Powered by TeNet

SNMP – v3

Prof. T.A. Gonsalves
TeNeT Group
Dept of CSE, IIT-Madras

SNMP v3

RFC 2271 Architecture for describing SNMP mgmt framework

RFC 2272 Message Processing and Dispatching for SNMP

RFC 2273 SNMP v3 applications

RFC 2274 User-based Security Model

RFC 2275 View based Access Control Model for SNMP

SNMPv3

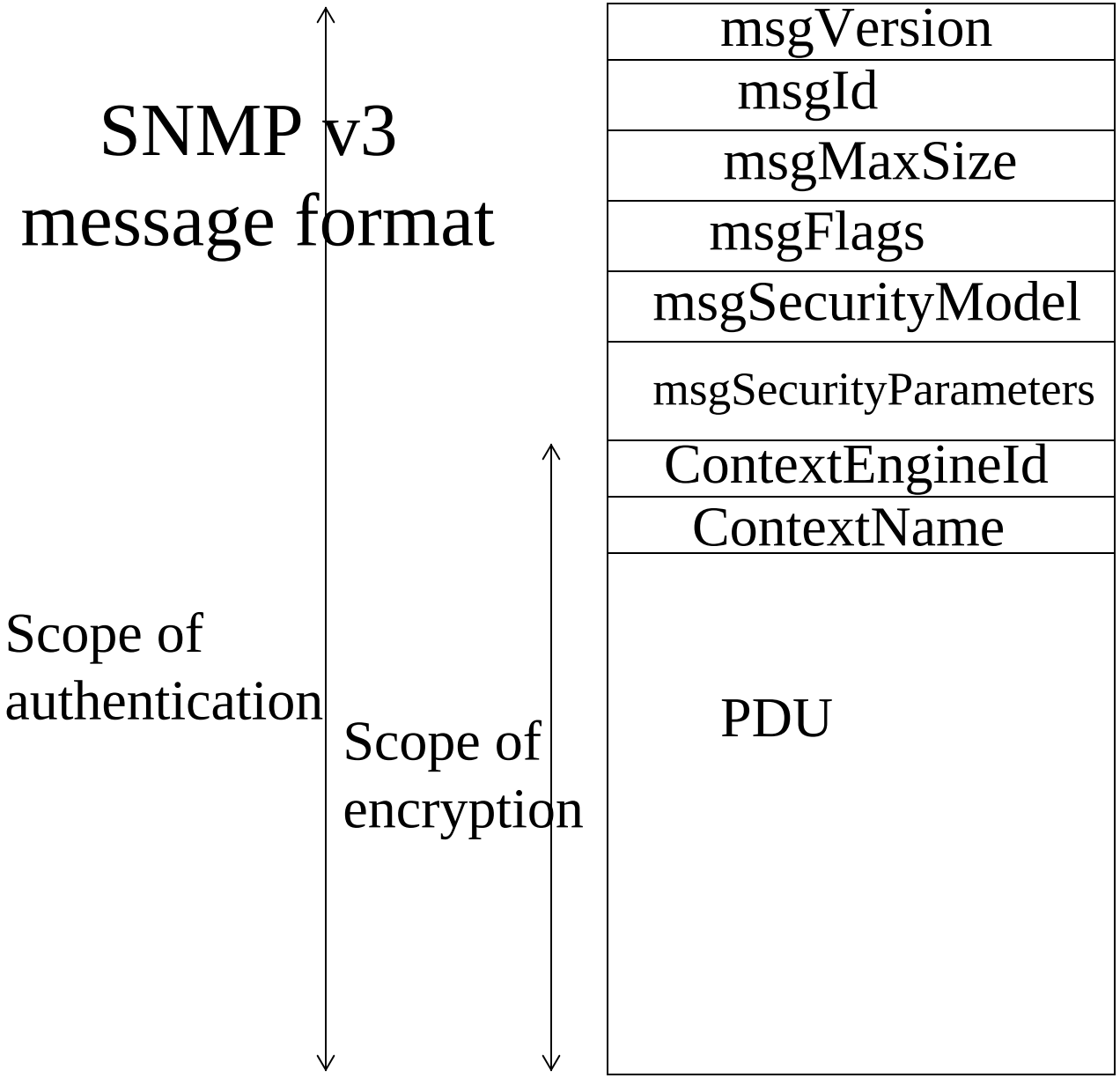
"SNMPv3 = SNMPv2 + security + admin"

Msg = MsgHeader + msgSecurityParms + msgData

MsgHeader =

- msgVersion (3)
- msgId
- msgMaxSize (484 — 2^{31}-1)
- msgFlags (authFlag, privFlag, reportFlag)
- msgSecurityModel (1 (v1), 2 (v2), 3 (v3 - USM))

