Netzwerkmanagement mit SNMP

→ Teil 4: SNMPv3

Prof. Dr. Norbert Pohlmann

Fachbereich Informatik

Verteilte Systeme und Informationssicherheit

Fachhochschule Gelsenkirchen

Inhalt

- Einführung
- Das Architekturmodell
- Das benutzerbasierte Sicherheitsmodell (USM)
- Das sichtenbasierte Zugriffskontrollmodell (VACM)

Weiterentwicklung von SNMP: SNMPv3

Einführung SNMPv3 (1/2)

- Nach dem Scheitern der SNMPv2 Arbeitsgruppe wurde die SNMPv3 Arbeitsgruppe ins Leben gerufen, die mit Hochdruck an einem einheitlichen Standard als Nachfolger von SNMPv1/2 arbeitet.
- Die Ziele der SNMPv3 Arbeitsgruppe waren:
  - Es sollte soviel wie möglich von den zwei konkurrierenden Versionen SNMPv2u und SNMPv2* übernommen werden.
  - Es sollte ein Sicherheitskonzept entwickelt werden, das es erlaubt, die set Operation über das weltweite Internet zu verwenden.
  - SNMPv3 sollte eine flexible Architektur bekommen, die
    - den Einsatz in einer Vielzahl von Umgebungen berücksichtigt (von kleinen kostengünstigen Umgebungen mit wenig Funktionalität bis hin zu großen weltweiten Netzen),
    - es ermöglichen, einzelne Teilbereiche der Architektur weiterzuentwickeln auch wenn die Entwicklung der Gesamtarchitektur noch nicht abgeschlossen ist,
    - und es ermöglicht, alternative Modelle zu integrieren.

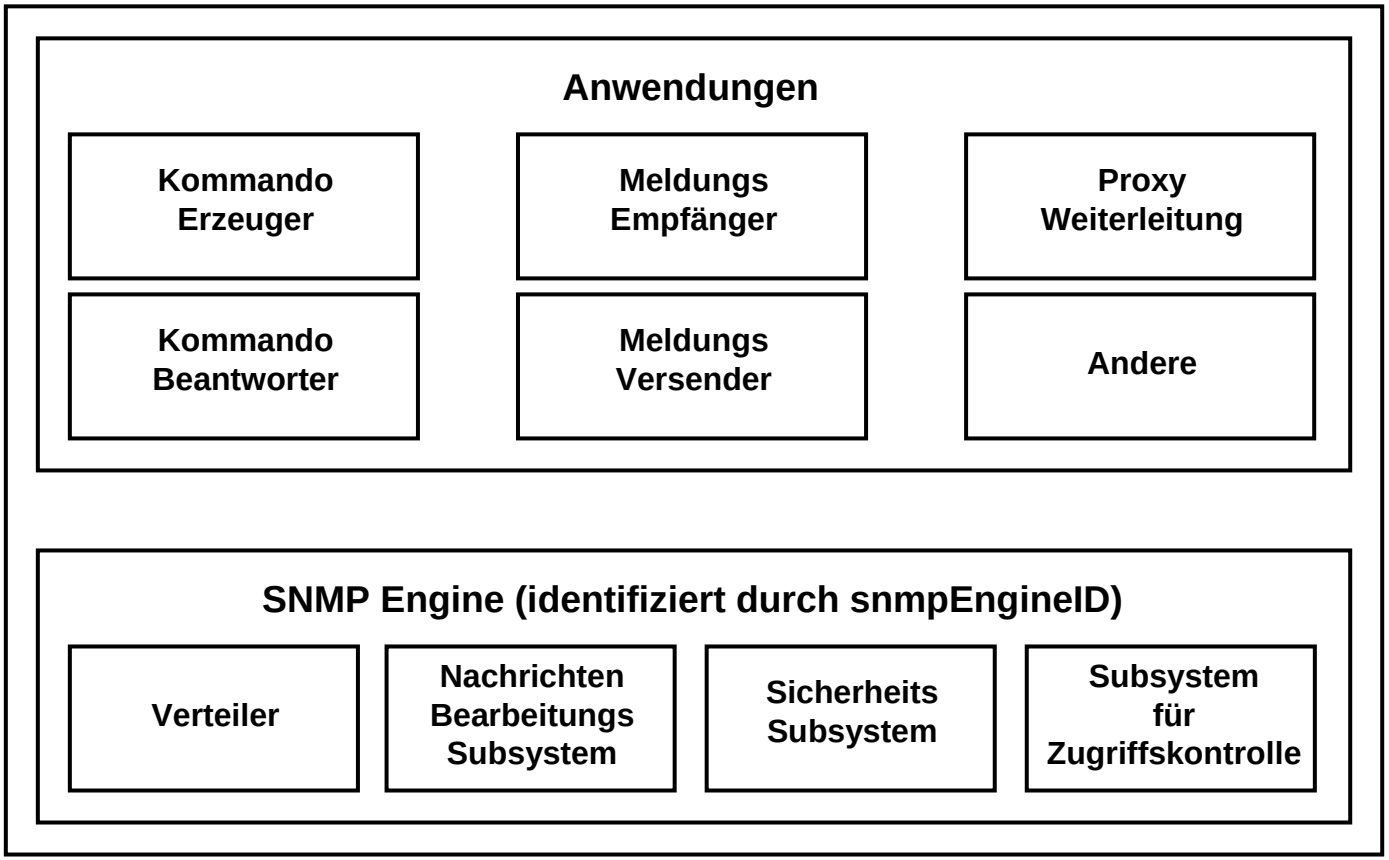
Einführung SNMPv3 (2/2)

- Folgende Dokumente wurden von der SNMPv3 Arbeitsgruppe erarbeitet:
  - RFC2271 *An Architecture for Describing SNMP Management Frameworks*
  - RFC2272 *Message Processing and Dispatching for the SNMP Protocol*
  - RFC2273 *SNMPv3 Applications*
  - RFC2274 *User-based Security Model (USM) for version 3 of SNMP*
  - RFC2275 *View-based Access Control Model (VACM) for SNMP*
- SNMPv3 ist kein neuer Standard für das Netzmanagement.
- SNMPv2 ist komplett in SNMPv3 übernommen.
- Neu in SNMPv3 ist das
  - Architekturmodell und die
  - Sicherheit.

Das Architekturmodell

- Eines der wichtigsten Ziele der SNMPv3 Arbeitsgruppe war die Entwicklung einer flexiblen Architektur, die es erlaubt, zukünftige Erweiterungen, bzw. bereits bestehende Differenzen zwischen den verschiedenen Versionen einfach zu integrieren.
- Die Gesamtarchitektur, wie sie in RFC2271 beschrieben ist, besteht aus einer Anzahl von verteilten, interagierenden SNMP Einheiten.
- Eine SNMP Einheit kann dabei die Rolle eines Agenten, die eines Managers oder eine Kombination aus beiden annehmen.
- Jede SNMP Einheit besteht aus mehreren diskreten Subsystemen, die einen bestimmten Dienst bereitstellen sowie aus mehreren Applikationen, die diese Dienste in Anspruch nehmen.
- Für jedes Subsystem wurde eine abstrakte Dienstschnittstelle definiert, die es anderen Subsystemen oder Applikationen erlauben, auf diesen Dienst zuzugreifen.

SNMP Einheit (1/3)



- Die einzelnen Subsysteme einer SNMP Einheit werden zu einer SNMP Engine zusammengefasst.
- Eine SNMP Engine wird durch eine systemweit eindeutige Engine-ID identifiziert und stellt Dienste für das Senden und Empfangen von SNMP Nachrichten, für Authentifizierung und Verschlüsselung und für die Zugriffskontrolle zur Verfügung.

SNMP Einheit (2/3)

- Die Applikationen sind ebenso wie die SNMP Engine in diskrete Module unterteilt und repräsentieren die Anwenderlogik in den Manager- und Agentenprozessen. Die folgenden Applikationen wurden bisher definiert:
  - **Kommandoerzeuger** — initiieren get, get-next, get-bulk und/oder set Operationen und verarbeiten die dazugehörigen Antworten.
  - **Meldungsempfänger** — behandeln asychrone Meldungen (v1Trap-, v2Trap- und inform Operationen)
  - **Kommandoempfänger** — verarbeiten get, get-next, get-bulk und/oder set Operationen, überprüfen die Zugriffsrechte und erzeugen entsprechende Antworten.
  - **Meldungserzeuger** — generieren beim Auftreten bestimmter Ereignisse asychrone Meldungen (v1Trap-, v2Trap- und inform Operationen)
  - **Proxy Weiterleitung** — leiten SNMP Nachrichten in transparenter Weise an andere SNMP-Einheiten weiter.

SNMP Einheit (3/3)

- Je nach Zusammenstellung der einzelnen Subsysteme und Applikationen agiert die SNMP Einheit als Manager oder Agent, bzw. als Kombination von beiden.
- Ein traditioneller SNMP Agent würde z.B. alle Subsysteme der SNMP Engine enthalten, sowie die Kommunikationsempfänger und Meldungsversender.
- Ein traditioneller SNMP Manager würde dagegen das Zugriffskontrollsubsystem nicht in seiner SNMP Engine enthalten, dafür aber die Applikationen Kommandoerzeuger, Meldeversender und Meldeempfänger.

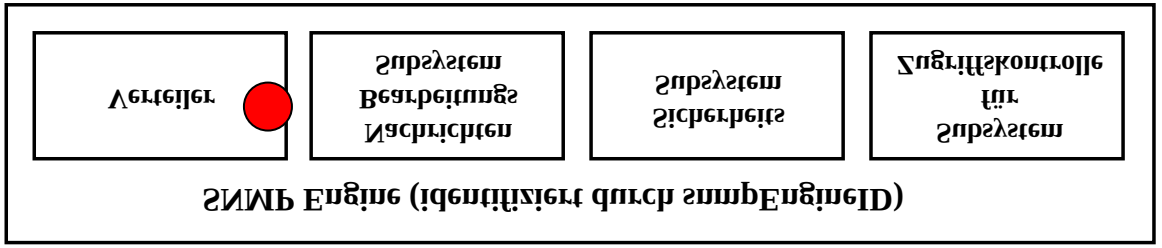
Abstrakte Dienstschnittstelle

- Die Dienste, die von den einzelnen Subsystemen einer SNMP Einheit bereitgestellt werden, werden in den RFCs mit Hilfe von **Dienstelementen** (service primitives) definiert.
- Ein Dienstelement spezifiziert die Funktion, die ausgeführt ist und die Parameter, die verwendet werden, um Daten oder Kontrollparameter zu übergeben.
- Die tatsächlichen Ausprägung eines Dienstelements wird nicht spezifiziert, sondern ist von der jeweiligen Implementierung abhängig.
- Ein typischer Aufruf einer Dienstschnittstelle sieht folgendermaßen aus:

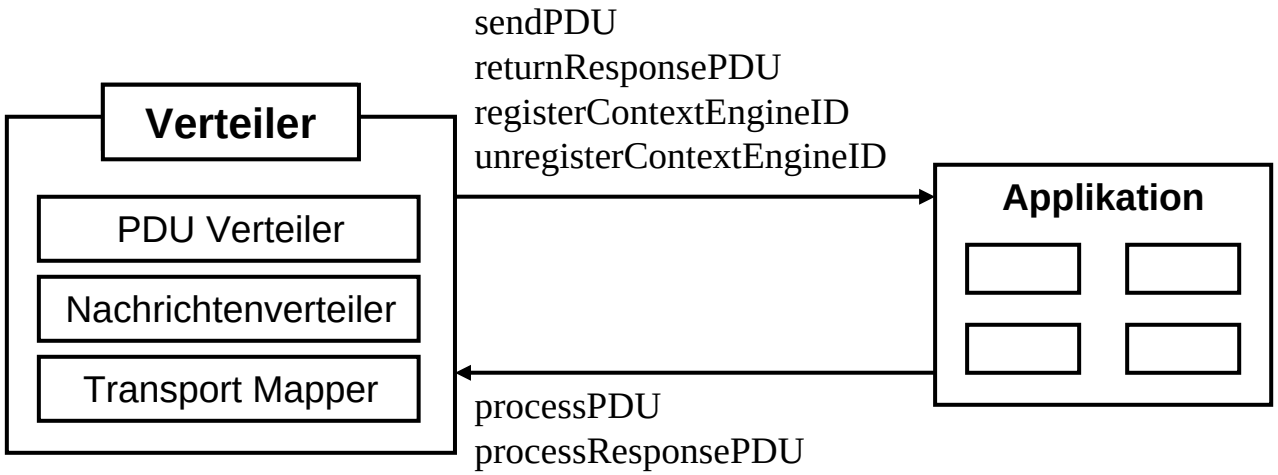
```
Rückgabewert = Funktionsanforderung (Eingabewert1, Eingabewert2, ...
                                     Ausgabewert1, Ausgabewert2, ... )
```

Der Verteiler (*Dispatcher*)

- Der Verteiler ist die **zentrale Schaltstelle** innerhalb einer SNMP Einheit und stellt Dienste zum Versenden von PDUs und Empfangen von Nachrichten bereit.
- Applikationen, wie z.B. Kommandoerzeuger oder Meldungsversender) übergeben die SNMP PDUs, die sie versenden möchten, an den Verteiler.
- Der Verteiler wiederum sendet die PDU an das entsprechende Nachrichtenbearbeitungssubsystem.
- Das Nachrichtenbearbeitungssubsystem wandelt die SNMP PDU in ein entsprechendes SNMP Nachrichtenformat (SNMPv1, SNMPv2 oder SNMPv3) um und gibt die erzeugte SNMP Nachricht wieder an den Verteiler zurück, der die vollständige Nachricht auf das jeweilige Transportprotokoll überträgt und an den Empfänger sendet.



Der Verteiler (*Dispatcher*)
→ Abstrakte Schnittstelle des Verteilers



- Im umgekehrten Fall empfängt der Verteiler SNMP-Nachrichten von der Transportschicht.
- Eingehende SNMP Nachrichten werden je nach Version an das entsprechende Nachrichtenbearbeitungssubsystem übergeben, das aus der Nachricht die SNMP PDU extrahiert und an den Verteiler zurückgibt.
- Die extrahierten PDUs werden nun vom Verteiler an die entsprechenden Applikationen verteilt.

