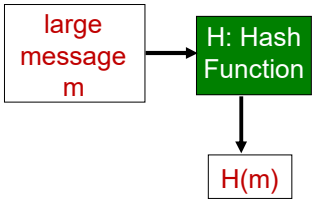
- ✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m

# Message digests

computationally expensive to public-key-encrypt long messages

*goal:* fixed-length, easy- to-compute digital "fingerprint"

- apply hash function H to *m*, get fixed size message digest, *H(m).*




Hash function properties:

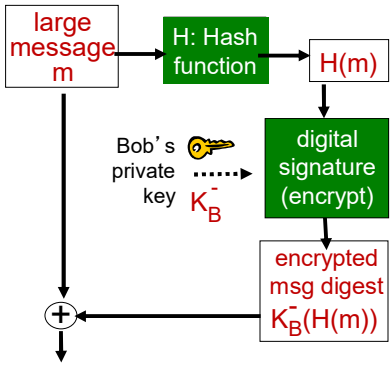
- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x, computationally infeasible to find m such that x = H(m)

# Hash function algorithms

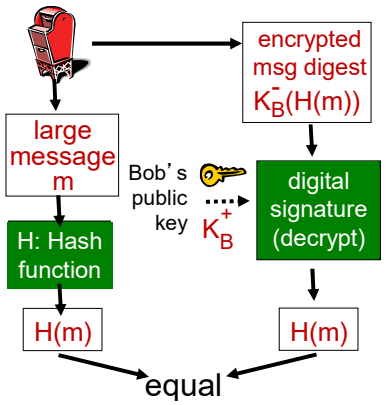
- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x
- SHA-1 is also used
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

# Digital signature = signed message digest

Bob sends digitally signed message:

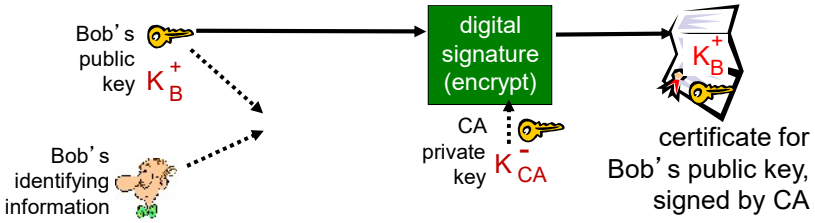



Alice verifies signature, integrity of digitally signed message:



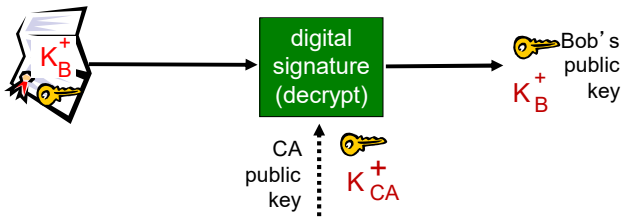
# Public-key certification: Certification authorities

- *certification authority (CA):* binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA – CA says "this is E's public key"



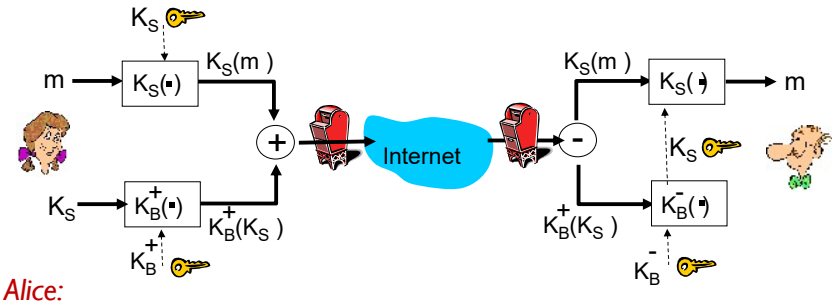
# Certification authorities

- when Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key




# Secure e-mail

- Alice wants to send confidential e-mail, m, to Bob.



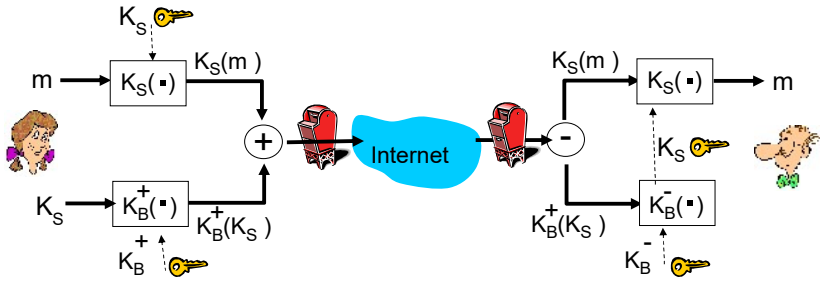

*Alice:*

- generates random *symmetric* private key, $K_S$
- encrypts message with $K_S$ (for efficiency)
- also encrypts $K_S$ with Bob's public key
- sends both $K_S(m)$ and $K_B(K_S)$ to Bob

Page 42

# Secure e-mail

- Alice wants to send confidential e-mail, m, to Bob.



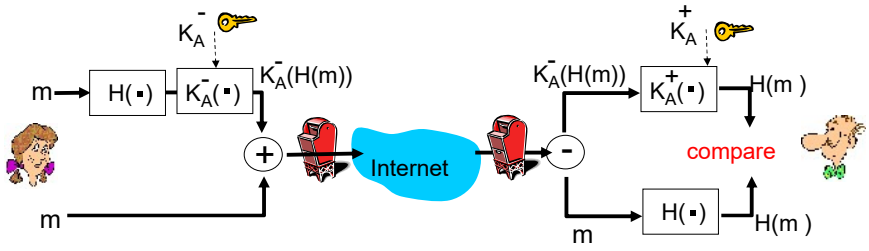

*Bob:*

- uses his private key to decrypt and recover $K_S$
- uses $K_S$ to decrypt $K_S(m)$ to recover m

Page 43

# Secure e-mail (continued)

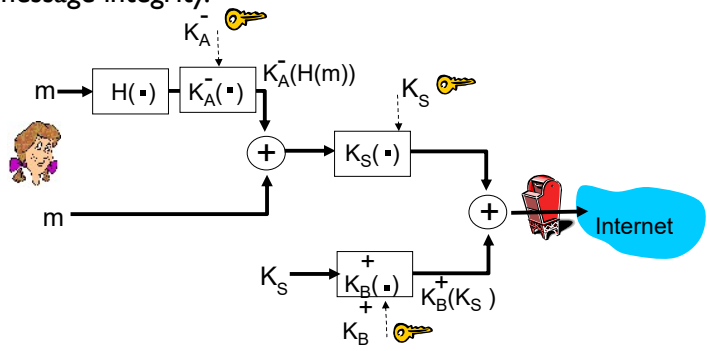
- Alice wants to provide sender authentication message integrity




- Alice digitally signs message
- sends both message (in the clear) and digital signature

# Secure e-mail (continued)

- Alice wants to provide secrecy, sender authentication, message integrity.



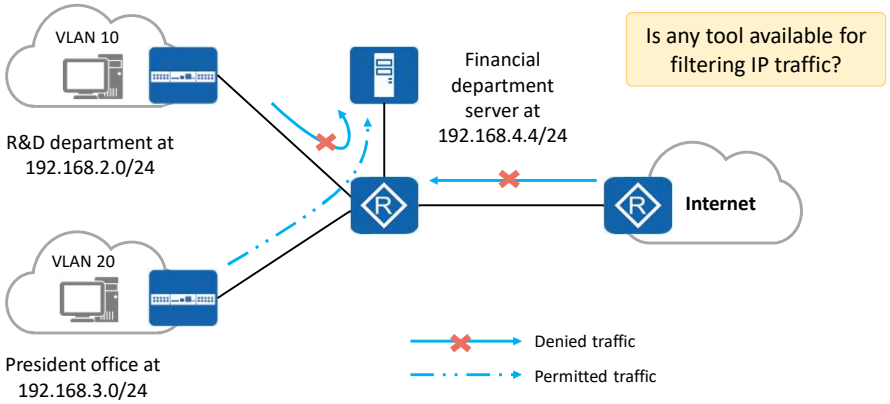

*Alice uses three keys:* her private key, Bob's public key, newly created symmetric key

Page 45

# Contents

1. Security Background
2. **ACL Overview**
3. Basic Concepts and Working Mechanism of ACLs
4. Basic Configurations and Applications of ACLs
5. AAA Overview
6. AAA Configuration

# Background: A Tool is Required to Filter Traffic




- To ensure financial data security, an enterprise prohibits the R&D department's access to the financial department server but allows the president office's access to the financial department server.