SNMP v3 message format

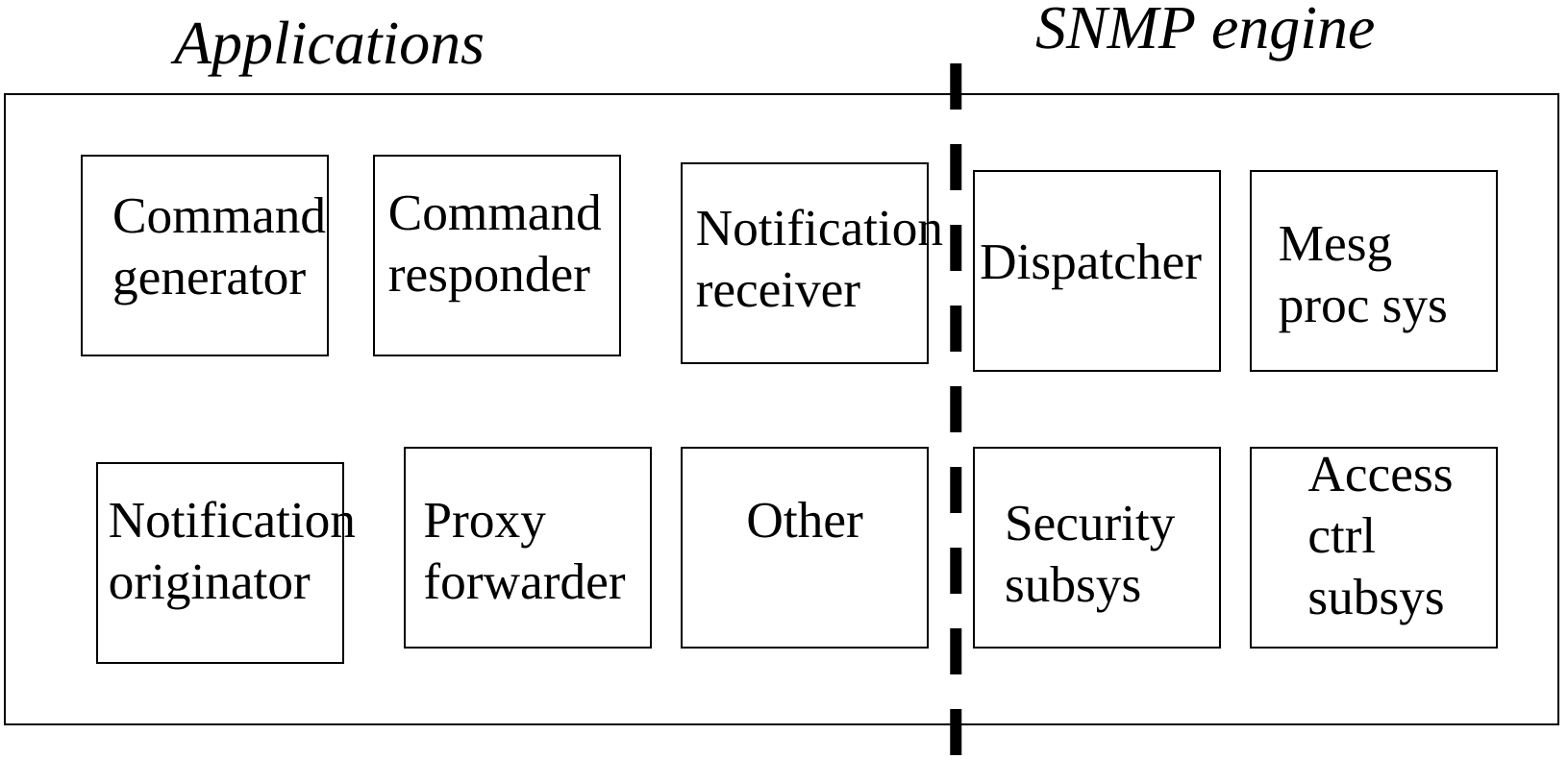


SNMPv3 engine

- An engine consists of
  - Dispatcher
  - Message processing subsystem
  - Security subsystem
  - Access control subsystem

SNMP architecture

- Managers and agents are ‘entities’
- An entity consists of an SNMP engine and one or more SNMP applications