Der Verteiler (*Dispatcher*)
→ Dienstelemente des Verteilers (1/2)

- **sendPDU**
  Applikationen, wie z.B. die Kommandoerzeuger Applikation, nutzen diese Funktion, um PDUs an andere SNMP Einheiten zu senden.
  Als Parameter werden die zu sendende PDU, das entsprechende Nachrichtenformat sowie verschiedene Sicherheitsparameter übergeben.
  Die Funktion liefert ein „Handle“ zurück, das der *request-ID* der SNMP Nachricht entspricht und für die Zuordnung der Antwort verwendet wird.

- **processResponsePDU**
  Diese Funktion wird vom Verteiler benutzt, um eintreffende Antwort-PDUs den entsp. Applikationen zuzuordnen.

- **processPDU**
  Diese Funktion wird vom Verteiler benutzt, um eintreffende SNMP PDUs an die entsp. Applikationen zu verteilen.

Der Verteiler (*Dispatcher*)
→ Dienstelemente des Verteilers (2/2)

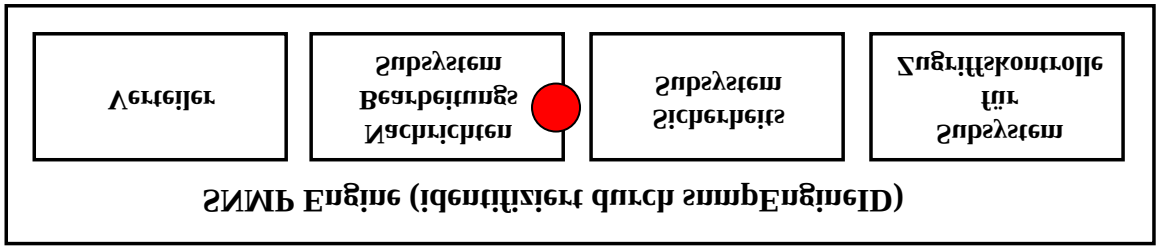
- **returnResponsePDU**
  Applikationen, wie z.B. die Kommandobeantworter Applikation, benutzen diese Funktion, um eine entsp. PDU als Antwort einer vorhergehenden Anfrage oder Meldung zurückzusenden.

- **registerContextEngineID**
  Applikationen nutzen diese Funktion, um sich für bestimmte PDU Typen zu registrieren.
  Kommandoerzeuger Applikationen müssen sich beispielsweise für „resonse-PDUs“ anmelden.

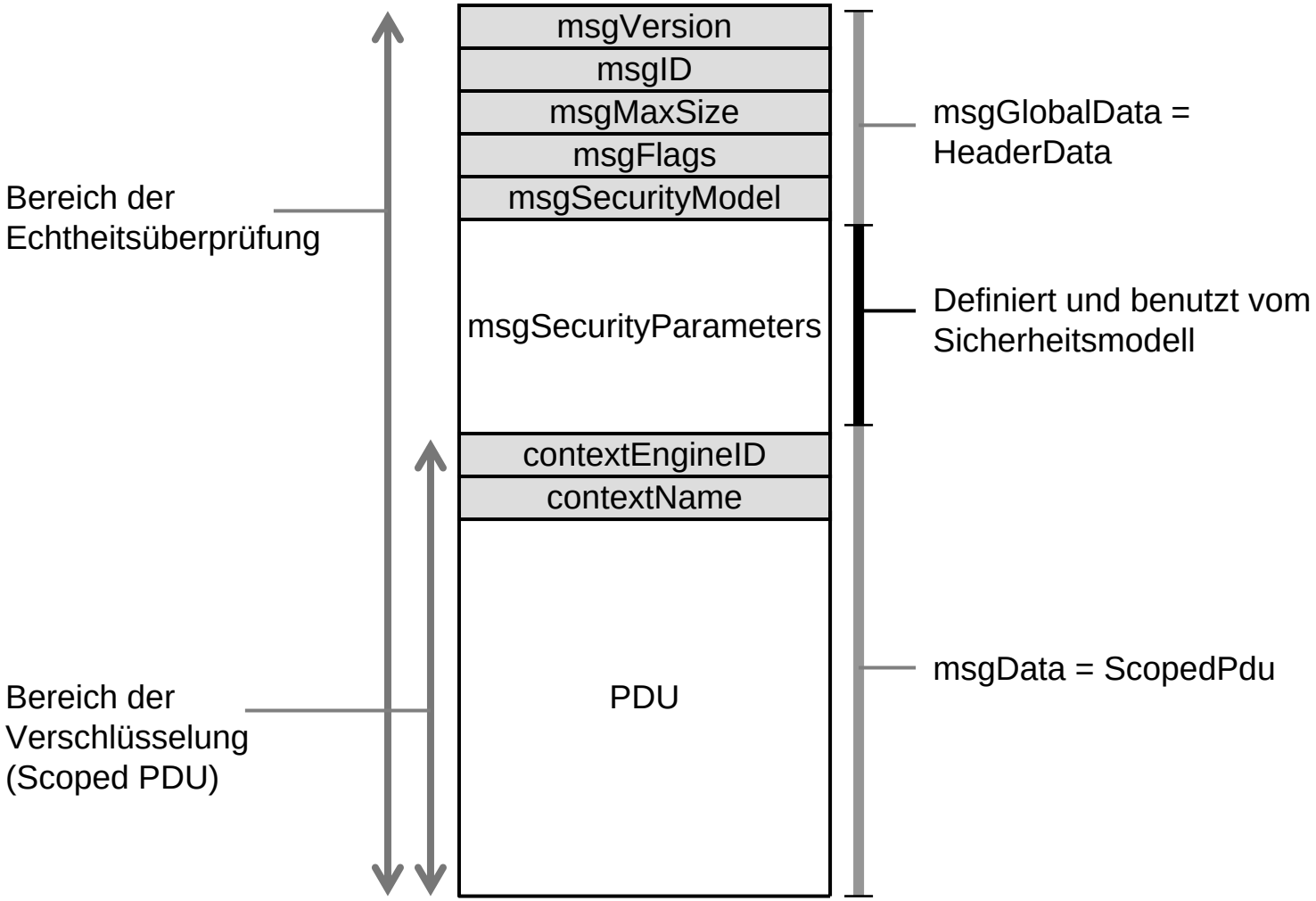
- **deregisterContextEngineID**
  Applikationen nutzen diese Funktion, um sich von bestimmten PDU Typen abzumelden.

Nachrichtenbearbeitungssubsystem
(*Message Processing Subsystem*)

- Das **Nachrichtenbearbeitungssubsystem** ist für die Interpretation und Erzeugung von SNMP Nachrichten zuständig.
- Das Subsystem kann aus verschiedenen Nachrichtenbearbeitungsmodellen (Message-Processing-Modellen) bestehen, wobei im aktuellen Ansatz Modelle für SNMPv1, SNMPv2c und SNMPv3 vorgesehen sind.



Nachrichtenbearbeitungssubsystem
→ Aufbau einer SNMPv3 Nachricht



Nachrichtenbearbeitungssubsystem
→ Bedeutung der Felder (1/2)

- **msgVersion**
  Version des SNMP Nachricht (2 = SNMPv3)
- **msgID**
  Eine eindeutige ID, welche von zwei SNMP Einheiten verwendet wird, um Anforderungen und Antworten miteinander zu koordinieren.
- **msgMaxSize**
  Beinhaltet die maximale Nachrichtengröße in Octests, welche der Sender zu verarbeiten vermag.
- **msgFlags**
  Dieser Octetstring beinhaltet drei Flags.
  - **reportableFlag** = 1 : Es muss eine Antwort erzeugt werden.
  - **privFlag** = 1 : Die Nachricht ist verschlüsselt.
  - **authFlag** = 1 : Die Nachricht kann auf Echtheit überprüft werden.

  Hinweis: Verschlüsselung ist nur mit Echtheitsüberprüfung erlaubt.

Nachrichtenbearbeitungssubsystem
→ Bedeutung der Felder (2/2)

- **msgSecurityModel**
  Eine Zahl zwischen 0 und 2^{31}-1.
  Hiermit wird die Art des verwendeten Sicherheitsmodells angezeigt.
  Vordefiniert sind: 1 für SNMPv1, 2 für SNMPv2 und 3 für USM

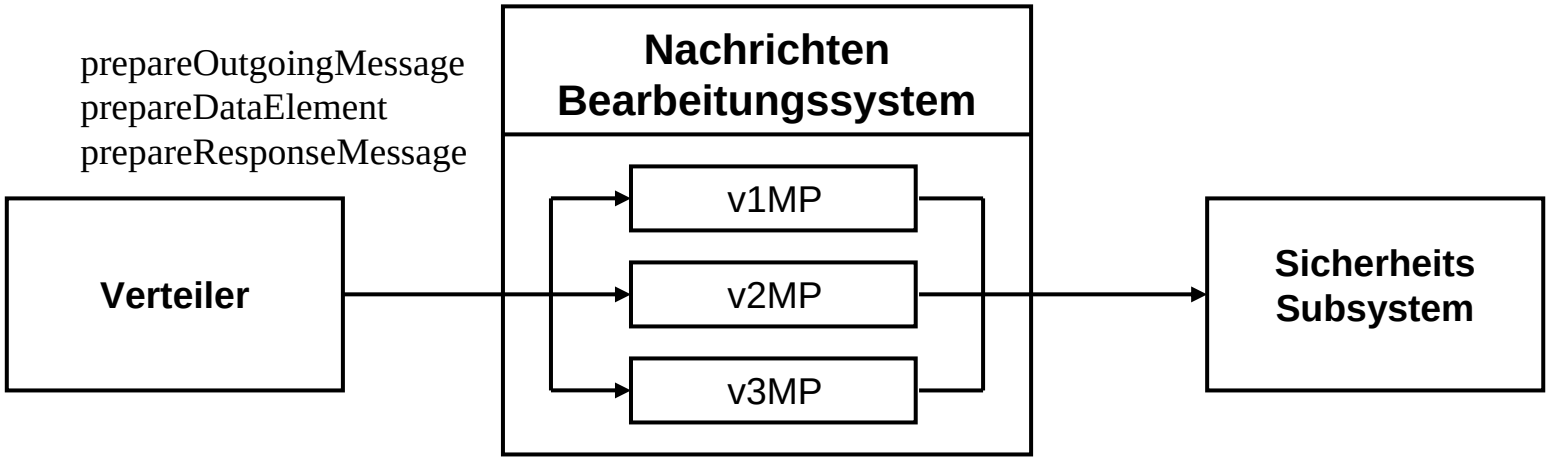
- **msgSecurityParameters**
  Ein Octetstring, welcher vom Sender generiert und vom Empfänger interpretiert wird. Er enthält Parameter für das Sicherheitssubsystem.

- **contextEngineID**
  Ein Octetstring, welcher die SNMP Einheit identifiziert.
  Bei eingehenden Nachrichten wird hiermit angezeigt, an welche Anwendung die PDU gesendet werden muss.
  Bei ausgehenden Nachrichten wird dieses Feld von der sendenden Anwendung gesetzt.

- **contextName**
  Ein String, welcher den Kontext identifiziert.

- **data**
  Die mitgesendete PDU.

Nachrichtenbearbeitungssubsystem
→ Schnittstellen



- Das Nachrichtenbearbeitungssubsystem berücksichtigt ausschließlich die ersten fünf Felder des Nachrichtenheaders, während die Sicherheitsparameter vom Sicherheitssubsystem interpretiert und gesetzt werden.

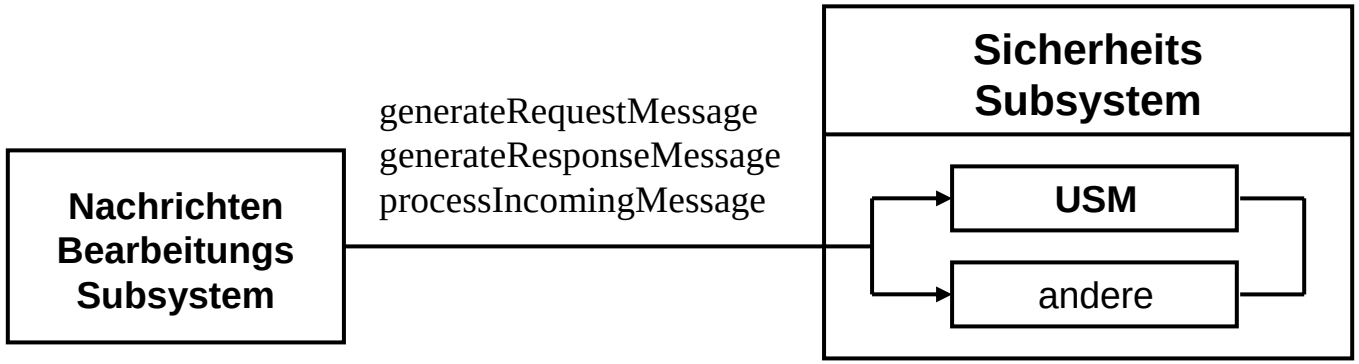
Nachrichtenbearbeitungssubsystem
→ Dienstelemente

- **prepareOutgoingMessage**
  Diese Funktion wird vom Verteiler genutzt, um zu sendende PDU in die entsprechende SNMP Nachricht zu konvertieren.
  Die Funktion erwarten eine PDU als Eingabeparameter und liefert die gewünschte SNMP Nachricht als Ausgabeparameter zurück.

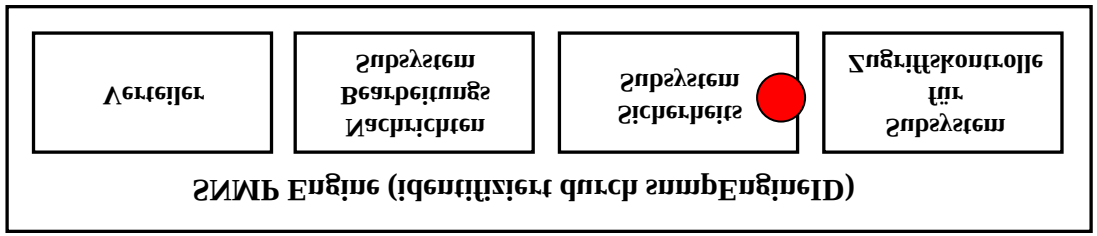
- **prepareDataElement**
  Diese Funktion wird vom Verteiler genutzt, um aus einer SNMP Nachricht die entsprechende PDU zu extrahieren.