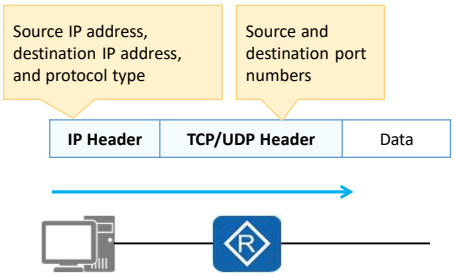
 

- Rapid network development brings the following issues to network security and QoS:
  - Resources on the key servers of an enterprise are obtained without permission, and confidential information of the enterprise leaks, causing a potential security risk to the enterprise.
  - The virus on the Internet spreads to the enterprise intranet, threatening intranet security.
  - Network bandwidth is occupied by services randomly, and bandwidth for delay-sensitive services such as voice and video cannot be guaranteed, lowering user experience.
- These issues seriously affect network communication, so network security and QoS need to be improved urgently. For example, a tool is required to filter traffic.

- https://support.huawei.com/enterprise/it/doc/EDOC1100086647 for more information

# ACL Overview

- An ACL is a set of sequential rules composed of permit or deny statements.
- An ACL matches and distinguishes packets.




**ACL Application**

- Matching IP traffic
- Invoked in a traffic filter
- Invoked in network address translation (NAT)
- Invoked in a routing policy
- Invoked in a firewall policy
- Invoked in QoS
- Others



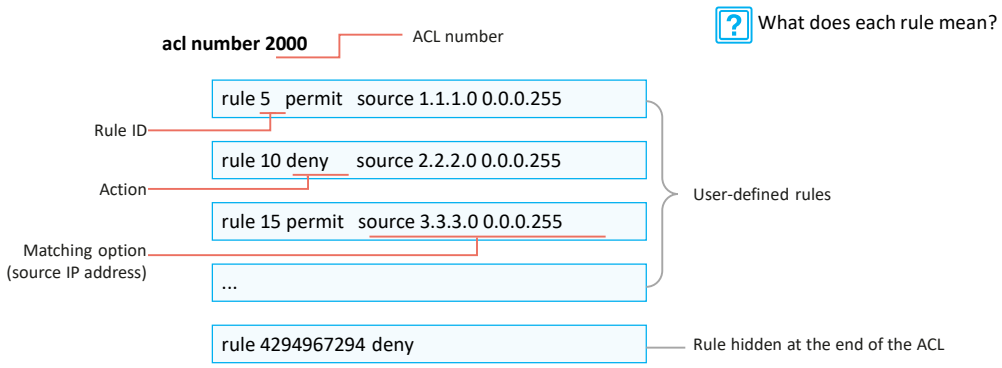

- ACLs accurately identify and control packets on a network to manage network access behaviors, prevent network attacks, and improve bandwidth utilization. In this way, ACLs ensure security and QoS.
  - An ACL is a set of sequential rules composed of permit or deny statements. It classifies packets by matching fields in packets.
  - An ACL can match elements such as source and destination IP addresses, source and destination port numbers, and protocol types in IP datagrams. It can also match routes.
- In this course, traffic filtering is used to describe ACLs.

# Contents

1. Security Background
2. ACL Overview
3. **Basic Concepts and Working Mechanism of ACLs**
4. Basic Configurations and Applications of ACLs
5. AAA Overview
6. AAA Configuration

# ACL Composition

- An ACL consists of several permit or deny statements. Each statement is a rule of the ACL, and permit or deny in each statement is the action corresponding to the rule.



- ACL composition:
  - ACL number: An ACL is identified by an ACL number. Each ACL needs to allocated an ACL number. The ACL number range varies according to the ACL type, which will be described later.
  - Rule: As mentioned above, an ACL consists of several permit/deny statements, and each statement is a rule of the ACL.
  - Rule ID: Each ACL rule has an ID, which identifies the rule. Rule IDs can be manually defined or automatically allocated by the system. A rule ID ranges from 0 to 4294967294. All rules are arranged in the ascending order of rule ID.
  - Action: Each rule contains a permit or deny action. ACLs are usually used together with other technologies, and the meanings of the permit and deny actions may vary according to scenarios.
    - For example, if an ACL is used together with traffic filtering technology (that is, the ACL is invoked in traffic filtering), the permit action allows traffic to pass and the deny action rejects traffic.
  - Matching option: ACLs support various matching options. In this example, the matching option is a source IP address. The ACL also supports other matching options, such as Layer 2 Ethernet frame header information (including source and destination MAC addresses and Ethernet frame protocol type), Layer 3 packet information (including destination address and protocol type), and Layer 4 packet information (including TCP/UDP port number).
- Question: What does rule 5 permit source 1.1.1.0 0.0.0.255 mean? This will be introduced later.

# Rule ID

```
acl number 2000
                Rule ID
 rule    5      deny      source  10.1.1.1  0
 rule    10     deny      source  10.1.1.2  0
 rule    15     permit    source  10.1.1.0  0.0.0.255
```

Step = 5

How do I add a rule?

**rule 11 deny source 10.1.1.3 0**

```
acl number 2000
 rule    5      deny      source  10.1.1.1  0
 rule    10     deny      source  10.1.1.2  0
 rule    11     deny      source  10.1.1.3  0
 rule    15     permit    source  10.1.1.0  0.0.0.255
```

## Rule ID and Step

- **Rule ID**
  Each rule in an ACL has an ID.
- **Step**
  A step is an increment between neighboring rule IDs automatically allocated by the system. The default step is 5. Setting a step facilitates rule insertion between existing rules of an ACL.
- **Rule ID allocation**
  If a rule is added to an empty ACL but no ID is manually specified for the rule, the system allocates a step value (5 for example) as the ID of the rule. If an ACL contains rules with manually specified IDs and a rule with no manually specified ID is added, the system allocates to this rule an ID that is greater than the largest rule ID in the ACL and is the smallest integer multiple of the step value.

- Rule ID and step:
  - Rule ID: Each ACL rule has an ID, which identifies the rule. Rule IDs can be manually defined or automatically allocated by the system.
  - Step: When the system automatically allocates IDs to ACL rules, the increment between neighboring rule IDs is called a step. The default step is 5. Therefore, rule IDs are 5, 10, 15, and so on.
    - If a rule is manually added to an ACL but no ID is specified, the system allocates to this rule an ID that is greater than the largest rule ID in the ACL and is the smallest integer multiple of the step value.
    - The step can be changed. For example, if the step is changed to 2, the system automatically renumbers the rule IDs as 2, 4, 6...
- What is the function of a step? Why can't rules 1, 2, 3, and 4 be directly used?
  - First, let's look at a question. How do I add a rule?
  - We can manually add rule 11 between rules 10 and 15.
  - Therefore, setting a step of a certain length facilitates rule insertion between existing rules.

# Wildcard (1)

acl number 2000

Wildcard

| | | | | | |
|---|---|---|---|---|---|
| rule | 5 | deny | source | 10.1.1.1 | 0 |
| rule | 10 | deny | source | 10.1.1.2 | 0 |
| rule | 15 | permit | source | 10.1.1.0 | 0.0.0.255 |

**Wildcard**

- A wildcard is a 32-bit number that indicates which bits in an IP address need to be strictly matched and which bits do not need to be matched.
- A wildcard is usually expressed in dotted decimal notation, as a network mask is expressed. However, their meanings are different.