SNMPv3

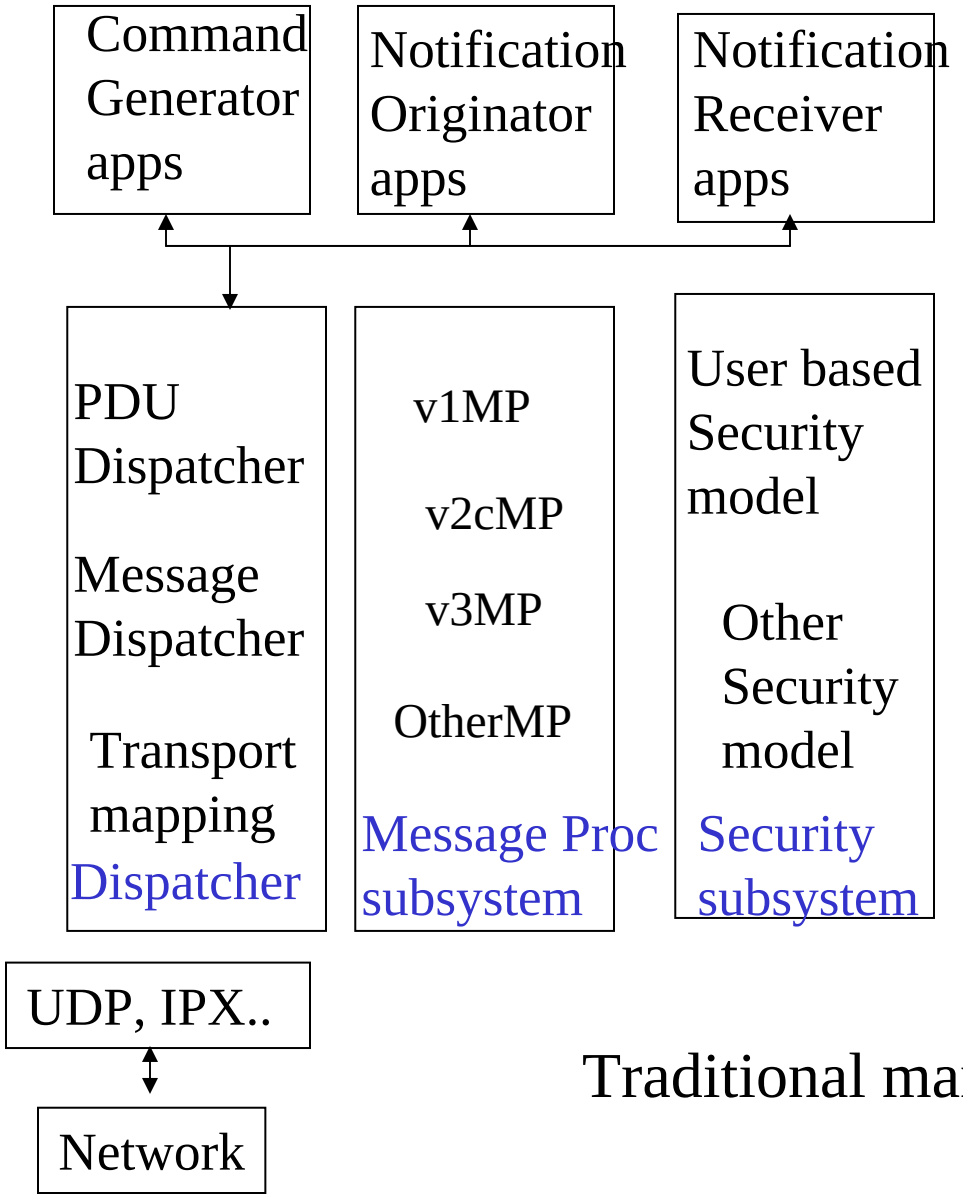
- Dispatcher
  - Allows for multiple versions of SNMP in the engine
  - Transmits SNMP messages to other entities
  - Hands off PDUs to Message Processing subsystem
- Message processing subsystem
  - Prepares messages for sending
  - Extracts data from received messages

SNMPv3

- Security subsystem
  - Authentication and privacy services
  - Multiple security models
- Access control subsystem
  - Authorisation services that can be used to check access rights