- **prepareResponseMessage**
  Diese Funktion wird vom Verteiler genutzt, um eine Antwort-PDU in eine entsprechende SNMP Nachricht umzuwandeln.

Sicherheitssubsystem
(*Security Subsystem*)



- Das Sicherheitssubsystem kann ebenfalls aus verschiedenen, parallel eingebundenen Sicherheits-Modellen (SM) bestehen.
- Das erste definierte Sicherheits-Modell (SM) in der SNMPv3 Architektur ist das **User Based Security Model** (USM), das zu einem Großteil von SNMPv2u übernommen wurde.
- Es muss die Datenintegrität und Authentizität sowie Vertraulichkeit und Aktualität der SNMP Nachricht gewährleisten.



Sicherheitssubsystem
→ Dienstelemente

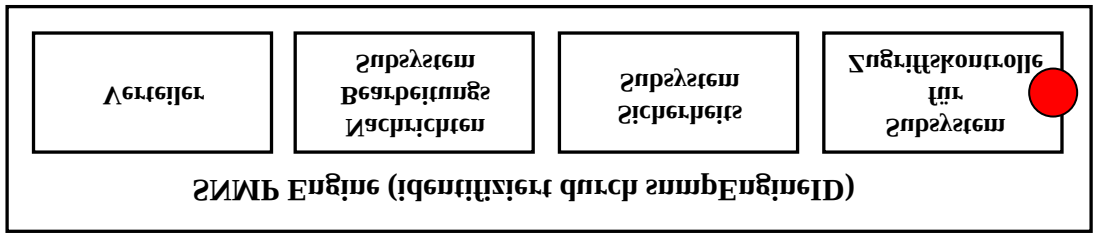
- **generateRequestMessage**
  Diese Funktion führt je nach Sicherheitsstufe die entsprechende Authentifizierung und Verschlüsselung einer SNMP Nachricht durch und gibt das Ergebnis zusammen mit den entsprechende Sicherheitsparametern zurück.

- **generateResponseMessage**
  Diese Funktion entspricht der generateRequestMessage Funktion mit dem Unterschied, dass diese Funktion für Antwort PDUs aufgerufen wird.

- **processIncomingMessage**
  Diese Funktion erwartet eine eventuell authentifizierte und verschlüsselte SNMP Nachricht und überprüft die Authentizität bzw. führt eine Entschlüsselung der Nachricht durch.

Zugriffskontrollsubsystem
(*Access Control Subsystem*)

- Das Zugriffkontrollsubsystem kontrolliert den Zugriff auf *Managed Objects* innerhalb eines Agenten und kann wie die anderen Subsysteme aus mehreren parallel eingebundenen Modellen bestehen.
- Für SNMPv3 ist als erstes das **View Access Control Model** (VACM) definiert worden.

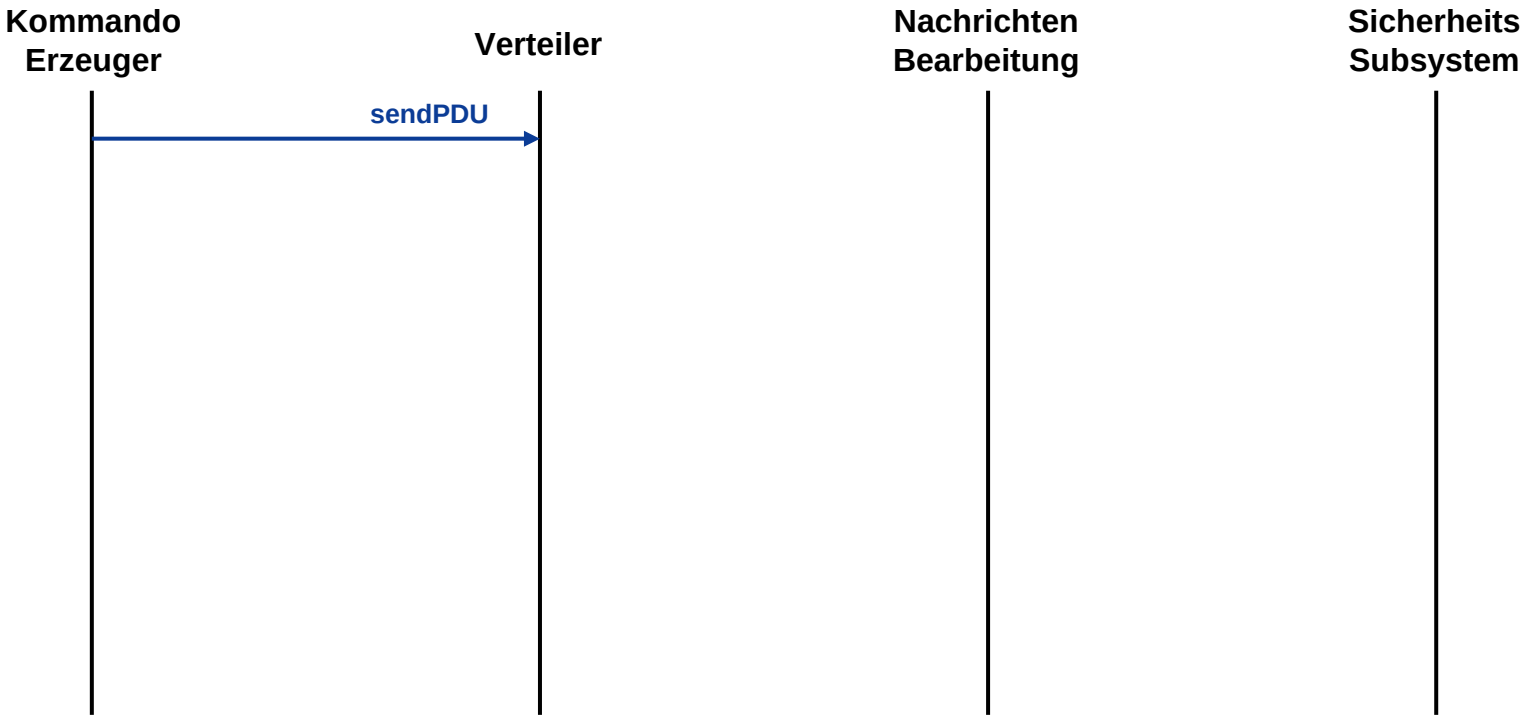


Zugriffskontrollsubsystem
→ Dienstelemente

- **isAccessAllowed**
  Diese Funktion wird von den Applikationen (z.B. Kommandobeantworter) aufgerufen, um den Zugriff auf *Managed Objects* des Agenten zu verifizieren.

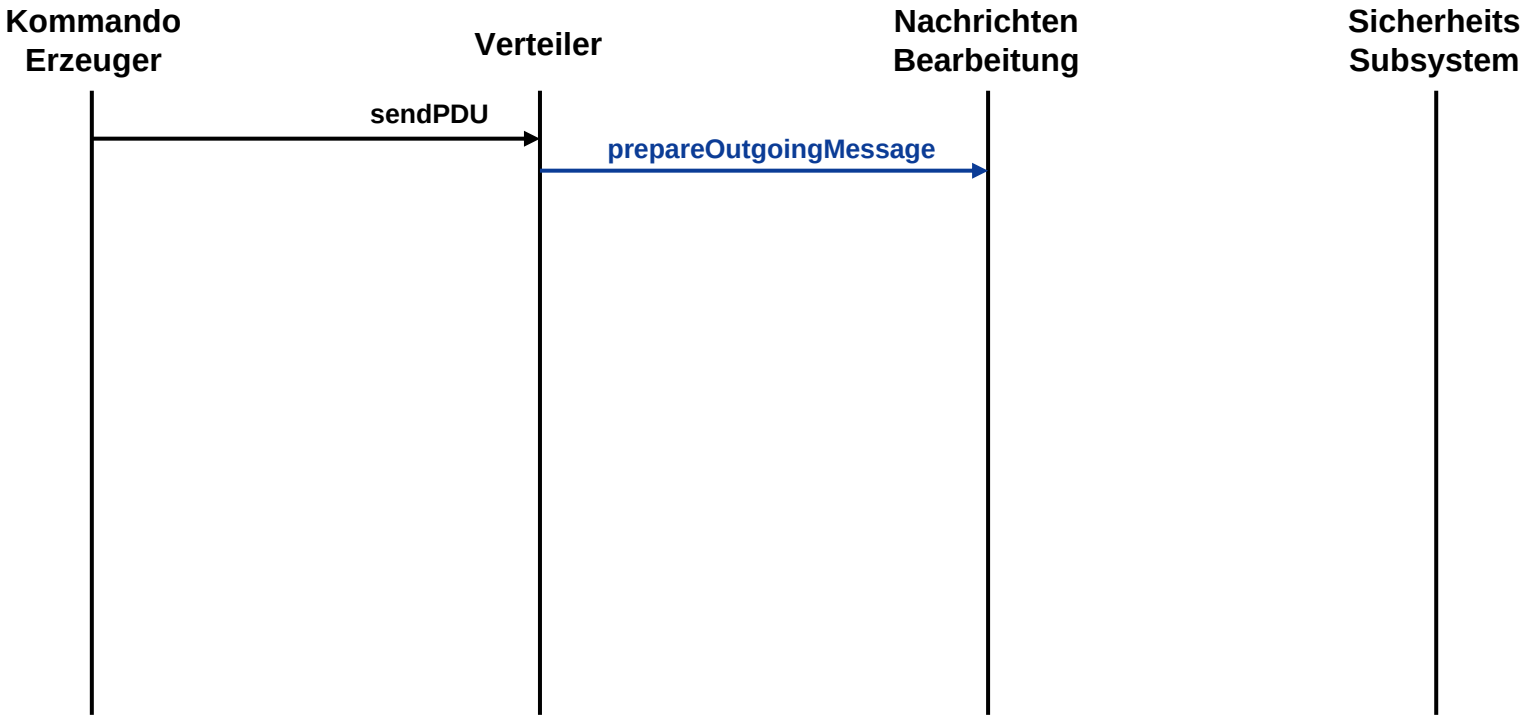
  Die Funktion erwartet als Eingabeparameter den Namen des Anwenders (principal), die Sicherheitsstufe (*security level*), die Art des Zugriffs, den Kontextnamen und den *Object Identifier* und gibt einen Wert zurück, der entweder den Zugriff gewährt (*access allowed*) oder aber verschiedene Fehlerursachen enthält.

Zusammenspiel von Applikation und Subsystem
→ Initiieren einer SNMP Nachricht (1/7)



Um eine SNMP Nachricht an eine andere SNMP Einheit (z.B. einen Agenten) zu senden, ruft der Kommandoerzeuger die sendPDU Funktion des Verteilers auf und übergibt die zu sendende PDU sowie mehrere Parameter die das gewünschte Nachrichtenformat und Sicherheitsmodell definieren.

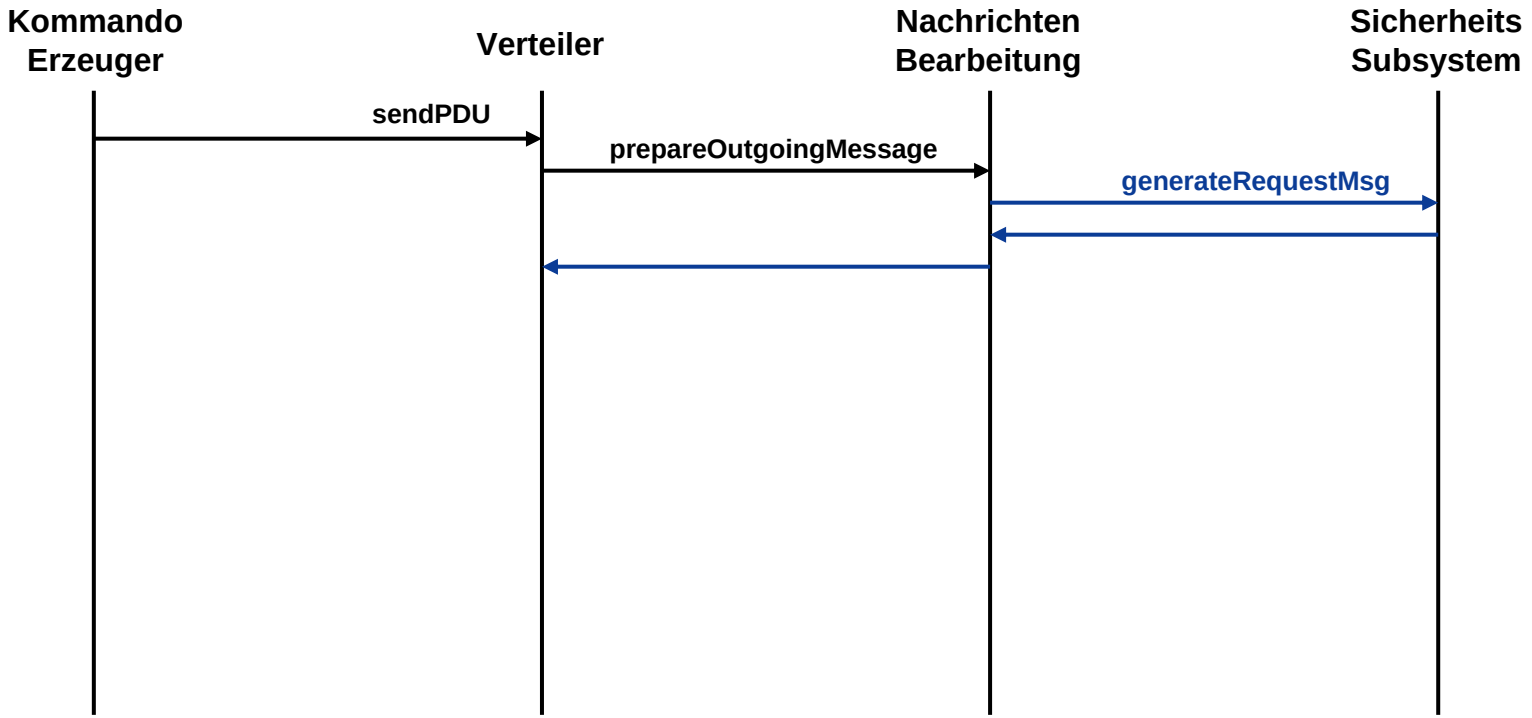
Zusammenspiel von Applikation und Subsystem
→ Initiieren einer SNMP Nachricht (2/7)



Der Verteiler leitet daraufhin die SNMP PDU an das entsp. Bearbeitungsmodul (für SNMPv1/v2/v3) weiter (prepareOutgoingMessage).