- Matching rule

  0: matching; 1: random allocation

How do I match the network segment address corresponding to 192.168.1.1/24?

| | | | | |
|---|---|---|---|---|
| 192.168.1.1 | 11000000 | 10101000 | 00000001 | 00000001 |
| 0.0.0.255 | 00000000 | 00000000 | 00000000 | 11111111 |
| | Strict matching | | | Random allocation |

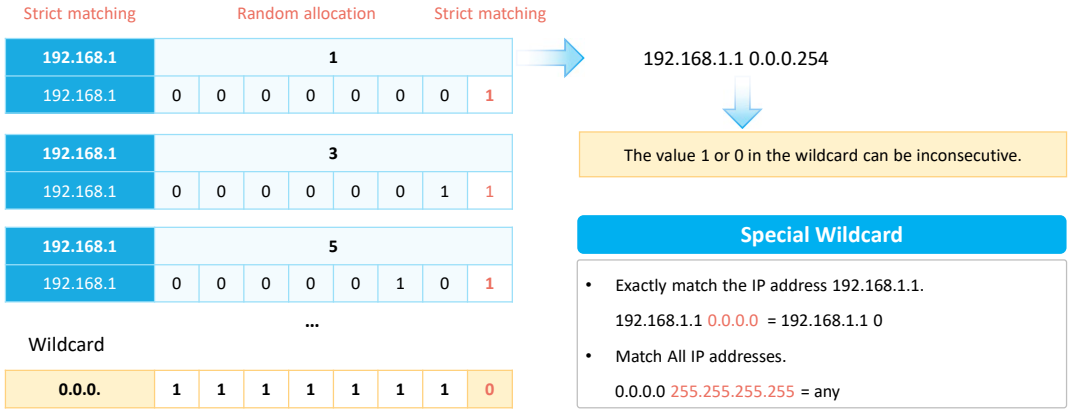
192.168.1.0/24 network segment




- When an IP address is matched, a 32-bit mask is followed. The 32-bit mask is called a wildcard.
- A wildcard is also expressed in dotted decimal notation. After the value is converted to a binary number, the value 0 indicates that the equivalent bit must match and the value 1 indicates that the equivalent bit does not matter.
- Let's look at two rules:
  - rule 5: denies the packets with the source IP address 10.1.1.1. Because the wildcard comprises all 0s, each bit must be strictly matched. Specifically, the host IP address 10.1.1.1 is matched.
  - rule 15: permits the packets with the source IP address on the network segment 10.1.1.0/24. The wildcard is 0.0.0.11111111, and the last eight bits are 1s, indicating that the bits do not matter. Therefore, the last eight bits of 10.1.1.xxxxxxxx can be any value, and the 10.1.1.0/24 network segment is matched.
- For example, if we want to exactly match the network segment address corresponding to 192.168.1.1/24, what is the wildcard?
  - It can be concluded that the network bits must be strictly matched and the host bits do not matter. Therefore, the wildcard is 0.0.0.255.

# Wildcard (2)

- A wildcard can be used to match odd IP addresses in the network segment 192.168.1.0/24, such as 192.168.1.1, 192.168.1.3, and 192.168.1.5.



- How do I set the wildcard to match the odd IP addresses in the network segment 192.168.1.0/24?
  - First, let's look at the odd IP addresses, such as 192.168.1.1, 192.168.1.5, and 192.168.1.11.
  - After the last eight bits are converted into binary numbers, the corresponding addresses are 192.168.1.00000001, 192.168.1.00000101, and 192.168.1.00001011.
  - We can see the common points. The seven most significant bits of the last eight bits can be any value, and the least significant bit is fixed to 1. Therefore, the answer is 192.168.1.1 0.0.0.254 (0.0.0.11111110).
- In conclusion, 1 or 0 in a wildcard can be inconsecutive.

- There are two special wildcards.
  - If a wildcard comprising all 0s is used to match an IP address, the address is exactly matched.
  - If a wildcard comprising all 1s is used to match 0.0.0.0, all IP addresses are matched.

# ACL Classification and Identification

- ACL classification based on ACL rule definition methods

| Category | Number Range | Description |
|---|---|---|
| Basic ACL | 2000 to 2999 | Defines rules based on source IPv4 addresses, fragmentation information, and effective time ranges. |
| Advanced ACL | 3000 to 3999 | Defines rules based on source and destination IPv4 addresses, IPv4 protocol types, ICMP types, TCP source/destination port numbers, UDP source/destination port numbers, and effective time ranges. |
| Layer 2 ACL | 4000 to 4999 | Defines rules based on information in Ethernet frame headers of packets, such as source and destination MAC addresses and Layer 2 protocol types. |
| User-defined ACL | 5000 to 5999 | Defines rules based on packet headers, offsets, character string masks, and user-defined character strings. |
| User ACL | 6000 to 6999 | Defines rules based on source IPv4 addresses or user control list (UCL) groups, destination IPv4 addresses or destination UCL groups, IPv4 protocol types, ICMP types, TCP source/destination port numbers, and UDP source/destination port numbers. |

- ACL classification based on ACL identification methods

| Category | Description |
|---|---|
| Numbered ACL | Traditional ACL identification method. A numbered ACL is identified by a number. |
| Named ACL | A named ACL is identified by a name. |

- Based on ACL rule definition methods, ACLs can be classified into the following types:
  - Basic ACL, advanced ACL, Layer 2 ACL, user-defined ACL, and user ACL
    - A user-defined ACL defines rules based on packet headers, offsets, character string masks, and user-defined character strings. With such a user-defined ACL configured, the system performs an AND operation on the packet bytes from a certain position behind the packet header and the character string mask, compares the extracted character string against the user-defined character string, and then filters IPv4 and IPv6 packets.
    - Compared with basic ACL, advanced ACL, and Layer 2 ACL, user-defined ACL is more accurate, flexible, and provides more functions. For example, if you want to filter ARP packets based on source IP addresses and ARP packet types, you can configure a user-defined ACL.
    - By default, an ACL always takes effect after it is applied to a service module. To make ACL rules work only in a certain period, you can define a time range and associate it with the ACL rules. In this way, services can be controlled through a time-based ACL. For example, with a time-based ACL, an enterprise can forbid employees to access the Internet during work hours and limit bandwidth for bandwidth-consuming services such as P2P and downloading services during peak hours to avoid network congestion.
    - Time ranges associated with ACL rules are classified into:
    - Periodic time range: defines a time range by week. The associated ACL rules take

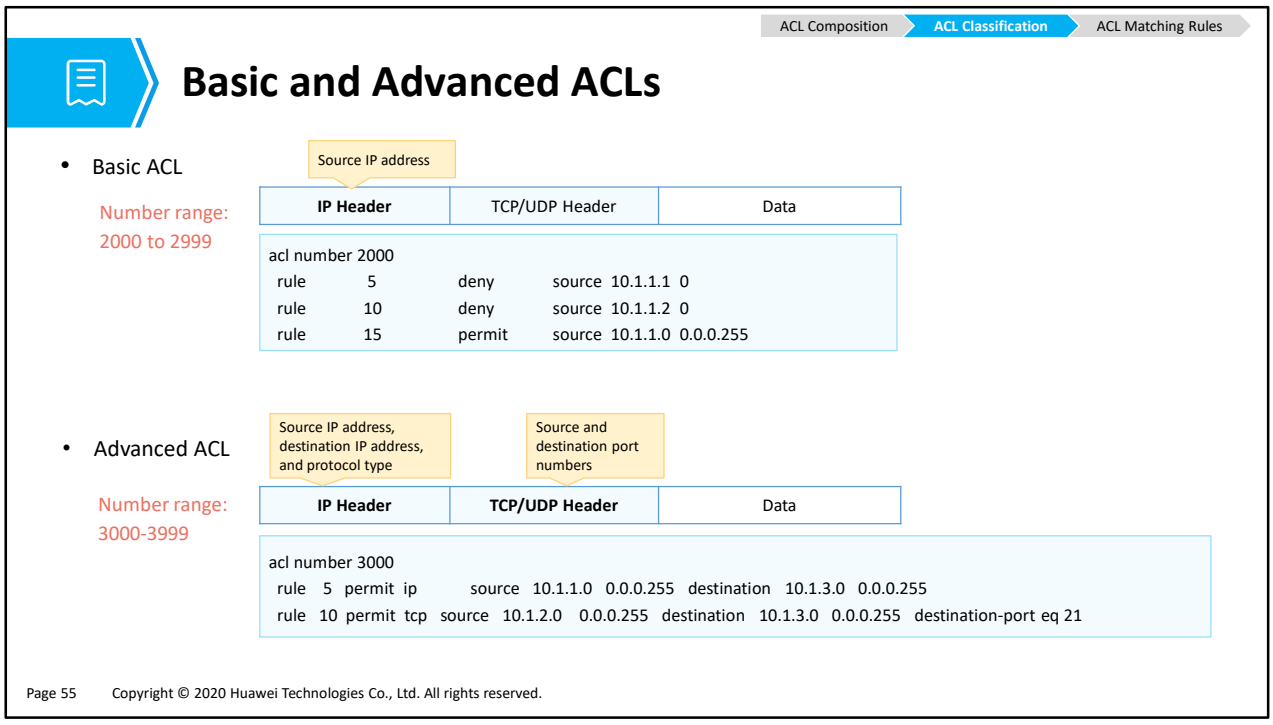
effect at an interval of one week. For example, if the time range of ACL rules is 8:00-12:00 on Monday, the ACL rules take effect at 8:00-12:00 on every Monday.