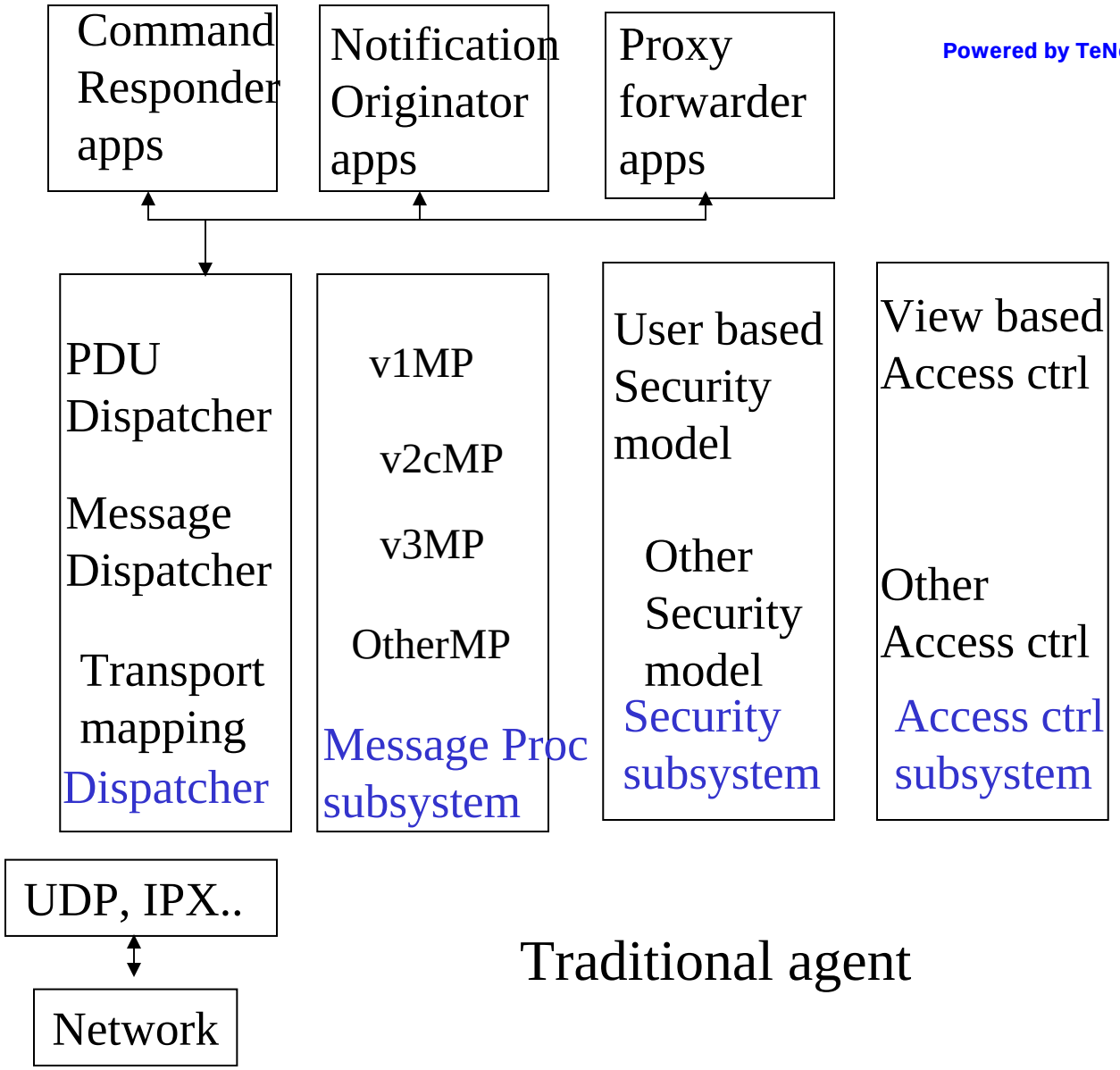
SNMP applications

- Command generator to initiate get, getNext, getBulk and set requests
- Command responder
  - Receives requests and performs the appropriate operation
- Notification originator
  - Generates Trap and/or Inform messages
  - Needs to know where to send notification, ver of SNMP to use etc
- Notification receiver
  - Generates responses to Inform messages