Dieses generiert aus der PDU und den übergebenen Parametern eine entsprechende SNMP Nachricht.

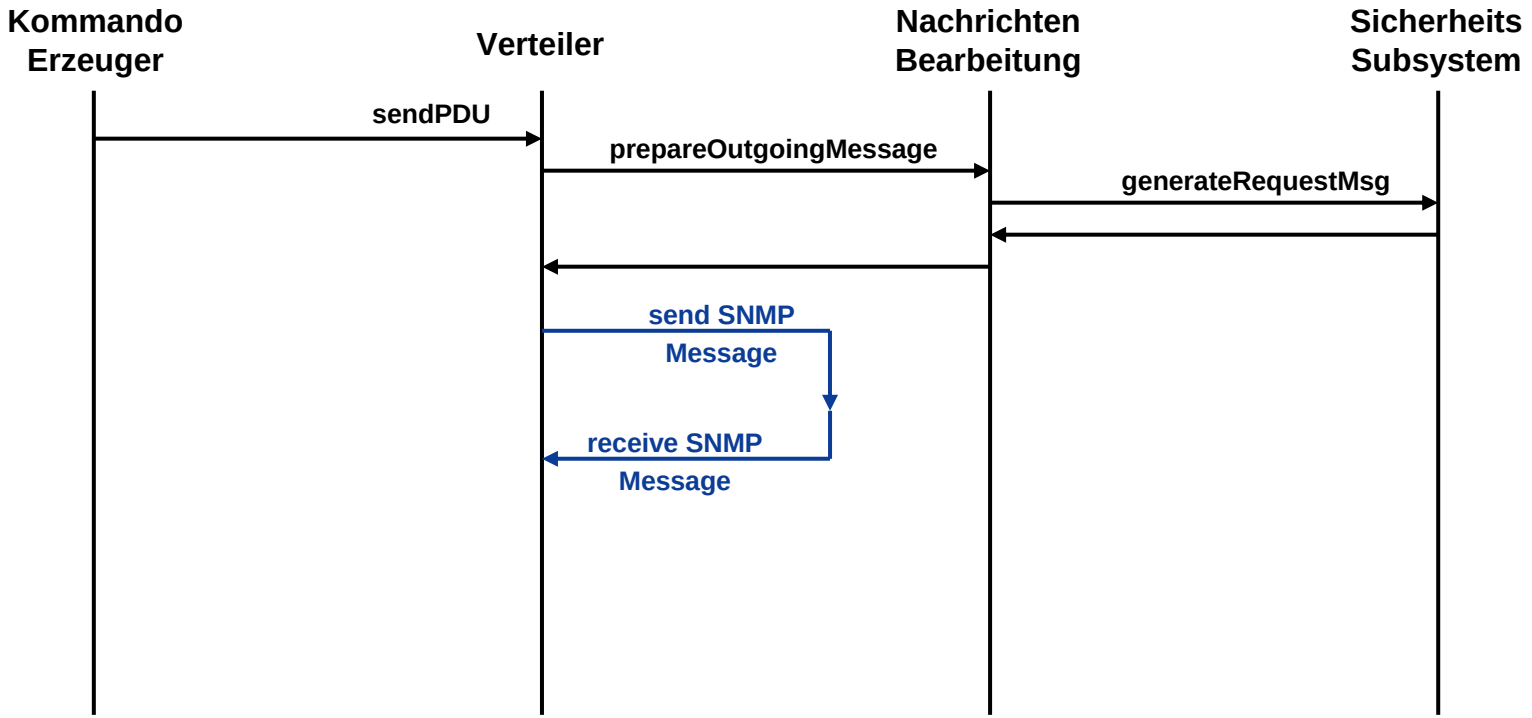
Zusammenspiel von Applikation und Subsystem
→ Initiieren einer SNMP Nachricht (3/7)



Falls Authentifizierung oder Verschlüsselung erforderlich ist, wird die SNMP Nachricht an das Sicherheitssubsystem weitergeleitet (generateRequestMsg).

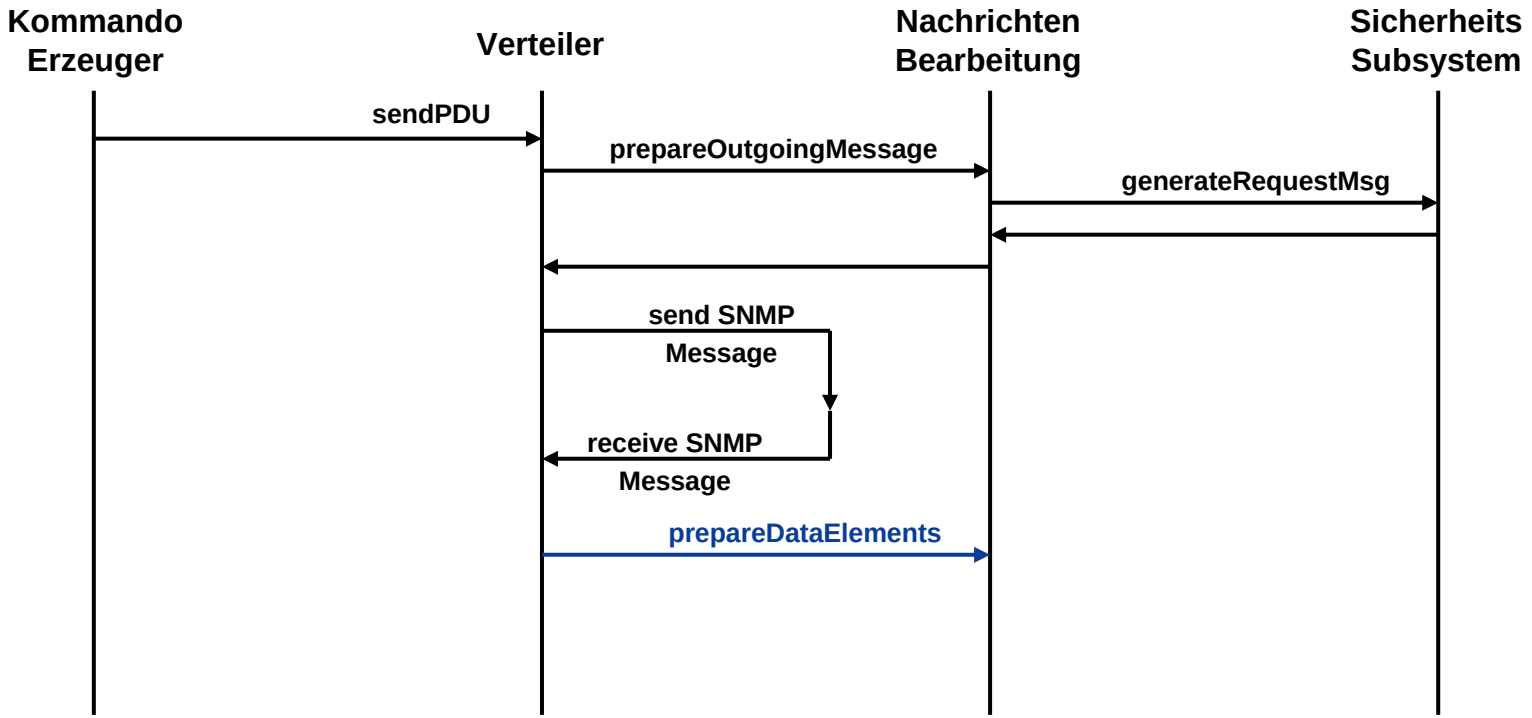
Das Sicherheitssystem führt je nach Sicherheitsstufe und -modell eine Authentifizierung und/oder Verschlüsselung durch, setzt die entsp. Sicherheitsparameter in der SNMP Nachricht und gibt diese wieder zurück.

Zusammenspiel von Applikation und Subsystem
→ Initiieren einer SNMP Nachricht (4/7)



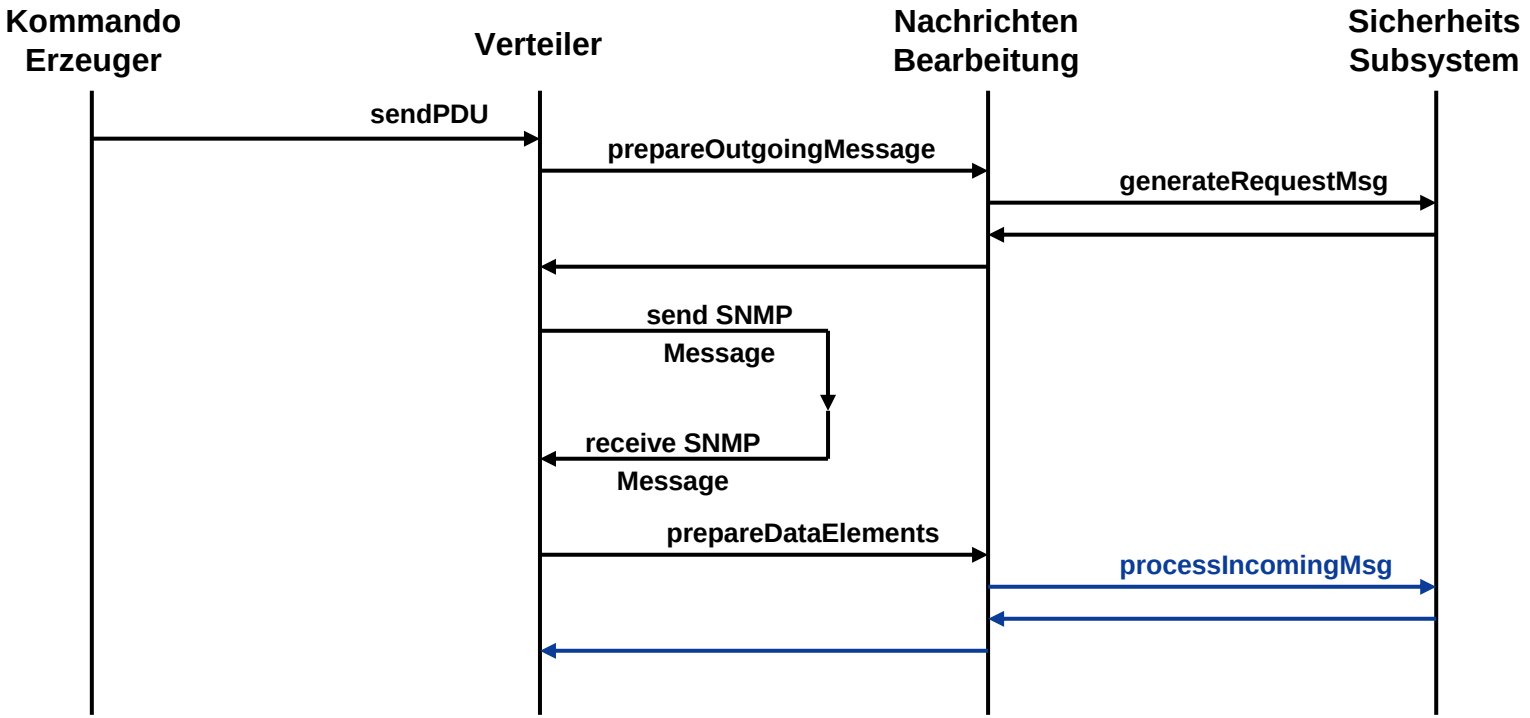
Der Verteiler überträgt nun die so generierte SNMP Nachricht auf das jeweilige Transportprotokoll und empfängt im Gegenzug die entsprechende Antwortnachricht.

Zusammenspiel von Applikation und Subsystem
→ Initiieren einer SNMP Nachricht (5/7)



Die Antwortnachricht übergibt der Verteiler nun an das Nachrichtenbearbeitungssystem (prepareDataElements).

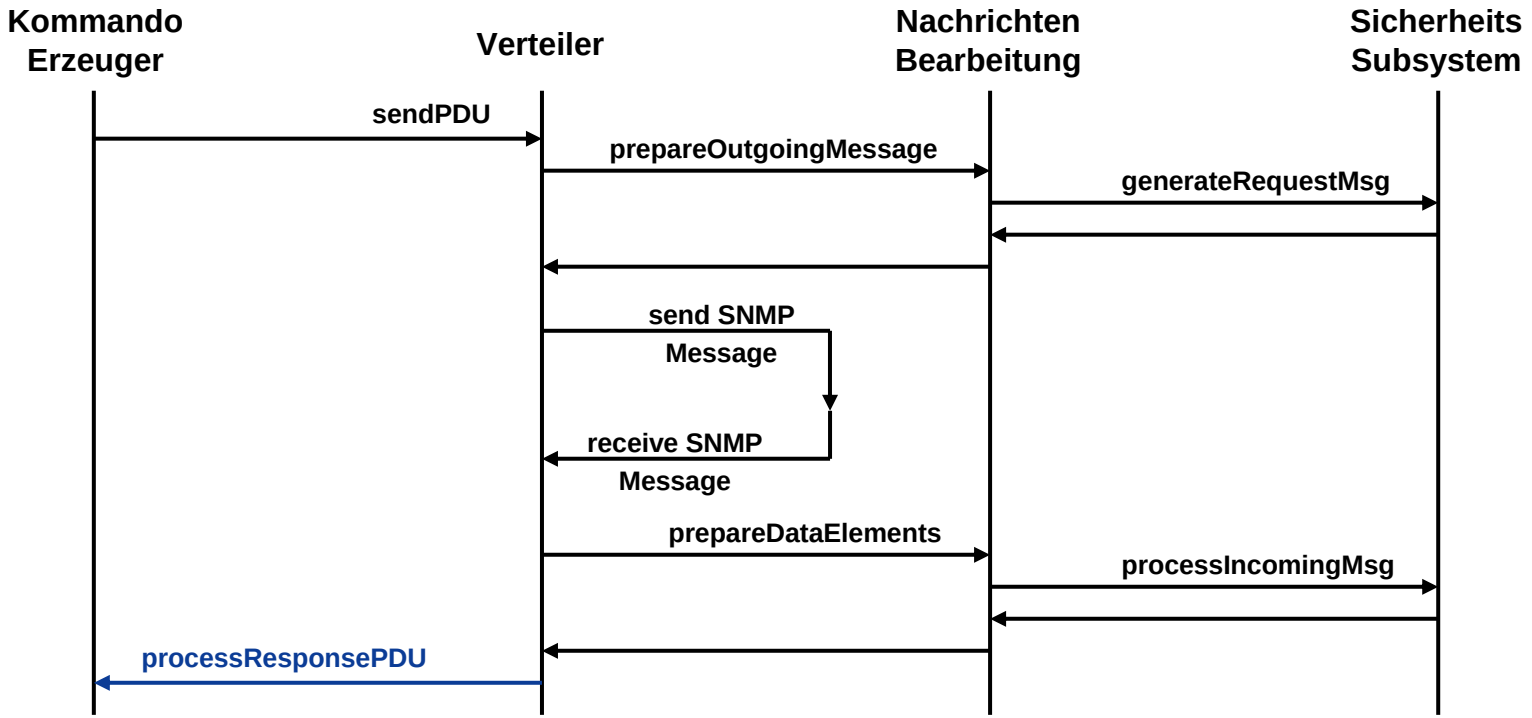
Zusammenspiel von Applikation und Subsystem
→ Initiieren einer SNMP Nachricht (6/7)



Das Nachrichtenbearbeitungsmodul überprüft, ob die Nachricht authentifiziert oder verschlüsselt wurde.
Falls ja, sendet sie die Nachricht an das Sicherheitssubsystem (processIncomingMsg), das die Nachricht entschlüsselt und die Authentizität überprüft.