- Absolute time range: defines a time range from YYYY/MM/DD hh:mm to YYYY/MM/DD hh:mm. The associated ACL rules take effect only in this period.
- A UCL group identifies user category created using the **ucl-group** command.

Each department has a large number of terminals and the terminals of the same department are located on different network segments. The workload of configuring network access policy for the terminals one by one is huge. Therefore, configure the UCL group to classify the terminals into different types, and associate a user ACL with the UCL group so that the terminals in each group can share the ACL rules. The workload of administrator is reduced, and ACL resource use efficiency on the device is improved.

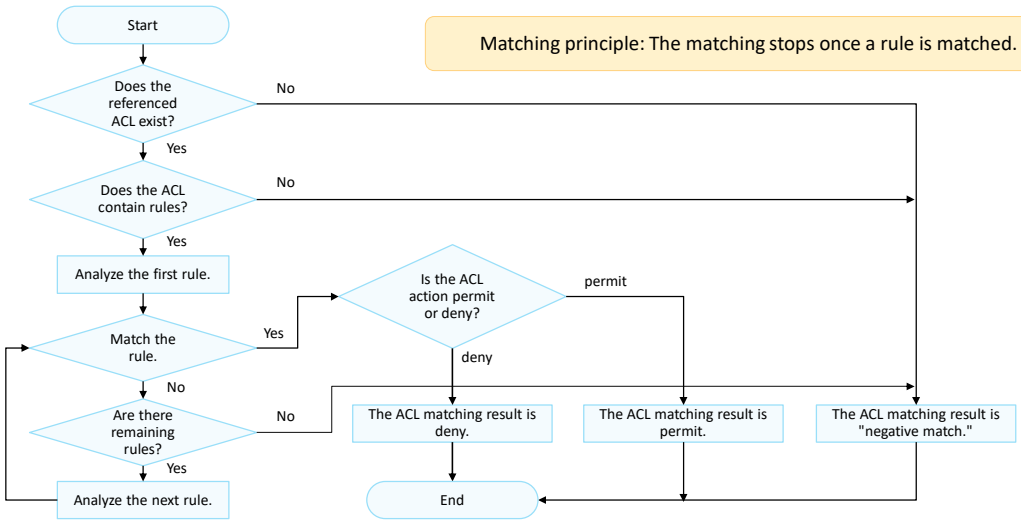
- Based on ACL identification methods, ACLs can be classified into the following types:
  - Numbered ACL and named ACL

- Note: You can specify a number for an ACL. The ACLs of different types have different number ranges. You can also specify a name for an ACL to help you remember the ACL's purpose. A named ACL consists of a name and number. That is, you can specify an ACL number when you define an ACL name. If you do not specify a number for a named ACL, the system automatically allocates a number to it.




- Basic ACL:
  - A basic ACL is used to match the source IP address of an IP packet. The number of a basic ACL ranges from 2000 to 2999.
  - In this example, ACL 2000 is created. This ACL is a basic ACL.
- Advanced ACL:
  - An advanced ACL can be matched based on elements such as the source IP address, destination IP address, protocol type, and TCP or UDP source and destination port numbers in an IP packet. A basic ACL can be regarded as a subset of an advanced ACL. Compared with a basic ACL, an advanced ACL defines more accurate, complex, and flexible rules.

# ACL Matching Mechanism



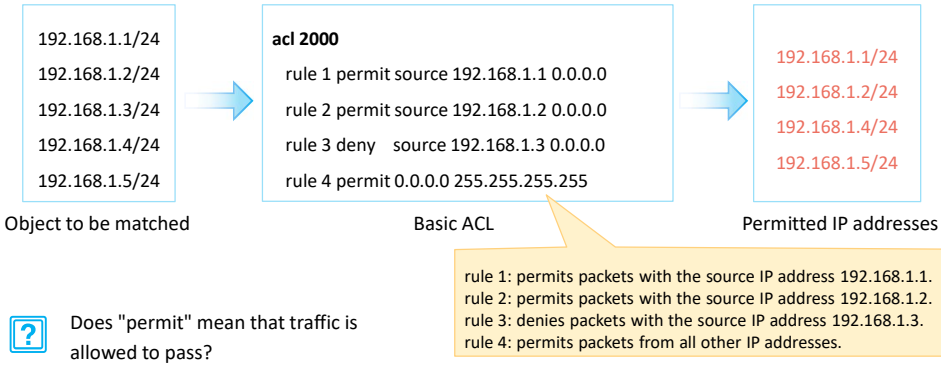

 

- The ACL matching mechanism is as follows:
  - After receiving a packet, the device configured with an ACL matches the packet against ACL rules one by one. If the packet does not match any ACL rule, the device attempts to match the packet against the next ACL rule.
  - If the packet matches an ACL rule, the device performs the action defined in the rule and stops the matching.
- Matching process: The device checks whether an ACL is configured.
  - If no ACL is configured, the device returns the result "negative match."
  - If an ACL is configured, the device checks whether the ACL contains rules.
    - If the ACL does not contain rules, the device returns the result "negative match."
    - If the ACL contains rules, the device matches the packet against the rules in ascending order of rule ID.
      - If the packet matches a permit rule, the device stops matching and returns the result "positive match (permit)."
      - If the packet matches a deny rule, the device stops matching and returns the result "positive match (deny)."
      - If the packet does not match any rule in the ACL, the device returns the result "negative match.“
- The ACL matching results include "positive match" and "negative match."

    - Positive match: Packets match a rule in an ACL. The result is "positive match" regardless of whether packets match a permit or deny rule in an ACL.
    - Negative match: No ACL exists, the ACL does not contain rules, or packets do not match any rule in an ACL.
- Matching principle: The matching stops once a rule is matched.

# ACL Matching Order and Result

- Configuration order (config mode)
  - The system matches packets against ACL rules in ascending order of rule ID. That is, the rule with the smallest ID is processed first.