Traditional manager

MIB instrumentation



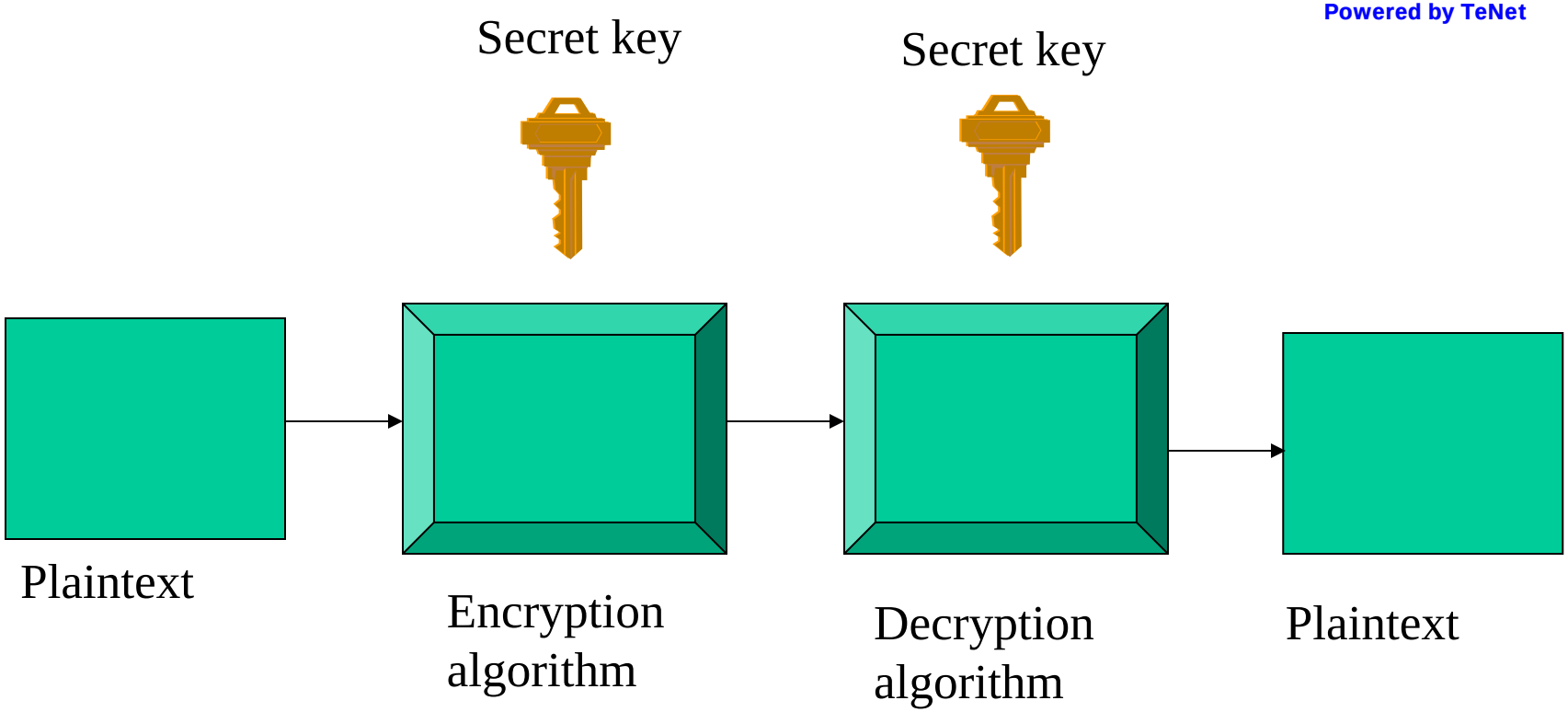
Traditional agent

Cryptographic techniques

- SNMP v3 uses four algorithms
  - DES (encryption)
  - MD5, SHA-1, HMAC for authentiaction

Encryption

- Conventional or symmetric key encryption has five ingredients
  - Plaintext
  - Encryption algorithm
  - Secret key
  - Ciphertext
  - Decryption algorithm



Conventional encryption algorithm

Hash functions

- Used for verifying integrity of messages
- Accept an arbitrary length input, produce a fixed length output
  - Standard output lengths - 16 byte, 20 byte

Message Authentication Codes

- Allows communicating parties to verify that received messages are authentic
  - Source is authentic
  - Contents have not been altered
- Communicating parties share a key
- Key is used to generate a short block of data which is appended to message
- On getting the message, recipient generates the same block of data

User-based Security Model

- Provides
  - timeliness: attacker cannot delay/replay a message
  - authentication: verify sender's identity
  - privacy: protect message contents
  - key management: generation of keys

USM Definitions

Defined by FRC 2274

Authoritative SNMP Engine: source of time for an SNMP transaction

- if request-response (eg. Get, getNext, getBulk, set, inform): receiver is authoritative, sender is non-authoritative
- if request-only (v2 trap, response): sender is authoritative

Engine ID: unique identifier for each SNMP entity (agent or manager)

... USM Definitions

EngineBoots: number of times the authoritative engine has rebooted

A boot occurs when clock reaches 2^31 - 1

EngineTime: time in seconds since the last reboot of the authoritative engine

- non-authoritative engine has an estimate only, updated whenever it receives a message from the authoritative engine

... USM Definitions

Powered by TeNet

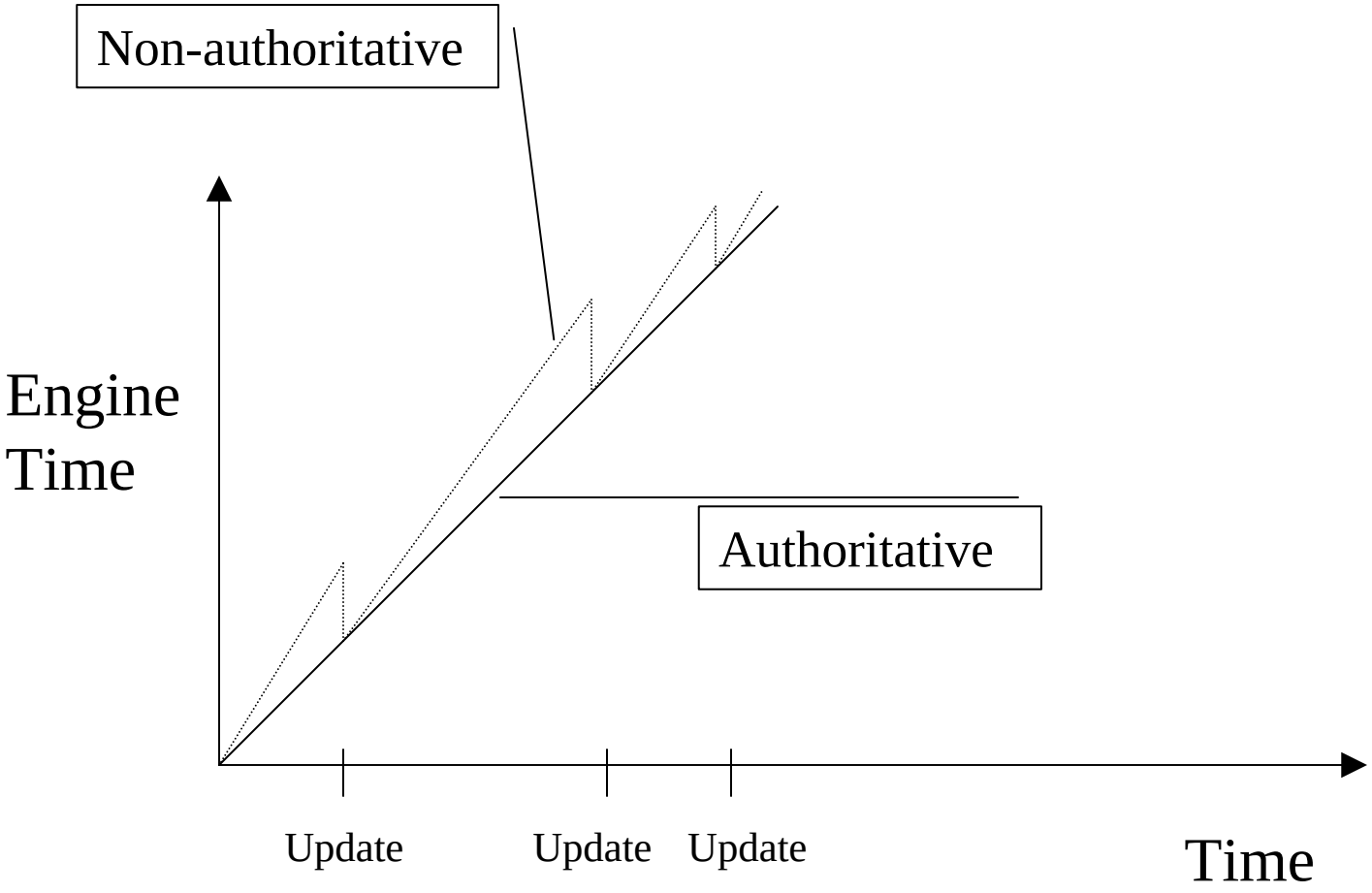
AuthenticationParameters: uses HMAC to compute a *signature* of the message