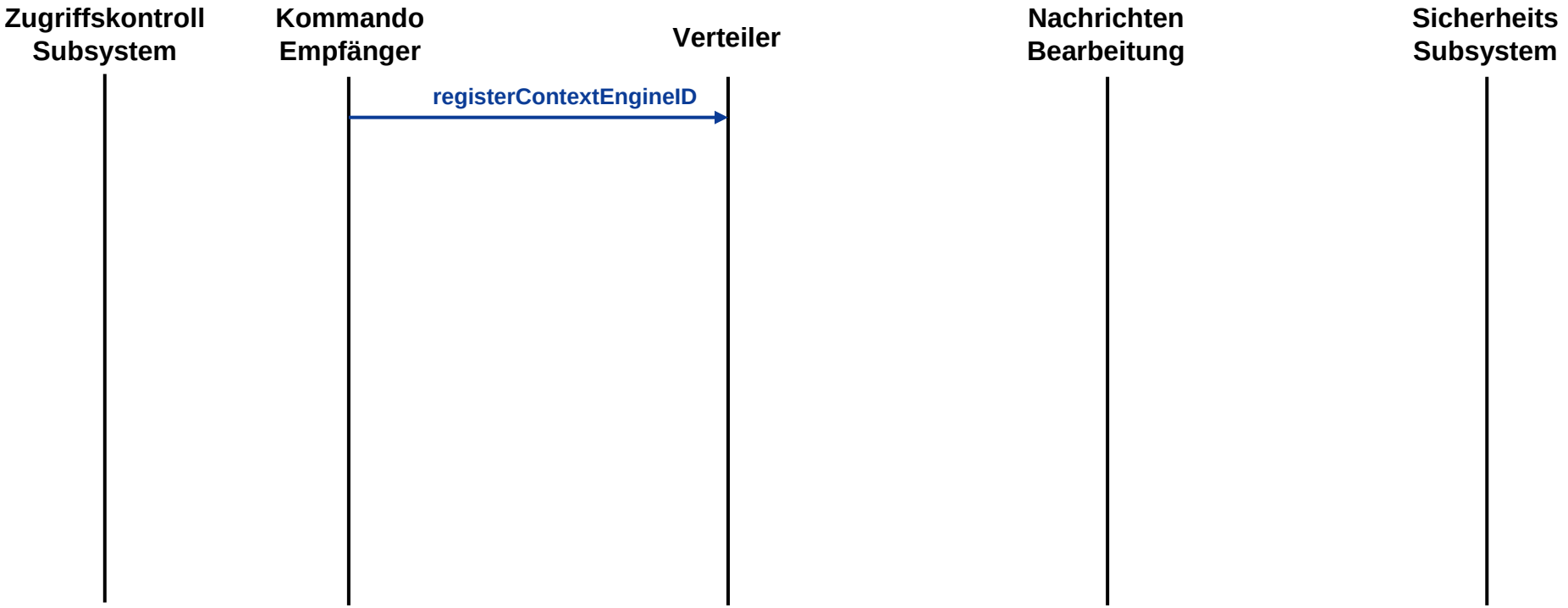
Anschließend extrahiert das Nachrichtenbearbeitungssubsystem die in der Nachricht enthaltene SNMP PDU und sendet diese an den Verteiler zurück.

Zusammenspiel von Applikation und Subsystem
→ Initiieren einer SNMP Nachricht (7/7)



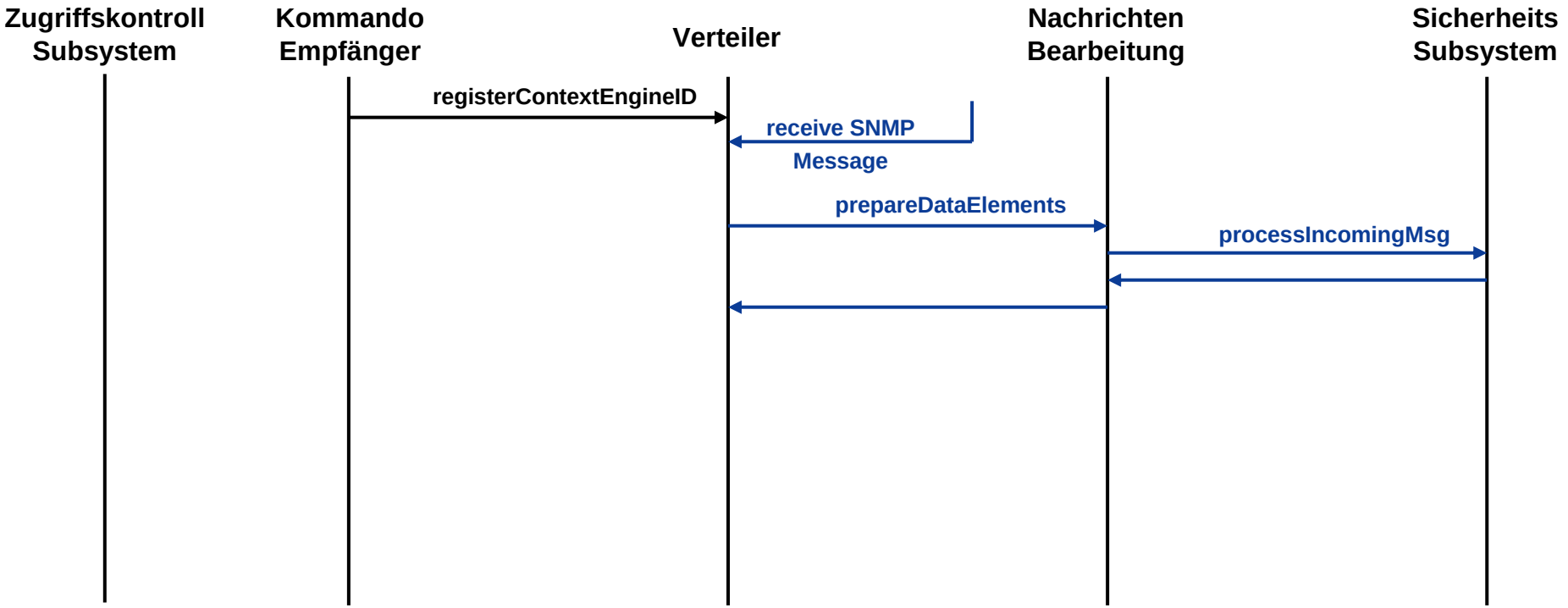
Der Verteiler sendet nun die so gewonnene SNMP PDU an die Kommandoerzeuger Applikation zurück (processResponsePDU), der die Antwort PDU entsp. bearbeitet.

Zusammenspiel von Applikation und Subsystem
→ Empfangen einer SNMP Nachricht (1/6)



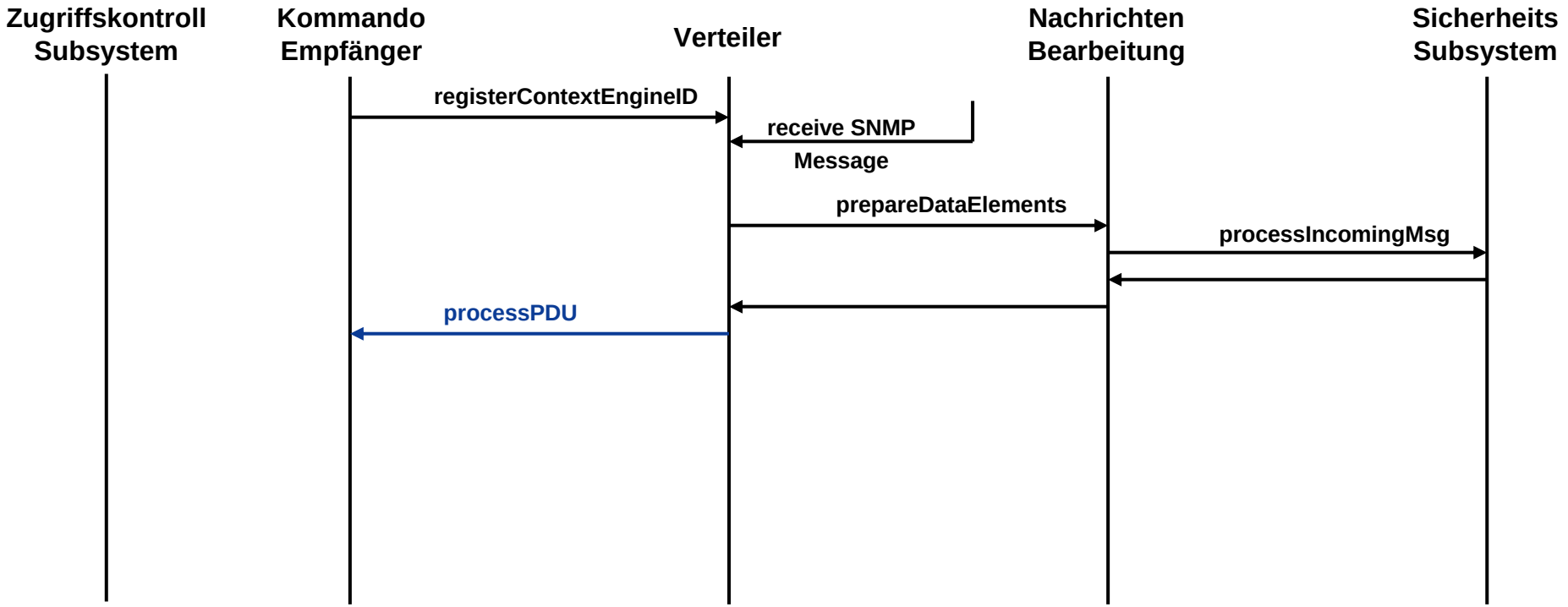
Bevor ein Kommandoempfänger SNMP PDUs vom Verteiler zugewiesen bekommt, muss er sich zunächst beim Verteiler dafür mit seiner SNMP *EngineID* anmelden (registerContextEngieID).

Zusammenspiel von Applikation und Subsystem
→ Empfangen einer SNMP Nachricht (2/6)



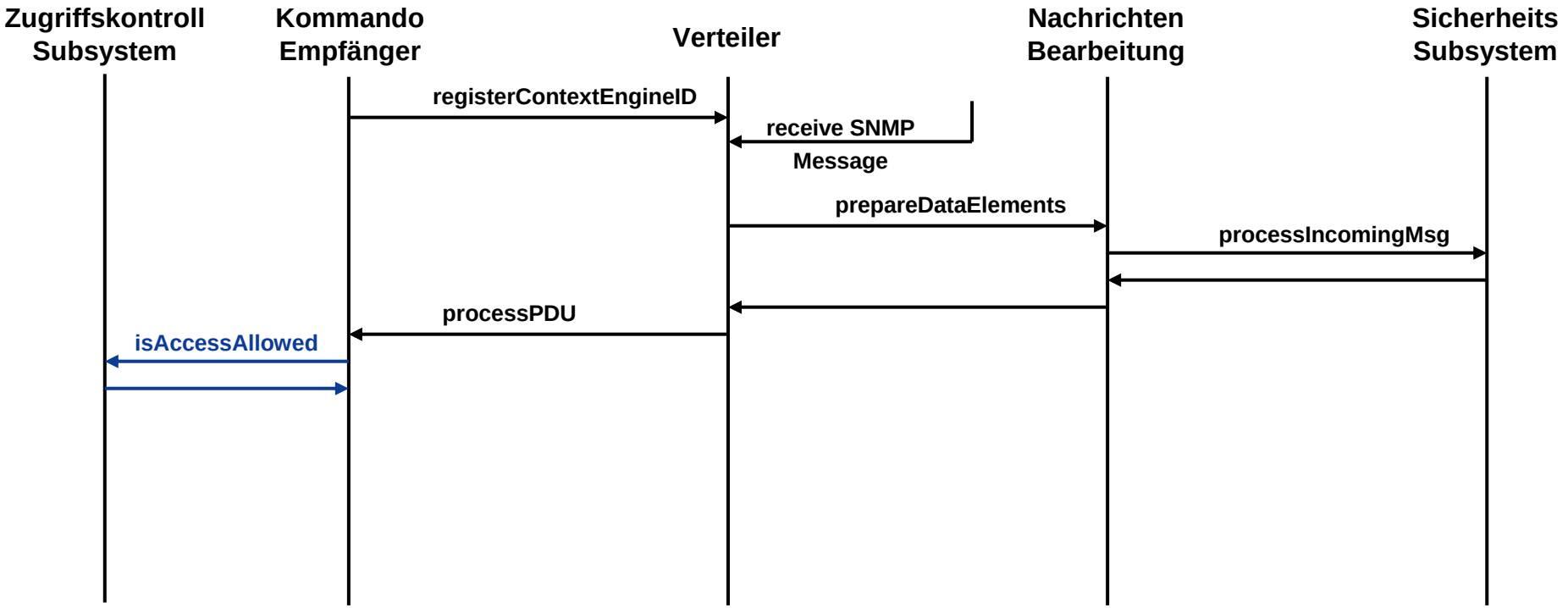
Empfängt der Verteiler eine SNMP Nachricht, so wird daraus mit Hilfe der anderen Subsysteme die SNMP PDU extrahiert.