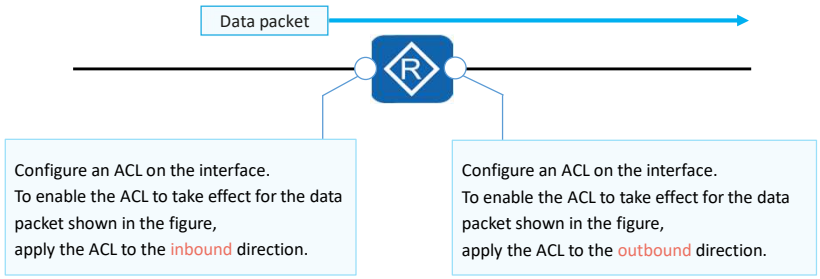
- An ACL can consist of multiple deny or permit statements. Each statement describes a rule. Rules may overlap or conflict. Therefore, the ACL matching order is very important.
- Huawei devices support two matching orders: automatic order (auto) and configuration order (config). The default matching order is config.
  - auto: The system arranges rules according to the precision of the rules ("depth first" principle), and matches packets against the rules in descending order of precision. —This is complicated and is not detailed here. If you are interested in it, you can view related materials after class.
  - config: The system matches packets against ACL rules in ascending order of rule ID. That is, the rule with the smallest ID is processed first. —This is the matching order mentioned above.
    - If another rule is added, the rule is added to the corresponding position, and packets are still matched in ascending order.
- Matching result:
  - First, let's understand the meaning of ACL 2000.
    - rule 1: permits packets with the source IP address 192.168.1.1.
    - rule 2: permits packets with the source IP address 192.168.1.2.
    - rule 3: denies packets with the source IP address 192.168.1.3.
    - rule 4: permits packets from all other IP addresses.
  - When packets with the source IP address 192.168.1.3 pass through the device configured

with the ACL:

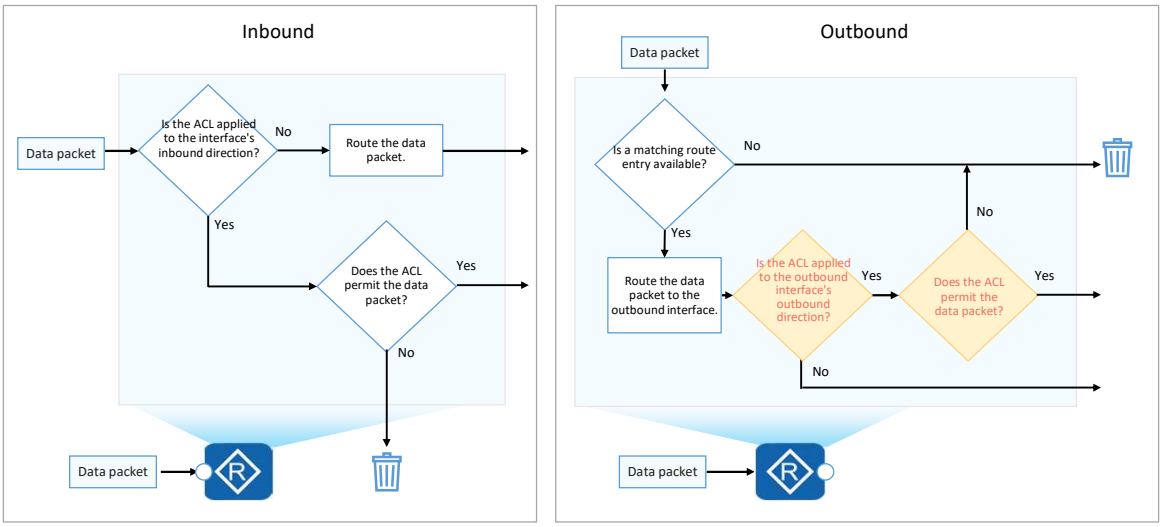
  - The device matches the packets against rule 1. The matching result is "negative match."
  - The device continues to match the packets against rule 2. The matching result is still "negative match."
  - The device continues to match the packets against rule 3. The matching result is "positive match," and the action is deny.

- Note: ACLs are usually used together with other technologies, and the meanings of the permit and deny actions may vary according to scenarios. For example, if an ACL is used together with traffic filtering technology (that is, the ACL is invoked in traffic filtering), the permit action allows traffic to pass and the deny action rejects traffic.
- https://support.huawei.com/enterprise/en/knowledge/KB1000076061/?idAbsPath=24030814%7C21432787%7C7923148%7C9858988%7C16527

# ACL Matching Position



# Inbound and Outbound Directions



# Contents

1. Security Background
2. ACL Overview
3. Basic Concepts and Working Mechanism of ACLs
4. **Basic Configurations and Applications of ACLs**
5. AAA Overview
6. AAA Configuration

# Basic Configuration Commands of Basic ACLs

1. Create a basic ACL.

   [Huawei] **acl** [ **number** ] *acl-number* [ **match-order config** ]

   Create a numbered basic ACL and enter its view.

   [Huawei] **acl name** *acl-name* { **basic** | *acl-number* } [ **match-order config** ]

   Create a named basic ACL and enter its view.

2. Configure a rule for the basic ACL.

   [Huawei-acl-basic-2000] **rule** [ *rule-id* ] { **deny** | **permit** } [ **source** { *source-address source-wildcard* | **any** } | **time-range** *time-name* ]

   In the basic ACL view, you can run this command to configure a rule for the basic ACL.

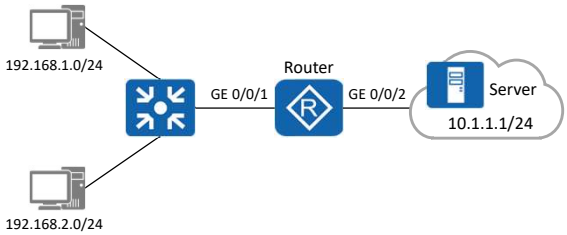
 

- Create a basic ACL.
- [Huawei] **acl** [ **number** ] *acl-number* [ **match-order config** ]
  - *acl-number*: specifies the number of an ACL.
  - **match-order config**: indicates the matching order of ACL rules. config indicates the configuration order.
- [Huawei] **acl name** *acl-name* { **basic** | *acl-number* } [ **match-order config** ]
  - *acl-name*: specifies the name of an ACL.
  - **basic**: indicates a basic ACL.
- Configure a rule for the basic ACL.
- [Huawei-acl-basic-2000] **rule** [ *rule-id* ] { **deny** | **permit** } [ **source** { *source-address source-wildcard* | **any** } | **time-range** *time-name* ]
  - *rule-id*: specifies the ID of an ACL rule.
  - **deny**: denies the packets that match the rule.
  - **permit**: permits the packets that match the rule.
  - **source** { *source-address source-wildcard* | **any** }: specifies the source IP address of packets that match the ACL rule. If no source address is specified, packets with any source addresses are matched.
    - *source-address*: specifies the source IP address of packets.

- *source-wildcard*: specifies the wildcard of the source IP address.
- **any**: indicates any source IP address of packets. That is, the value of source-address is 0.0.0.0 or the value of source-wildcard is 255.255.255.255.

- **time-range** *time-name*: specifies a time range in which the ACL rule takes effect. time-name specifies the name of a time range. If no time range is specified, the ACL rule is always valid.

# Case: Use a Basic ACL to Filter Data Traffic




- Requirements:

  To prevent the user host on the network segment 192.168.1.0/24 from accessing the network where the server resides, configure a basic ACL on the router. After the configuration is complete, the ACL filters out the data packets whose source IP addresses are on the network segment 192.168.1.0/24 and permits other data packets.

1. Configure IP addresses and routes on the router.

2. Create a basic ACL on the router to prevent the network segment 192.168.1.0/24 from accessing the network where the server resides.

```
[Router] acl 2000
[Router-acl-basic-2000] rule deny source 192.168.1.0  0.0.0.255
[Router-acl-basic-2000] rule permit source any
```

3. Configure traffic filtering in the inbound direction of GE 0/0/1.

```
[Router] interface GigabitEthernet 0/0/1
[Router-GigabitEthernet0/0/1] traffic-filter inbound acl 2000
[Router-GigabitEthernet0/0/1] quit
```

- Configuration roadmap:
  - Configure a basic ACL and traffic filtering to filter packets from a specified network segment.
- Procedure:
  1. Configure IP addresses and routes on the router.
  2. Create ACL 2000 and configure ACL rules to deny packets from the network segment 192.168.1.0/24 and permit packets from other network segments.
  3. Configure traffic filtering.
- Note:
  - The **traffic-filter** command applies an ACL to an interface to filter packets on the interface.
  - Command format: **traffic-filter** { **inbound** | **outbound** } **acl** { *acl-number* | **name** *acl-name* }
    - **inbound**: configures ACL-based packet filtering in the inbound direction of an interface.
    - **outbound**: configures ACL-based packet filtering in the outbound direction of an interface.
    - **acl**: filters packets based on an IPv4 ACL.

# Basic Configuration Commands of Advanced ACLs (1)

1. Create an advanced ACL.

   [Huawei] **acl** [ **number** ] *acl-number* [ **match-order config** ]

   Create a numbered advanced ACL and enter its view.