PrivacyParameters: encrypts message data using cipher-block chaining mode of DES with 56-bit key

... USM Definitions

NMSWorks

Powered by TeNet



USM Timeliness

NMSWorks

Powered by TeNet

Four aspects

- Management of authoritative clocks
- Synchronization
- Timeliness checking by receiver (Authoritative)
- Timeliness checking by receiver (nonauthoritative)

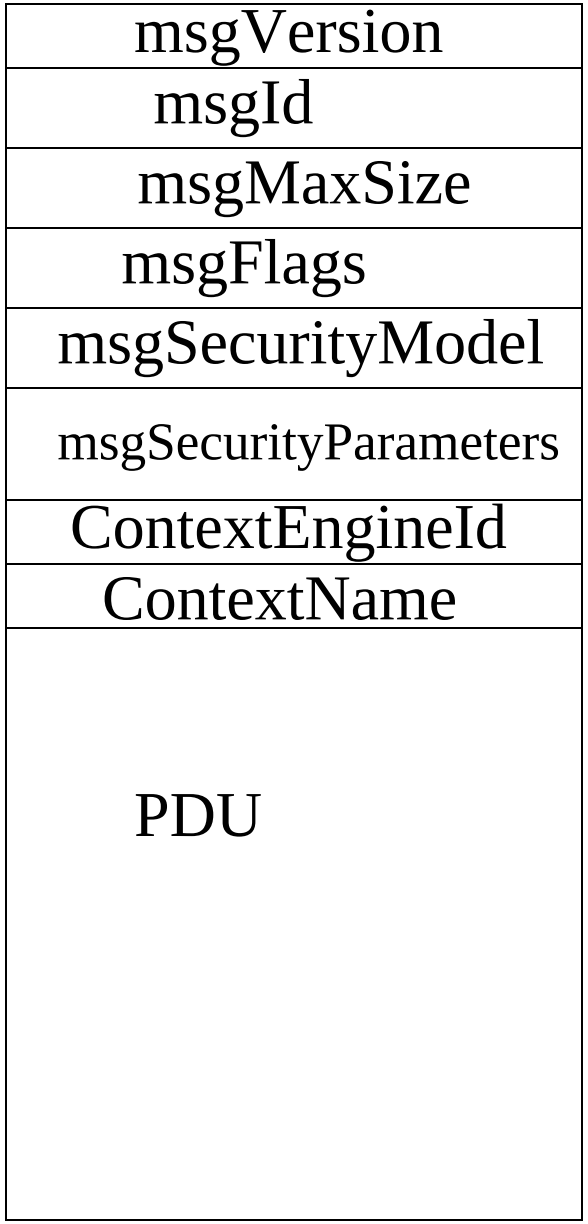
Authoritative clocks

- Authoritative engine maintains snmpEngineBoots, snmpEngineTime (both initialised to 0)
- Thereafter, snmpEngineTime is incremented once per second
- If snmpEngineTime reaches its max of 2^31 -1, it is reset to 0 as if the engine has rebooted
- snmpEngineReboot incremented by 1