Zusammenspiel von Applikation und Subsystem
→ Empfangen einer SNMP Nachricht (3/6)



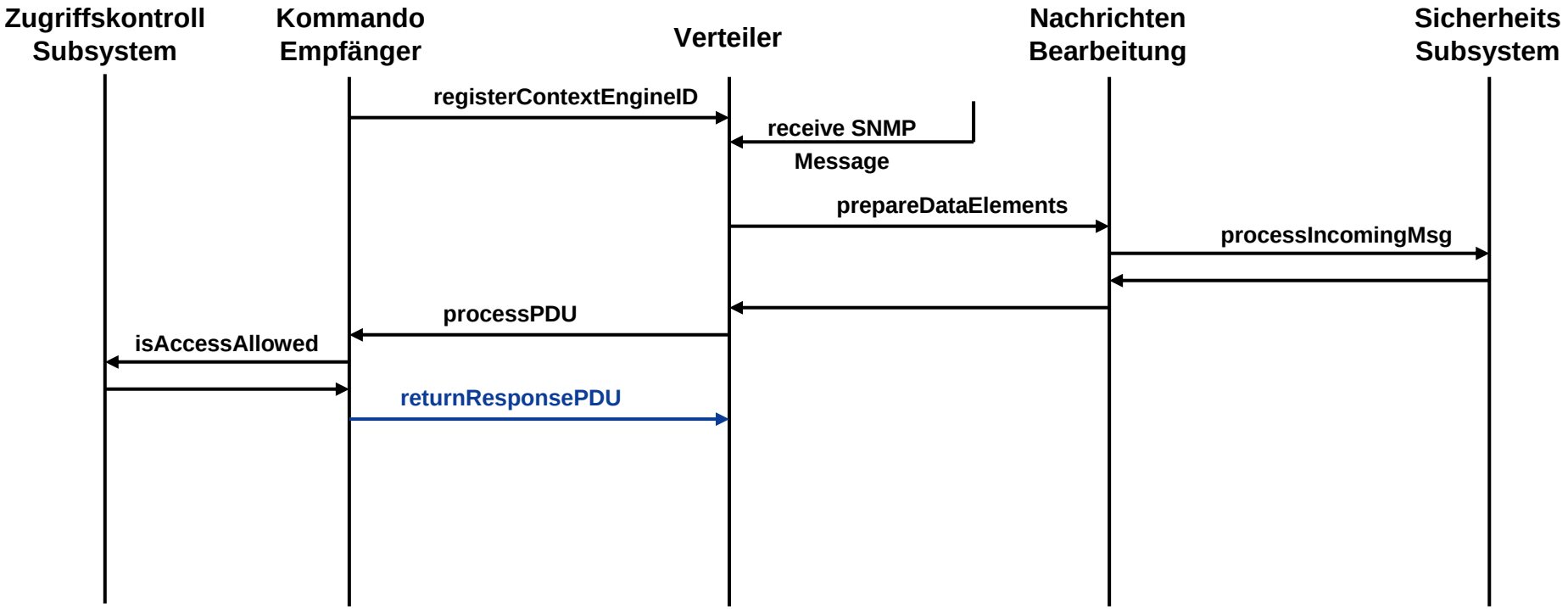
Der Verteiler ruf daraufhin die Funktion processPDU des Kommandoempfängers.

Zusammenspiel von Applikation und Subsystem
→ Empfangen einer SNMP Nachricht (4/6)



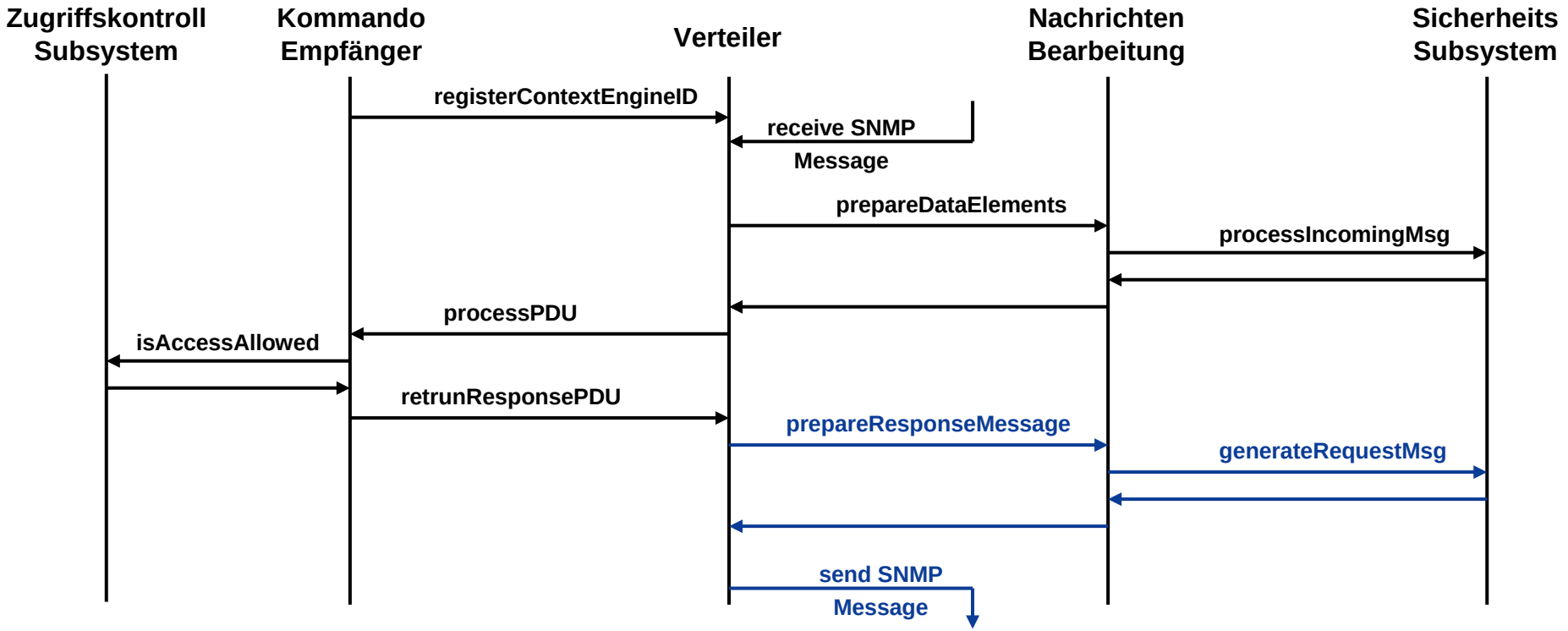
Der Kommdoempfänger überprüft zunächst, ob der Zugriff erlaubt ist, indem er die isAccessAllowed Funktion des Zugriffskontrollsubsystems aufruft.

Zusammenspiel von Applikation und Subsystem
→ Empfangen einer SNMP Nachricht (5/6)



Ist der Zugriff erlaubt, so bearbeitet der Kommandoempfänger die PDU entsprechend, generiert eine Antwort PDU und sendet diese an den Verteiler zurück (retrunResonsePDU).

Zusammenspiel von Applikation und Subsystem
→ Empfangen einer SNMP Nachricht (6/6)



Die Anwort PDU wird dann vom Verteiler an das Nachrichtenbearbeitungssubsystem weitergesendet (prepareResponseMessage), das eventuell mit Hilfe des Sicherheitssubsystems eine SNMP Nachricht daraus generiert und an den Verteiler zurückgibt.

Dieser überträgt nun die SNMP Nachricht auf das entsprechende Transportprotokoll.

Das benutzerbasierte Sicherheitsmodell (USM)

- Das benutzerbasierte Sicherheitsmodell für SNMPv3 wurde in RFC2274 definiert und umfasst die folgende Punkte:
  - **Aktualität:** Das SNMP Protokoll basiert typischerweise auf einem verbindungslosen Netzwerkdienst.
    Dadurch können Nachrichten verzögert, umgeordnet oder mehrmals wiedergegeben werden
  - **Authentifizierung:** Mit der Authentisierung einer Nachricht wird die Identität der sendenden Einheit sichergestellt. Weiterhin garantiert eine Authentisierung, dass die Nachricht nicht modifiziert wurde.
  - **Verschlüsselung:** Mit einer Verschlüssung einer Nachricht wird sichergestellt, dass es Unberechtigten nicht möglich ist, die ursprüngliche Nachricht mitzulesen.
  - **Schlüsselverwaltung:** Authentifikation und Verschlüsselung setzen voraus, dass beide SNMP Einheiten dieselben Schlüssel kennen.
    Dies wiederum bedingt, das sichere Übertragen von geheimen Schlüsseln über das Netzwerk sowie das Speichern von Schlüsseln in den lokalen SNMP-Einheiten.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Das Konzept der maßgebenden SNMP Einheit

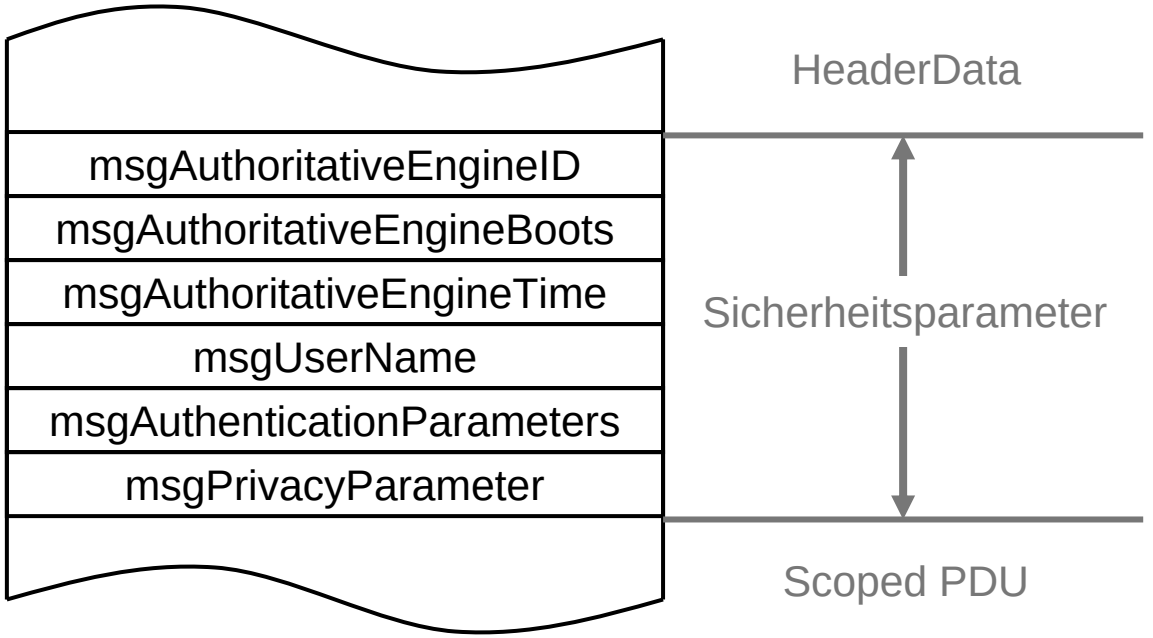
- Wann immer zwei SNMP Einheiten Nachrichten austauschen, übernimmt eine von beiden SNMP Einheiten die Rolle der **maßgebenden** (authoritative) SNMP Einheit:
  - Bei SNMP Nachrichten, die eine Antwort erfordern (wie z.B. get, set oder inform Operationen), ist der Empfänger die maßgebende Einheit.
  - Bei SNMP Nachrichten, die keine Antwort erfordern (wie z.B. trap Operationen), ist der Sender der Nachricht die maßgebende Einheit.
- Die Unterscheidung zwischen einer maßgebenden und nicht-maßgebenden SNMP Einheit spielt vor allen bei Aktualität einer SNMP Nachricht, sowie bei der Schlüsselverwaltung eine Rolle.

Das benutzerbasierte Sicherheitsmodell (USM)

Warum benötigen wir IT-Sicherheit beim Netzwerkmanagement ?

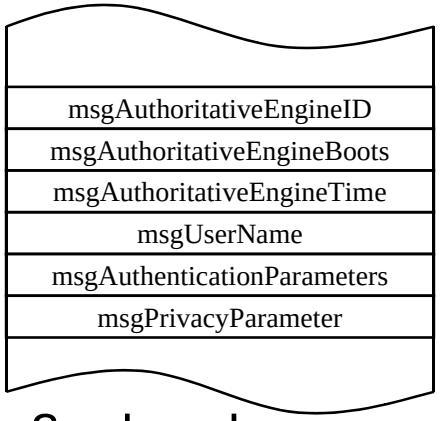
- **Hacker:** brechen in Netzwerke ein, weil sie darin eine Herausforderung sehen und mit dem Erfolg ihren Status vergrößern wollen. Oft handelt es sich um Jugendliche, die aus »Spieltrieb«, also ohne böse Absicht handeln.
  Sie sind aber unberechenbar und können hohen Schaden verursachen.
- **IT-Spione:** bezahlte Spezialisten – teilweise mit einem sehr hohen Budget – versuchen, über gezielte Angriffe an Informationen zu kommen.
  Ihre Ziele sind politisch oder auch wirtschaftlich begründet (siehe »Echelon«).
- **IT-Terroristen:** können Netzwerke angreifen, um aus politischen Gründen Angst und Chaos zu verursachen.
- **Unternehmens-Cracker:** dies sind Mitarbeiter, die auf Netzwerke von Konkurrenzunternehmen zugreifen, um ihrem Unternehmen finanzielle Vorteile zu schaffen. Dazu spähen sie beispielsweise Strategien aus.
- **Professionelle Kriminelle:** diese Personen wollen sich mit Angriffen persönlich bereichern, beispielsweise durch die nicht bezahlte Nutzung von Dienstleistungen.
- **Vandalen:** sind Personen, die Angriffe durchführen, um Organisationen oder Personen gezielt Schaden zuzufügen.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Die Sicherheitsparameter (1/2)



- Die Sicherheitsparameter einer SNMPv3 Nachricht werden ausschließlich vom jeweiligen Sicherheitssubsystem interpretiert, bzw. gesetzt.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Die Sicherheitsparameter (2/2)