   [Huawei] **acl name** *acl-name* { **advance** | *acl-number* } [ **match-order config** ]

   Create a named advanced ACL and enter its view.

- Create an advanced ACL.
- [Huawei] **acl** [ **number** ] *acl-number* [ **match-order config** ]
  - *acl-number*: specifies the number of an ACL.
  - **match-order config**: indicates the matching order of ACL rules. config indicates the configuration order.
- [Huawei] **acl name** *acl-name* { **advance** | *acl-number* } [ **match-order config** ]
  - *acl-name*: specifies the name of an ACL.
  - **advance**: indicates an advanced ACL.

# Basic Configuration Commands of Advanced ACLs (2)

2. Configure a rule for the advanced ACL.
   You can configure advanced ACL rules according to the protocol types of IP packets. The parameters vary according to the protocol types.

   ▫ When the protocol type is IP, the command format is:

   **rule** [ *rule-id* ] { **deny** | **permit** } **ip** [ **destination** { *destination-address destination-wildcard* | **any** } | **source** { *source-address source-wildcard* | **any** } | **time-range** *time-name* | [ **dscp** *dscp* | [ **tos** *tos* | **precedence** *precedence* ] ] ]

   In the advanced ACL view, you can run this command to configure a rule for the advanced ACL.

   ▫ When the protocol type is TCP, the command format is:

   **rule** [ *rule-id* ] { **deny** | **permit** } { *protocol-number* | **tcp** } [ **destination** { *destination-address destination-wildcard* | **any** } | **destination-port** { **eq** port | **gt** port | **lt** port | **range** port-start port-end } | **source** { *source-address source-wildcard* | **any** } | **source-port** { **eq** port | **gt** port | **lt** port | **range** port-start port-end } | **tcp-flag** { **ack** | **fin** | **syn** } * | **time-range** time-name ] *

   In the advanced ACL view, you can run this command to configure a rule for the advanced ACL.

- Configure a rule for the advanced ACL.
- When the protocol type is IP:
  - **rule** [ *rule-id* ] { **deny** | **permit** } **ip** [ **destination** { *destination-address destination-wildcard* | **any** } | **source** { *source-address source-wildcard* | **any** } | **time-range** *time-name* | [ **dscp** *dscp* | [ **tos** *tos* | **precedence** *precedence* ] ] ]
    - **ip**: indicates that the protocol type is IP.
    - **destination** { *destination-address destination-wildcard* | any }: specifies the destination IP address of packets that match the ACL rule. If no destination address is specified, packets with any destination addresses are matched.
    - **dscp** *dscp*: specifies the differentiated services code point (DSCP) of packets that match the ACL rule. The value ranges from 0 to 63.
    - **tos** *tos*: specifies the ToS of packets that match the ACL rule. The value ranges from 0 to 15.
    - **precedence** *precedence*: specifies the precedence of packets that match the ACL rule. The value ranges from 0 to 7.

- When the protocol type is TCP:
  - **rule** [ *rule-id* ] { **deny** | **permit** } { *protocol-number* | **tcp** } [ **destination** { *destination-address destination-wildcard* | **any** } | **destination-port** { **eq** *port* | **gt** *port* | **lt** *port* | **range** *port-start port-end* } | **source** { *source-address source-wildcard* | **any** } | **source-port** { **eq**

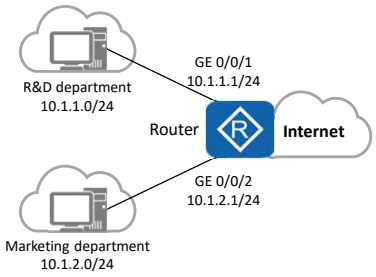
*port* | **gt** *port* | **lt** *port* | **range** *port-start port-end* } | **tcp-flag** { **ack** | **fin** | **syn** } * | **time-range** *time-name* ] *

- **tcp**: indicates that the protocol type is TCP. You can set protocol-number to 6 to indicate TCP.
- **destination-port** { **eq** *port* | **gt** *port* | **lt** *port* | **range** *port-start port-end* }: specifies the TCP destination port number of packets that match the ACL rule. The value is valid only when the protocol type is TCP. If no destination port number is specified, packets with any TCP destination port numbers are matched.
  - **eq** *port*: equal to the destination port number
  - **gt** *port*: greater than the destination port number
  - **lt** *port*: less than the destination port number
  - **range** *port-start port-end*: specifies a source port number range.
- **tcp-flag**: indicates the SYN Flag in the TCP packet header.

- Bits 012345 67

  DSCP ECN
- Bits 0 1 2 3 4 5 6 7
- Precedence Type of Service

## Case: Use Advanced ACLs to Prevent User Hosts on Different Network Segments from Communicating (1)




Requirements:

- The departments of a company are connected through the router. To facilitate network management, the administrator allocates IP addresses of different network segments to the R&D and marketing departments.
- The company requires that the router prevent the user hosts on different network segments from communicating to ensure information security.

1. Configure IP addresses and routes on the router.

2. Create ACL 3001 and configure rules for the ACL to deny packets from the R&D department to the marketing department.

```
[Router] acl 3001
[Router-acl-adv-3001] rule deny ip source 10.1.1.0 0.0.0.255 destination
10.1.2.0 0.0.0.255
[Router-acl-adv-3001] quit
```

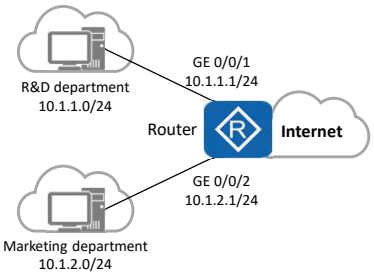
3. Create ACL 3002 and configure rules for the ACL to deny packets from the marketing department to the R&D department.

```
[Router] acl 3002
[Router-acl-adv-3002] rule deny ip source 10.1.2.0 0.0.0.255 destination
10.1.1.0 0.0.0.255
[Router-acl-adv-3002] quit
```

- Configuration roadmap:
  - Configure an advanced ACL and traffic filtering to filter the packets exchanged between the R&D and marketing departments.
- Procedure:
  1. Configure IP addresses and routes on the router.
  2. Create ACL 3001 and configure rules for the ACL to deny packets from the R&D department to the marketing department.
  3. Create ACL 3002 and configure rules for the ACL to deny packets from the marketing department to the R&D department.

## Case: Use Advanced ACLs to Prevent User Hosts on Different Network Segments from Communicating (2)




Requirements:

- The departments of a company are connected through the router. To facilitate network management, the administrator allocates IP addresses of different network segments to the R&D and marketing departments.
- The company requires that the router prevent the user hosts on different network segments from communicating to ensure information security.

4. Configure traffic filtering in the inbound direction of GE 0/0/1 and GE 0/0/2.

```
[Router] interface GigabitEthernet 0/0/1
[Router-GigabitEthernet0/0/1] traffic-filter inbound acl 3001
[Router-GigabitEthernet0/0/1] quit

[Router] interface GigabitEthernet 0/0/2
[Router-GigabitEthernet0/0/2] traffic-filter inbound acl 3002
[Router-GigabitEthernet0/0/2] quit
```

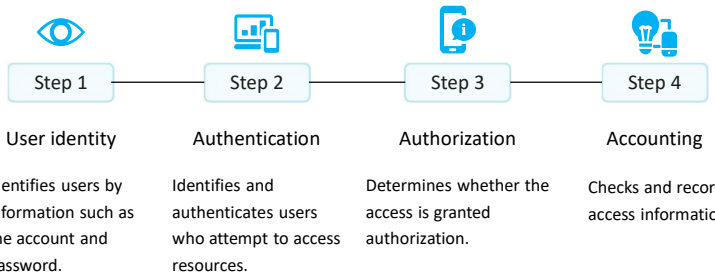
- Procedure:
    4. Configure traffic filtering in the inbound direction of GE 0/0/1 and GE 0/0/2.
- Note:
    - The **traffic-filter** command applies an ACL to an interface to filter packets on the interface.
    - Command format: **traffic-filter** { **inbound** | **outbound** } **acl** { *acl-number* | **name** *acl-name* }
        - **inbound**: configures ACL-based packet filtering in the inbound direction of an interface.
        - **outbound**: configures ACL-based packet filtering in the outbound direction of an interface.
        - **acl**: filters packets based on an IPv4 ACL.