Synchronization

Powered by TeNet

- Synchronization between each nonauthoritative engine and each authoritative engine with which it communicates
- Following variables maintained for this
  - snmpEngineBoots
  - snmpEngineTime
  - latestReceivedEngineTime
- Appropriate field in message header are updated with these values
- Update occurs if boot value has increased since last update
- If boot value has not increased, incoming engine time should be greater than latest received engine time



msgAuthoritativeEngineId

msgAuthoritativeEngineBoots

msgAuthoritativeEngineTime

msgUserName

msgAuthenticationParameters

msgPrivacyParameters

USM Timeliness

Receiver accepts message only if within a time window

Else, may be replay attack or partner malfunctioning

Authoritative receiver accepts if:

msg.engineBoots = authEngineBoots AND

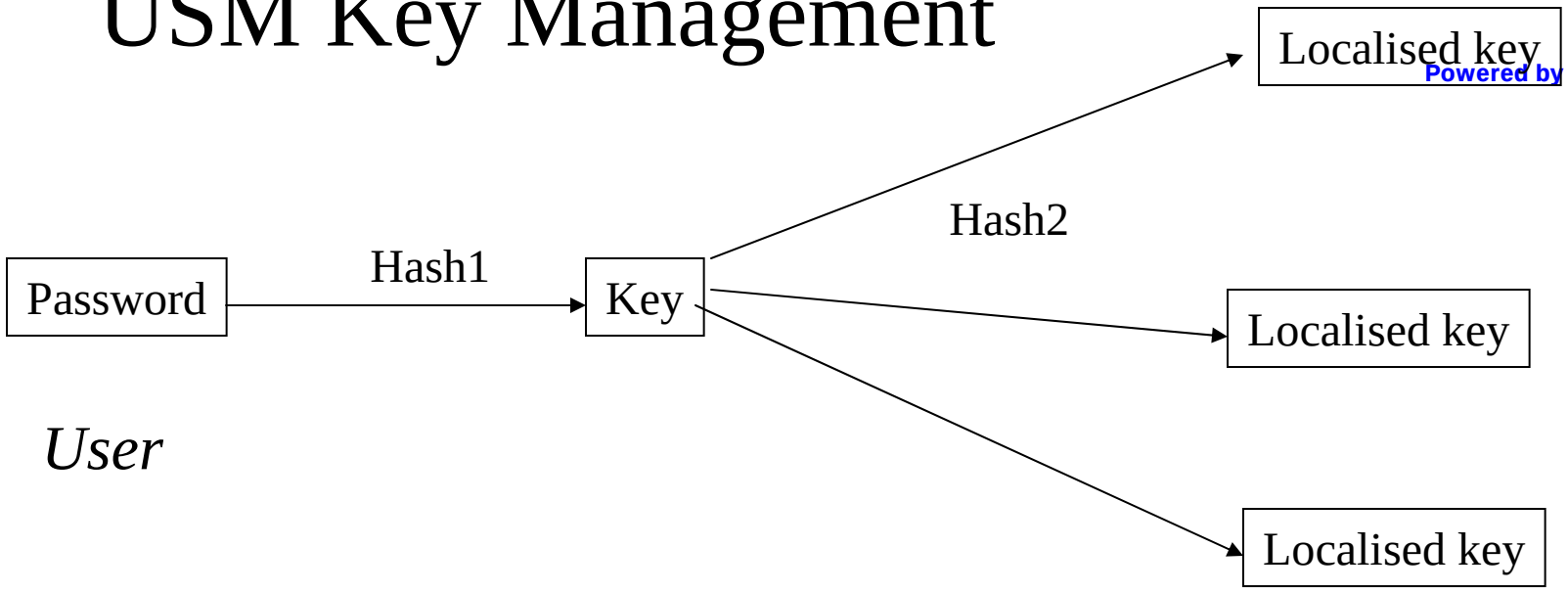
msg.engineTime = authEngineTime ± 150 secs

Non-authoritative receiver accepts if:

engineBoots = authEngineBoots AND

authEngineTime ≥ engineTime – 150 secs

USM Key Management



Hash1:

- repeat password to get 2^{20} octet string (digest0)
- take MD5 or SHA hash of digest0 to get 16- or 20-octet key

Hash2:

- take MD5 or SHA hash of key+agent engine ID to get 16- or 20-octet localised key

... Key Management

Powered by TeNet

- cracking the password is difficult
- if one agent is compromised, other agents are not affected
- user can manage agent from anywhere, not only from an NMS

Key Update

- deliver localised key to agent (outside SNMP)
- *set(keyChange)* $\Rightarrow$ agent changes to next key

View-based Access Control

- V1 and V2 use a single community string for many purposes
- *V3 provides several variables for finer access control via VACM*

... VACM

Groups: set of <securityModel, principal> tuples (vacmSecurityToGroupTable) on whose behalf managed objects can be accessed

Security Level: noAuthNoPriv, authNoPriv, authPriv

- agent may allow greater access for more secure messages

Contexts: named subsets of object instances in the local MIB (vacmContextTable)

MIB Views: collection of MIB sub-trees, each included or excluded from the view (vacmMIBViews)

- each entry in vacmAccessTable has read, write, notify views

Powered by TeNet

... VACM

Access Policy:

- permit or deny access based on:
  - principal
  - security level
  - context
  - object instance
  - type of access