- **msgAuthoritativeEngineID**
  SNMP-EngineID der maßgebenden SNMP Einheit.
- **msgAuthoritativeEngineBoots**
  Anzahl der Bootvorgänge in der maßgebenden SNMP Einheit.
- **msgAuthoritativeEngineTime**
  Zeit in Sekunden seit dem letzten Bootvorgang in der maßgebenden SNMP Einheit.
- **msgUserName**
  Name des Anwenders, der die SNMP Nachricht initiiert.
- **msgAuthenticationParameters**
  Ist leer, falls keine Authentifizierung vorgenommen wurde, andernfalls enthält dieser Parameter einen Authentifizierungscode (Hash-Wert).
- **msgPrivacyParameters**
  Falls die Nachricht verschlüsselt wurde, enthält dieser Parameter einen für das Verschlüsselungsprotokoll spezifischen Anfangswert (Initialisierungsvektor), ansonsten ist dieser Parameter leer.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Aktualität einer SNMP Nachricht (1/4)

- Die Aktualität einer SNMP Nachricht wird mit Hilfe zweier Zähler snmpEngineBoots und snmpEngineTime sichergestellt.
- Der erste Zähler gibt die Anzahl der Bootvorgänge an, während der zweite Zähler die Anzahl der Sekunden seit dem letztem Bootvorgang enthält.
- Eine SNMP Einheit im maßgebenden Modus muss beide Zähler verwalten und ständig aktualisieren.
- Wenn zwei SNMP Einheiten miteinander kommunizieren, so muss sich die SNMP Einheit im nicht-maßgebenden Modus mit der anderen Einheit ständig synchronisieren.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Aktualität einer SNMP Nachricht (2/4)

- Dazu verwaltet die SNMP Einheit im nicht-maßgebenden Modus für jede ihr bekannte *maßgebende* Einheit lokale Kopie der folgenden Variablen:
  - snmpEngineBoots: Der neuste geschätzte Wert des snmpEngineBoots Zählers der maßgebenden SNMP Einheit.
  - snmpEngineTime: Der geschätzte Wert des snmpEngineTime Zählers der maßgebenden SNMP Einheit.
    Dieser Wert wird von der nicht-maßgebenden SNMP Einheit jede Sekunde inkrementiert.
  - latestReceivedEngineTime: Der letzte empfangene Wert des snmpEngineTime Zählers der maßgebenden SNMP Einheit.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Aktualität einer SNMP Nachricht (3/4) → Synchronisierung

- Um eine lose Synchronisation zwischen zwei SNMP Einheiten zu gewährleisten, sendet die SNMP Einheit im maßgebenden Modus die aktuellen Werte von snmpEngineTime und snmpEngineBoots, sowie ihre SNMP EngineID in jeder SNMP Nachricht mit.
- Falls die Nachricht authentisch ist und die folgende Bedingung erfüllt ist, aktualisiert die nicht-maßgebende SNMP Einheit ihre eigenen Kopien:

(msgAuthoritativeEngineBoots > snmpEngineBoots) OR
[(msgAuthoritativeEngineBoots = snmpEngineBoots) AND
(msgAuthoritativeEngineTime > latestReceiveEngineTime)]

- Eine Synchronisation zwischen zwei SNMP Einheiten findet nur dann statt, wenn die SNMP Nachrichten authentifiziert wurde.
- Diese Einschränkung ist erforderlich, da nur dann gewährleistet ist, dass die Werte der Zähler gültig sind.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Aktualität einer SNMP Nachricht (4/4) → Zeitfenster

- SNMPv3 definiert ein Zeitfenster, in dem SNMP Nachrichten empfangen werden müssen.
- Dieses Zeitfenster hängt vom Modus der SNMP Einheit ab.
- SNMP Einheiten im maßgebenden Modus betrachten Nachrichten außerhalb dieser Zeitfenster, falls folgende Bedingung zutrifft:

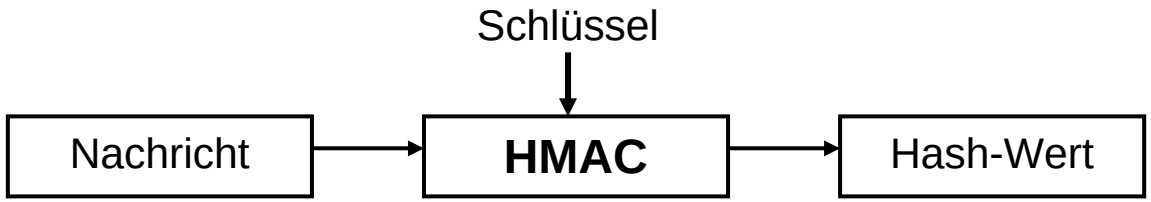
  snmpEngineBoots = 2^{31} -1 OR
  msgAuthoritativeEngineBoots != snmpEngineBoots OR
  msgAuthoritativeEngineTime NOT IN [snmpEngineTime-150, snmpEngineTime+150]

- Für SNMP Einheiten im nicht-maßgebenden Modus wird eine SNMP Nachrichten zurückgewiesen, falls folgende Bedingung zutrifft:

  snmpEngineBoots = 2^{31} -1 OR
  msgAuthoritativeEngineBoots >= snmpEngineBoots OR
  [(msgAuthoritativeEngineBoots = snmpEngineBoots) AND
  (msgAuthoritativeEngineTime < snmpEngineTime-150)]

- Damit ist die Überprüfung der Aktualität in einer nicht-maßgebenden SNMP Einheit toleranter.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Authentikation einer SNMP Nachricht (1/2)



- Die **Authentifizierung** einer SNMP Nachricht wird in SNMPv3 durch den Einsatz von zwei alternativen Verfahren erreicht: HMAC-MD5-96 und HMAC-SHA-96 (siehe RFC2104).
- **HMAC** ist im Internet ein weit verbreiteter Algorithmus zur Authenifizierung von Nachrichten und erlaubt die Einbindung verschiedener **Hash-Funktionen**.
- Eine Hash-Funktion erzeugt aus einer beliebig langen Nachricht einen sogenannten Hash-Wert mit fester Größe als Ausgabe.
- Eine identische Nachricht erzeugt immer einen identischen Hash-Wert, wobei es unmöglich ist, aus dem Hash-Wert die Nachricht zurückzugewinnen.
- Das USM verwendet alternativ die Hash-Funktionen MD5 und SHA.

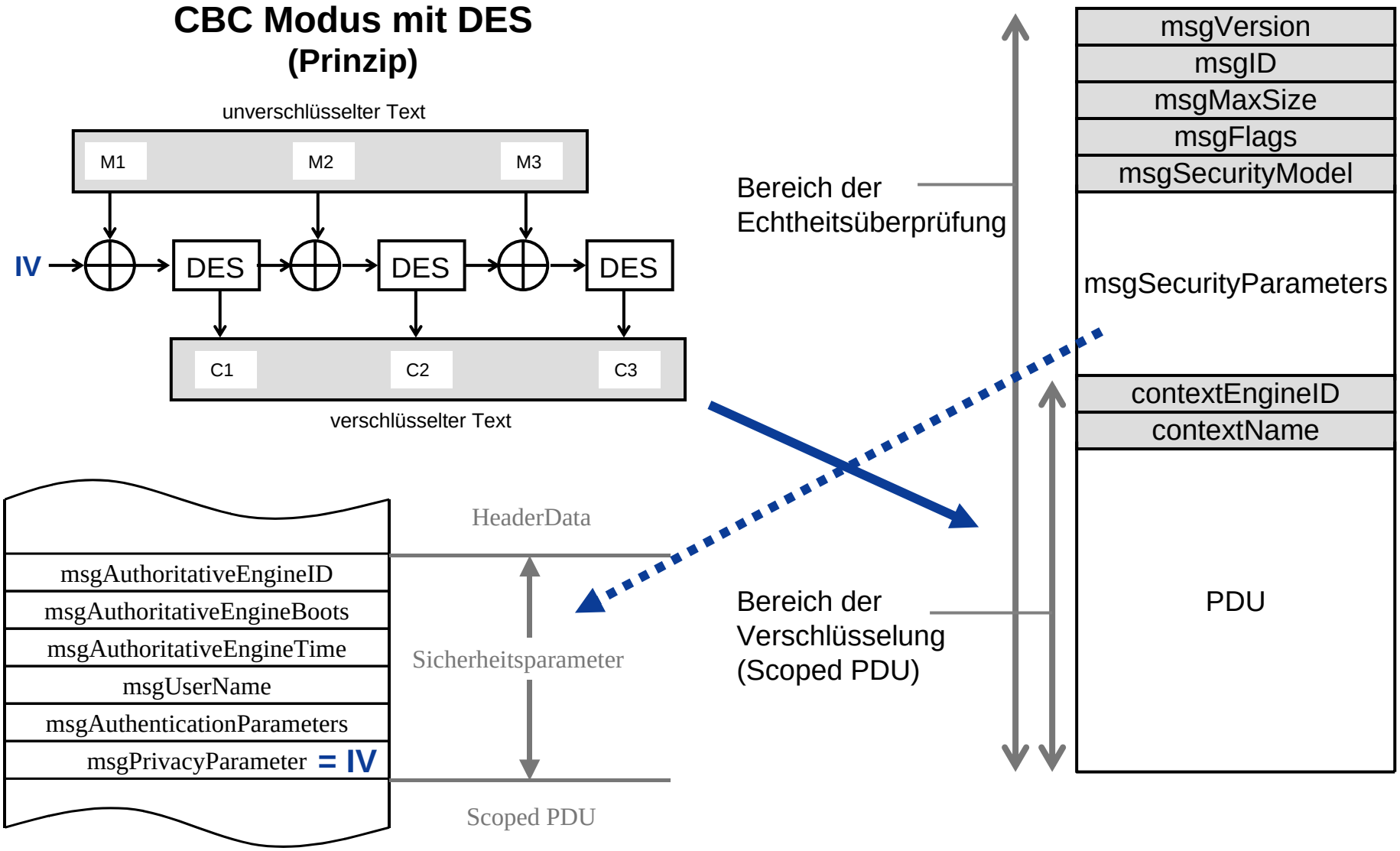
Das benutzerbasierte Sicherheitsmodell (USM)
→ Authentikation einer SNMP Nachricht (2/2)

- Die Authentifizierung einer SNMP Nachricht erfolgt in drei Schritten:
  - Der Sender generiert aus der zu sendenden Nachricht mit Hilfe eines geheimen Schlüssels den Hash-Wert.
  - Dieser Hash-Wert wird an die Nachricht angehängt und zusammen zum Empfänger geschickt.
  - Der Empfänger berechnet mit seinem Schlüssel und der empfangenen Nachricht denselben Hash-Wert und vergleicht nun beide.
- Wenn beide übereinstimmen, so lässt sich daraus schließen, dass die Nachricht nicht verändert wurde und dass die Nachricht vom richtigen Absender ist, da nur dieser den notwendigen Schlüssel kennt.
- Der vom Sender genierte Hash-Wert wird in dem Sicherheitsparameter msgAuthenticationParameters einer SNMP Nachricht übertragen.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Verschlüsselung einer SNMP Nachricht (1/2)

- Mit der Verschlüsselung einer SNMP Nachricht wird sichergestellt, dass unberechtigte Personen den ursprünglichen Inhalt der Nachricht „nicht“ lesen können.
- In SNMPv3 wurde der DES Algorithmus festgelegt.
- Um größere Datenmengen verschlüsseln zu können, müssen diese in 64 Bit Blöcke aufgeteilt werden.
- Der in SNMPv3 festgeschriebene CBC (*Ciper Block Chaining*) Modus gewährleistet, dass wiederholende 64 Bit Blöcke zu einem unterschiedlichen verschlüsselten Text führen, um eine mögliche Entschlüsselung aufgrund von wiederkehrenden Mustern zu erschweren.
- Der CBC Modus benötigt einen Anfangswert IV (*initial value*) für die Verschlüsselung.
- Dieser Anfangswert wird über einen sogenannten *salt value* berechnet, der im Sicherheitsparameter msgPrivacyParameters der SNMPv3 Nachricht übertragen wird.
- Die Verschlüsselung betrifft nur den *scoped PDU* Teil einer SNMP Nachricht.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Verschlüsselung einer SNMP Nachricht (2/2)



Das benutzerbasierte Sicherheitsmodell (USM)
→ Schlüsselverwaltung (1/5)

- Das Sicherheitsmodell in SNMPv3 erfordert, dass für jede Kommunikation zwischen einem Manager und einem Agenten ein privater Schlüssel für die Authentifizierung und ein privater Schlüssel für die Verschlüsselung von SNMP Nachrichten auf beiden Seiten bekannt sein muss.
- RFC2274 definiert Richtlinien für die Erzeugung, die Aktualisierung und die Verwaltung von Schlüsseln.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Schlüsselverwaltung (2/5) → Erzeugen von Schlüsseln

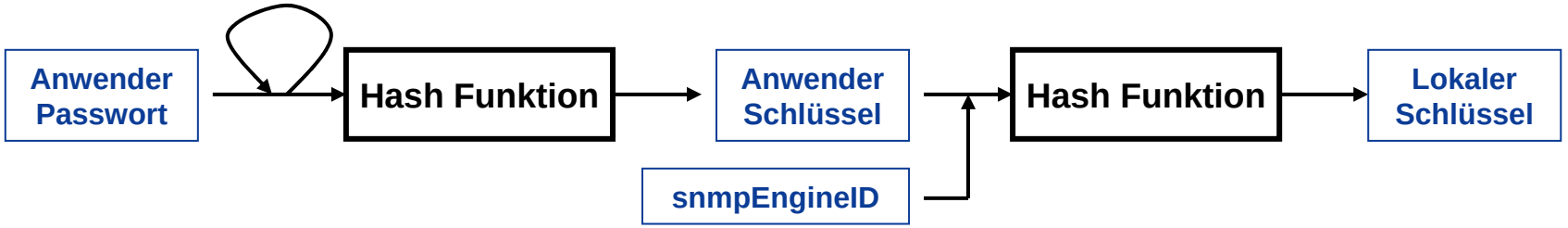
- Für die Authentifizierung und Verschlüsselung muss der Anwender 16/20 Byte lange Schlüssel zur Verfügung stellen.
- Anwender arbeiten jedoch generell mit Passwörtern, die einfach gelesen und behalten werden können.
- In RFC2274 wurde deshalb der folgende Algorithmus definiert.
  - Man nehme das Passwort des Anwenders und wiederhole das Passwort so lange bis eine Zeichenkette von 2^{20} Byte entsteht.
  - Wird ein 16 Byte Schlüssel benötigt, so nimmt man die MD5 Hash Funktion, wird ein 20 Byte langer Schlüssel benötigt, so nimmt man die SHA Hash Funktion.
  - Das Ergebnis der Hash-Funktion ist der Schlüssel für die Authentifizierung bzw. Verschlüsselung.
- **Problematisch, da Angriff auf Passwort möglich ist ! ! !**
- **Die Gesamtsicherheit hängt von der Qualität des Passwortes ab.**

Das benutzerbasierte Sicherheitsmodell (USM)
→ Schlüsselverwaltung (3/5) → Verwaltung von Schlüsseln

- Aus Gründen der Sicherheit, muss jeder Anwender für jeden Agenten einen geheimen Schlüssel für die Authentifizierung und Verschlüsselung besitzen.
- Um die Anwender nicht mit einer Vielzahl von verschiedenen Schlüsseln zu belasten, erlaubt SNMPv3, dass jeder Anwender nur jeweils einen Schlüssel besitzen muss.
- Gelingt es aber nun einem Angreifer, diesen einen Schlüssel herauszufinden, so ist er in der Lage, Managementzugriffe auf allen Agenten auszuführen.
- Um diese zu verhindern, werden für jeden Agenten unterschiedliche Schlüssel verwendet, ohne jedoch den Anwender mit einer Vielzahl von unterschiedlichen Passwörter zu belasten.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Schlüsselverwaltung (4/5) → Lokale Schlüssel

- Verfahren zur Berechnung von lokalen Schlüsseln
  - Man nehme den vom Anwenderpasswort erzeugten Schlüssel und füge daran den Wert der Variablen snmpEngineID des Agenten an.
  - Falls ein 16 Byte langer Schlüssel benötigt wird, nimmt man die MD5 Hash Funktion, wird ein 20 Byte langer Schlüssel benötigt, so nimmt man die SHA Hash Funktion
  - Das Ergebnis ist der lokale Schlüssel, der für jeden Agenten unterschiedlich ist.