# Contents

1. Security Background
2. ACL Overview
3. Basic Concepts and Working Mechanism of ACLs
4. Basic Configurations and Applications of ACLs
5. **AAA Overview**
6. AAA Configuration

# Basic Concepts of AAA

- Authentication, authorization, and accounting (AAA) provides a management mechanism for network security.

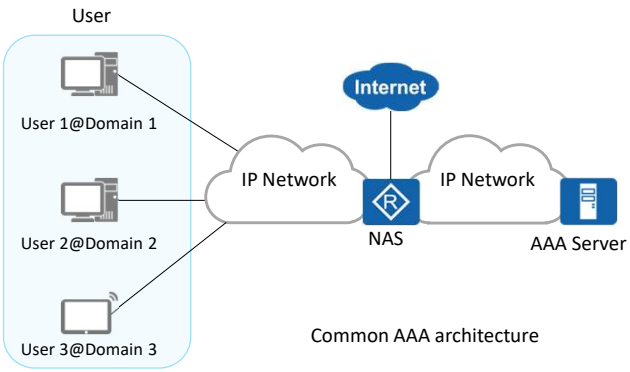


- Authentication: determines which users can access the network.
- Authorization: authorizes users to access specific services.
- Accounting: records network resource utilization.
- The Internet service provider (ISP) needs to authenticate the account and password of a home broadband user before allowing the user to access the Internet. In addition, the ISP records the online duration or traffic of the user. This is the most common application scenario of the AAA technology.

# Common AAA Architecture

- A common AAA architecture includes the user, network access server (NAS), and AAA server.




Common AAA architecture

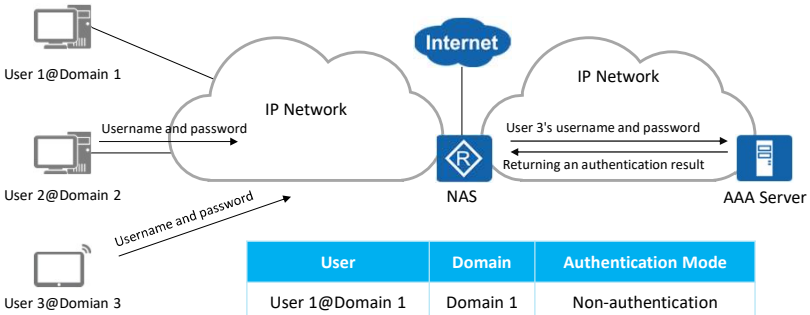
- The NAS collects and manages **user access requests** in a centralized manner.
- Multiple domains are created on the NAS to manage users. Different domains can be associated with different AAA schemes, which include the **authentication scheme**, **authorization scheme**, and **accounting scheme**.
- When receiving a user access request, the NAS determines the domain to which the user belongs based on the username and performs user management and control based on the AAA schemes configured for the domain.




- The NAS manages users based on domains. Each domain can be configured with different authentication, authorization, and accounting schemes to perform authentication, authorization, and accounting for users in the domain.
- Each user belongs to a domain. The domain to which a user belongs is determined by the character string following the domain name delimiter @ in the user name. For example, if the user name is user 1@domain 1, the user belongs to domain 1. If the user name does not end with @, the user belongs to the default domain.

# Authentication

- AAA supports the following authentication modes: non-authentication, local authentication, and remote authentication.



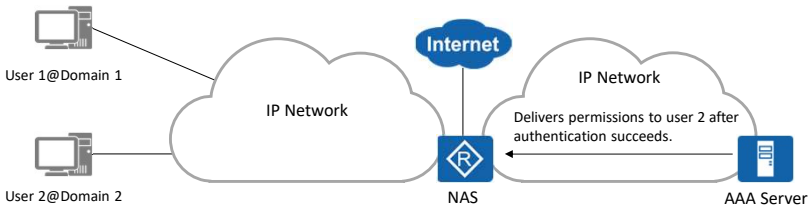

| User | Domain | Authentication Mode |
|---|---|---|
| User 1@Domain 1 | Domain 1 | Non-authentication |
| User 2@Domain 2 | Domain 2 | Local authentication |
| User 3@Domain 3 | Domain 3 | Remote authentication |




- AAA supports three authentication modes:
  - Non-authentication: Users are fully trusted and their identities are not checked. This authentication mode is seldom used for security purposes.
  - Local authentication: Local user information (including the username, password, and attributes) is configured on the NAS. In this case, the NAS functions as the AAA server. Local authentication features fast processing and low operational costs. The disadvantage is that the amount of stored information is limited by device hardware. This authentication mode is often used to manage login users, such as Telnet and FTP users.
  - Remote authentication: User information (including the username, password, and attributes) is configured on the authentication server. Remote authentication can be implemented through RADIUS or HWTACACS. The NAS functions as a client to communicate with the RADIUS or HWTACACS server.

# Authorization

- AAA supports the following authorization modes: non-authorization, local authorization, and remote authorization.
- Authorization information includes the user group, VLAN ID, and ACL number.



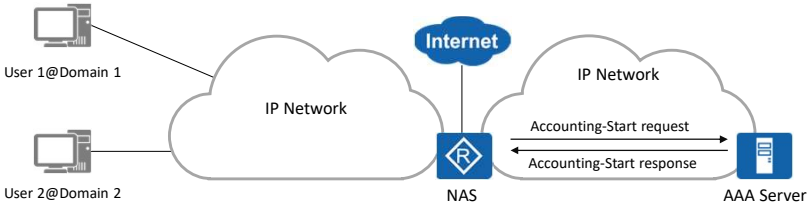

| User | Domain | Authorization Mode | Authorization Content |
|---|---|---|---|
| User 1@Domain 1 | Domain 1 | Non-authorization | None |
| User 2@Domain 2 | Domain 2 | Local authorization | Internet access is allowed. |
| User 3@Domain 3 | Domain 3 | Remote authorization | Authorization is granted by a remote server. |




- The AAA authorization function grants users the permission to access specific networks or devices. AAA supports the following authorization modes:
  - Non-authorization: Authenticated users have unrestricted access rights on a network.
  - Local authorization: Users are authorized based on the domain configuration on the NAS.
  - Remote authorization: The RADIUS or HWTACACS server authorizes users.
    - In HWTACACS authorization, all users can be authorized by the HWTACACS server.
    - RADIUS authorization applies only to the users authenticated by the RADIUS server. RADIUS integrates authentication and authorization. Therefore, RADIUS authorization cannot be performed singly.
- When remote authorization is used, users can obtain authorization information from both the authorization server and NAS. The priority of the authorization information configured on the NAS is lower than that delivered by the authorization server.

# Accouting

- The accounting function monitors the network behavior and network resource utilization of authorized users.
- AAA supports two accounting modes: non-accounting and remote accounting.



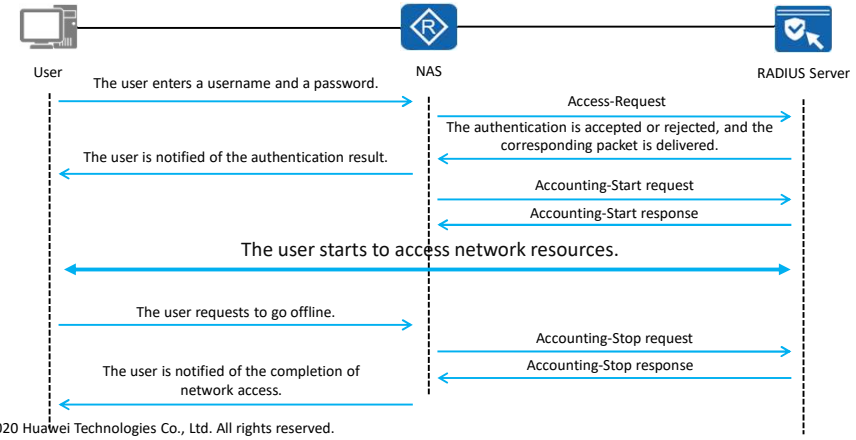

| User | Domain | Accounting Mode |
|---|---|---|
| User 1@Domain 1 | Domain 1 | Non-accounting |
| User 2@Domain 2 | Domain 2 | Non-accounting |
| User 3@Domain 3 | Domain 3 | Remote accounting |




- AAA supports the following accounting modes:
  - Non-accounting: Users can access the Internet for free, and no activity log is generated.
  - Remote accounting: Remote accounting is performed through the RADIUS server or HWTACACS server.