... VACM Example

Powered by TeNet

SecurityToGroupTable

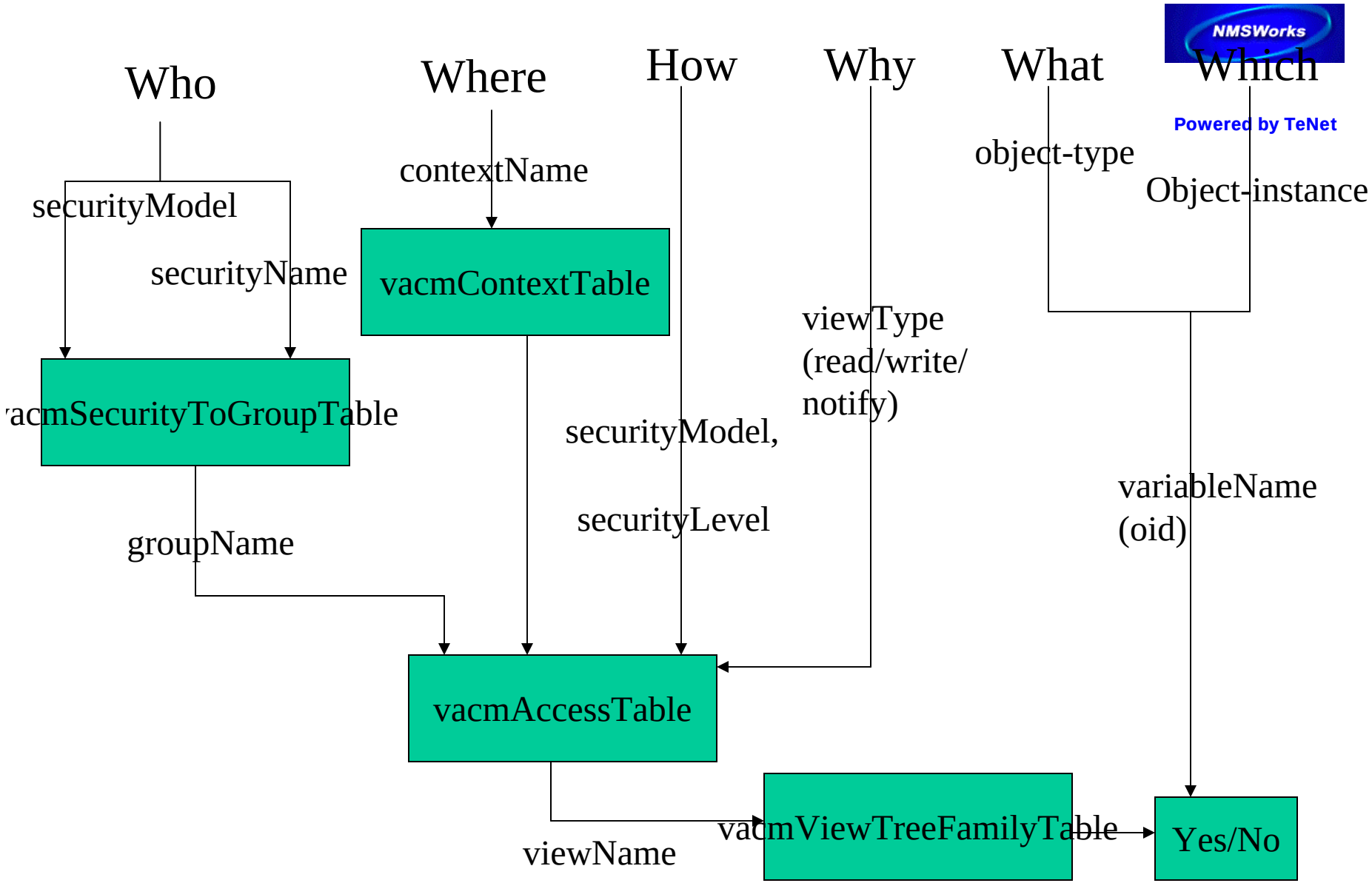
Sec.Model	Sec.Name	GroupName
V1	“director”	“public”
USM (V3)	“director”	“admin”

ViewAccessTable

GroupName	Sec.Level	ReadView	WriteView
“admin”	authPriv	“internet”	“internet”
“admin”	noAuthNoPriv	“restricted”	“”
“public”	authPriv	“restricted”	“”

ViewTreeFamilyTable

ViewName	SubTree
“internet”	1.3.6.1 (internet)
“restricted”	1.3.6.1.2.1.1 (system)
“restricted”	1.3.6.1.2.1.11 (snmp)


Who
Where
How
Why
What
Which
NMSWorks
Powered by TeNet
securityModel
securityName
contextName
vacmContextTable
object-type
Object-instance
viewType
(read/write/
notify)
vacmSecurityToGroupTable
securityModel,
securityLevel
variableName
(oid)
groupName
vacmAccessTable
viewName
vacmViewTreeFamilyTable
Yes/No

Summary

SNMPv3

- Security: authentication and encryption
- Flexible view-based access control