# AAA Implementation Protocol - RADIUS

- Of the protocols that are used to implement AAA, RADIUS is the most commonly used. RADIUS uses the User Datagram Protocol (UDP) as the transmission protocol and uses UDP ports 1812 and 1813 as the authentication and accounting ports, respectively.



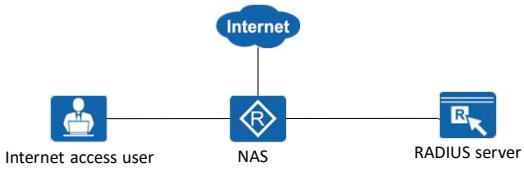

 

- Of the protocols that are used to implement AAA, RADIUS is the most commonly used. RADIUS is a distributed information exchange protocol based on the client/server structure. It implements user authentication, accounting, and authorization.
- Generally, the NAS functions as a RADIUS client to transmit user information to a specified RADIUS server and performs operations (for example, accepting or rejecting user access) based on the information returned by the RADIUS server.
- RADIUS servers run on central computers and workstations to maintain user authentication and network service access information. The servers receive connection requests from users, authenticate the users, and send the responses (indicating that the requests are accepted or rejected) to the clients. RADIUS uses the User Datagram Protocol (UDP) as the transmission protocol and uses UDP ports 1812 and 1813 as the authentication and accounting ports, respectively. RADIUS features high real-time performance. In addition, the retransmission mechanism and standby server mechanism are also supported, providing good reliability.
- The message exchange process between the RADIUS server and client is as follows:
    1. When a user accesses the network, the user initiates a connection request and sends the username and password to the RADIUS client (NAS).
    2. The RADIUS client sends an authentication request packet containing the username and password to the RADIUS server.
    3. If the request is valid, the RADIUS server completes authentication and sends the required authorization information to the RADIUS client. If the request is invalid, the RADIUS server sends the authorization failure information to the RADIUS client.

4. The RADIUS client notifies the user of whether authentication is successful.
5. The RADIUS client permits or rejects the user according to the authentication result. If the user is permitted, the RADIUS client sends an Accounting-Request (Start) packet to the RADIUS server.
6. The RADIUS server sends an Accounting-Response (Start) packet to the RADIUS client and starts accounting.
7. The user starts to access network resources.
8. When a user does not want to access network resources, the user sends a logout request to stop accessing network resources.
9. The RADIUS client sends an Accounting-Request (Stop) packet to the RADIUS server.
10. The RADIUS server sends an Accounting-Response (Stop) packet to the RADIUS client and stops accounting.
11. The RADIUS client notifies the user of the processing result, and the user stops accessing network resources.

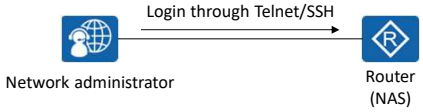
# Common AAA Application Scenarios

## AAA for Internet Access Users Through RADIUS




- AAA schemes are configured on the NAS to implement interworking between the NAS and RADIUS server.
- After the user enters a username and a password on the client, the NAS sends the username and password to the RADIUS server for authentication.
- If the authentication succeeds, the user is granted the Internet access permission.
- The RADIUS server can record the user's network resource utilization during Internet access.

## Local Authentication and Authorization for Administrative Users




- After local AAA schemes are configured on Router, Router compares the username and password of the network administrator with the locally configured username and password when the network administrator logs in to Router.
- After the authentication succeeds, Router grants certain administrator permissions to the network administrator.

# Contents

1. Security Background
2. ACL Overview
3. Basic Concepts and Working Mechanism of ACLs
4. Basic Configurations and Applications of ACLs
5. AAA Overview
6. **AAA Configuration**

# AAA Configuration (1)

1. Enter the AAA view.

   [Huawei] **aaa**

   Exit the system view and enter the AAA view.

2. Create an authentication scheme.

   [Huawei-aaa] **authentication-scheme** *authentication-scheme-name*

   Create an authentication scheme and enter the authentication scheme view.

   [Huawei-aaa-authentication-scheme-name] **authentication-mode** { **hwtacacs** | **local** | **radius** }

   Set the authentication mode to local authentication. By default, the authentication mode is local authentication.

- The **authorization-scheme** *authorization-scheme-name* command configures an authorization scheme for a domain. By default, no authorization scheme is applied to a domain.
- The **authentication-mode** { **hwtacacs** | **local** | **radius** } command configures an authentication mode for the current authentication scheme. By default, local authentication is used.

- Huawei (HW) Terminal Access Controller Access Control System (HWTACACS) is an enhanced security protocol based on TACACS (RFC 1492). HWTACACS is similar to RADIUS, and uses a client/server model for information exchange between the NAS and the HWTACACS server. Uses TCP, which provides reliable network transmission.

# AAA Configuration (2)

3. Create a domain and bind an authentication scheme to the domain.

   [Huawei-aaa] **domain** *domain-name*

   Create a domain and enter the domain view.

   [Huawei-aaa-domain-name] **authentication-scheme** *authentication-scheme-name*

   Bind the authentication scheme to the domain.

4. Create a user.

   [Huawei-aaa] **local-user** *user-name* **password cipher** *password*

   Create a local user and configure a password for the local user.

   - If the username contains a delimiter "@", the character before "@" is the username and the character after "@" is the domain name.
   - If the value does not contain "@", the entire character string represents the username and the domain name is the default one.

# AAA Configuration (3)

5. Configure a user access type.

   [Huawei-aaa] **local-user user-name service-type { { terminal | telnet | ftp | ssh | snmp | http } | ppp | none }**

   Configure the access type of the local user. By default, all access types are disabled for a local user.

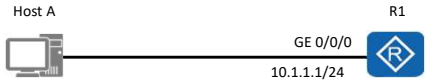
6. Configure a user level.

   [Huawei-aaa] **local-user** *user-name* **privilege level** *level*

   Specify the permission level of the local user.

# AAA Configuration Examples

- After a user password and a user level are configured on R1, host A can use the configured username and password to remotely log in to R1.




```
[R1]aaa
[R1-aaa]local-user huawei password cipher huawei123
[R1-aaa]local-user huawei service-type telnet
[R1-aaa]local-user huawei privilege level 0
[R1]user-interface vty 0 4
[R1-ui-vty0-4]authentication-mode aaa
```

# Configuration Verification (1)

- In AAA, each domain is associated with an authentication scheme, an authorization scheme, and an accounting scheme. In this example, the default domain is used.

```
[R1]display domain name default_admin
  Domain-name:                  default_admin
  Domain-state:                 Active
  Authentication-scheme-name:   default
  Accounting-scheme-name:       default
  Authorization-scheme-name:    -
  Service-scheme-name:   -
  RADIUS-server-template:       -
  HWTACACS-server-template:     -
  User-group:                   -
```

- The **display domain** [ **name** *domain-name* ]command displays the configuration of a domain.
- If the value of **Domain-state** is Active, the domain is activated.
- If the username does not end with @, the user belongs to the default domain. Huawei devices support the following default domains:
  - The **default** domain is for common users.
  - The **default_admin** domain is the default domain for administrators.

# Configuration Verification (2)

- After the user properly logs in and logs out, you can view the user record.

```
[R1]display aaa offline-record all
  ------------------------------------------------------------------------------
  User name:              huawei
  Domain name:            default_admin
  User MAC:               00e0-fc12-3456
  User access type:       telnet
  User IP address:        10.1.1.2
  User ID:                1
  User login time:        2019/12/28 17:59:10
  User offline time:      2019/12/28 18:00:04
  User offline reason:    user request to offline
```

- The **display aaa offline-record** command displays user offline records.
- After a STA goes offline, you can use this command to check the reason why the STA goes offline.