- Nur dieser lokale Schlüssel wird im Agenten gespeichert.
- Die erstmalige Installation des lokalen Schlüssels muss aus Sicherheitsgründen vor Ort geschehen.

Das benutzerbasierte Sicherheitsmodell (USM)
→ Schlüsselverwaltung (5/5) → Aktualis. von Schlüsseln

- Algorithmus zur Aktualisierung der lokalen Schlüssel:
  - Der Manager erzeugt eine beliebige Zufallszahl: random
  - Berechne den Wert: digest
    digest = Hash (keyOld || random)
  - Berechne daraus:
    protocolKeyChange = (random || ((digest XOR keyNew) = **delta**))
- Der Wert von protocolKeyChange wird dann in einem set Befehl über das Netzwerk gesendet.
- Um den Schlüssel zu extrahieren, muss der Agent folgendes ausführen:
  - Berechne den Wert: digest
    digest = Hash (keyOld || random)
  - Berechne daraus den neuen Schlüssel:
    keyNew = digest XOR **delta**
  - Der Agent trägt den neuen Schlüssel in seine Tabelle (usmUser Gruppe)

Das sichtenbasierte Zugriffkontrollmodell (VACM)

Das sichtenbasierte Zugriffkontrollmodell

- SNMPv3 spezifiziert in RFC2574 das sichtenbasierte Zugriffskontrollmodell (*View-based access control model*).
- Die Zugriffskontrolle wird bei SNMPv3 immer dann in Anspruch genommen, wenn von außen auf ein Objekt lesend oder schreibend zugegriffen wird.
- Das Zugriffskontrollmodell stellt seine Dienste direkt den Applikationen Kommandobeantworter und Meldeerzeuger zur Verfügung.

Das sichtenbasierte Zugriffkontrollmodell
→ Elemente (1/4) → Gruppen (groups)

- Eine Gruppe ist definiert als ein Satz von <securityModel, securityName> Paaren.
- Der Parameter securityModel identifiziert das eingesetzte Sicherheitsmodell (z.B. USM o. *Community*-basiert) und securityName ist der Anwender (*principal*), der die SNMP Nachricht initiiert hat.
- Eine Gruppe wird eindeutig durch einen Gruppennamen identifiziert.
- Alle Anwender einer Gruppe besitzen dieselben Rechte, wobei ein <securityModel, securityName> Paar immer nur Teil einer Gruppe sein kann.

Das sichtenbasierte Zugriffkontrollmodell
→ Elemente (2/4) → Sicherheitsstufe (security level)

- SNMPv3 definiert drei verschiedene Sicherheitsstufen
  - noAuthNoPriv,
  - authNoPriv und
  - authPriv.
- Für jede Sicherheitsstufe können unterschiedliche Zugriffsrechte definiert werden.
- So ist es z.B. möglich, schreibende Zugriffe nur zuzulassen, falls die Nachrichten authentifiziert und verschlüsselt wurden.

Das sichtenbasierte Zugriffkontrollmodell
→ Elemente (3/4) → Kontext (context)

- Ein MIB Kontext identifiziert den Kontext, unter dem der Zugriff auf die *Managed Objects* einer MIB erfolgt.
- Ein MIB Kontext hat die folgenden Eigenschaften:
  - Eine SNMP Einheit kann mehr als einen Kontext verwalten.
  - Ein Objekt oder eine Objektinstanz kann in mehr als einem Kontext erscheinen.
  - Falls eine SNMP Einheit mehr als ein Kontext besitzt, muss zusätzlich der contextName angegeben werden.
- Als Beispiel für die Verwaltung des Kontext Konzeptes stelle man sich eine SNMP Einheit vor, die Managementinformationen für zwei Repeater bereitstellt.
- Dies könnte dann der Fall sein, wenn die Repeater selbst nicht SNMP fähig sind und so ein anderer Netzknoten als eine Art Proxy für die beiden Repeater arbeitet.
- In diesem Fall wird für jeden Repeater ein Kontextname definiert.

Das sichtenbasierte Zugriffkontrollmodell
→ Elemente (4/4) → MIB-Schichten (MIB views)

- Eine MIB-Schicht ist eine Untermenge von *Managed Objects* der lokalen MIB einer SNMP Einheit.
- Eine MIB-Schicht wird definiert als eine Sammlung von Teilbäumen, wobei ein Teilbaum entweder Teil der Sicht ist oder explizit von dieser Sicht ausgeschlossen ist.
- Ein Teilbaum ist einfach definiert als ein Knoten in der MIB Hierarchie sowie alle darunterliegenden Knoten.
- Damit besitzen alle Elemente innerhalb eines Teilbaums einen gemeinsamen *Object Identifier* Präfix.
- Der Teilbaum wird durch diesen Präfix benannt und identifiziert.

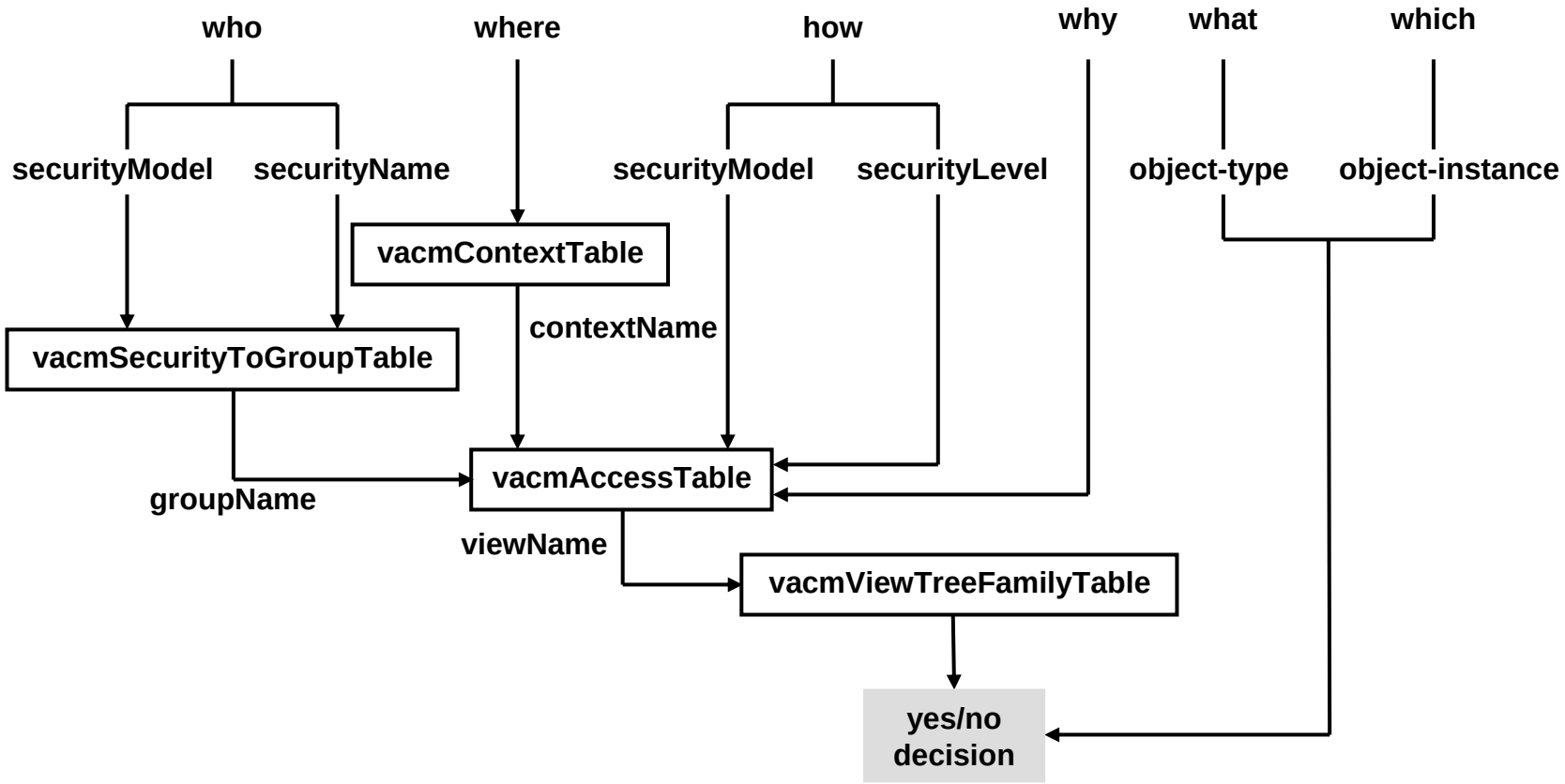
Das sichtenbasierte Zugriffkontrollmodell
→ Zugriffskontroll-Logik (1/9) → Eingabeparameter

- Die abstrakte Dienstschnittstelle des Zugriffskontrollmodul definiert nur eine Funktion isAccessAllowed mit den folgenden Eingabeparameter:
  - securityModel: Das verwendete Sicherheitsmodell (USM/*Community*)
  - securityName: Name des Anwenders für den der Zugriff überprüft werden muss
  - securityLevel: Sicherheitsstufe(noAuthNoPriv, authNoPriv, authPriv)
  - viewType: Bestimmt eine aus drei MIB-Schichten (read, write oder notify)
  - contextName: Name des Kontextes unter dem der Zugriff gewährt werden soll
  - variableName: *Object Identifier* und Instanz des Objektes

Das sichtenbasierte Zugriffkontrollmodell
→ Zugriffskontroll-Logik (2/9) → Eingabeparameter

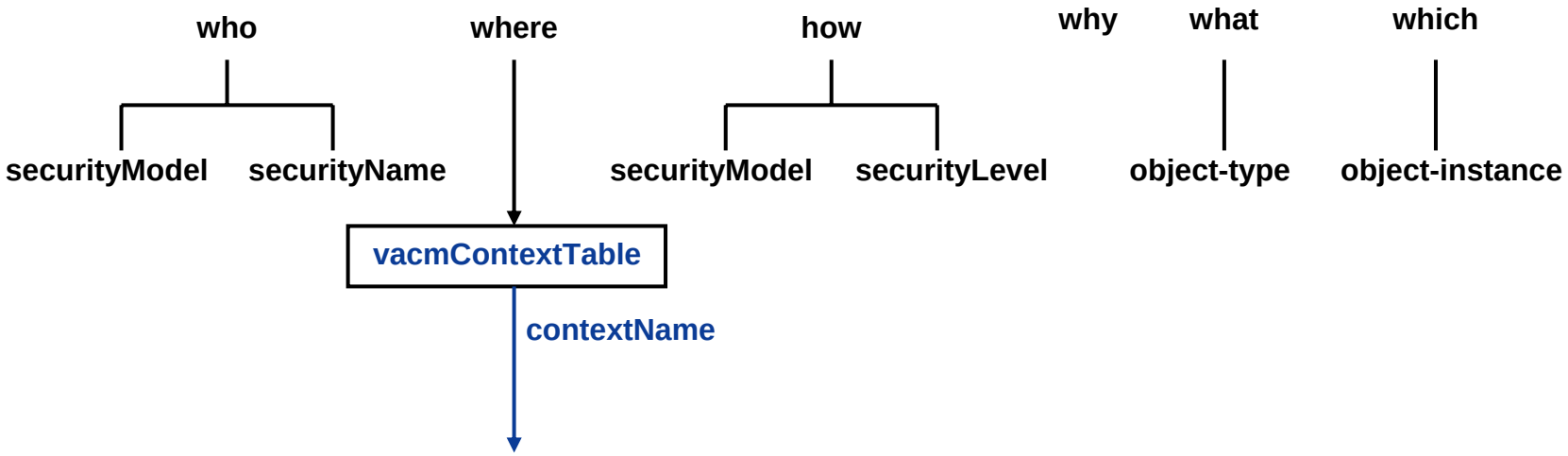
- Die Vielzahl der Eingabeparameter erlaubt es dem Netzwerkadministrator den Zugriff auf die MIB eines Agenten sehr fein zu granulieren.
- Die zum Teil umfangreiche Konfiguration eines Agenten erfolgt in einer eigens dafür definierten VACM-Management Information Base.
- Diese MIB enthält mehrere Tabellen, die Informationen über die dem Agenten bekannten Kontext, Gruppen und Views speichern.
- Der Netzwerkadmistrator hat die Möglichkeit, diese Tabellen über SNMP „remote“ zu konfigurieren.

Das sichtenbasierte Zugriffkontrollmodell
→ Zugriffskontroll-Logik (3/9) → Entscheidungsbaum



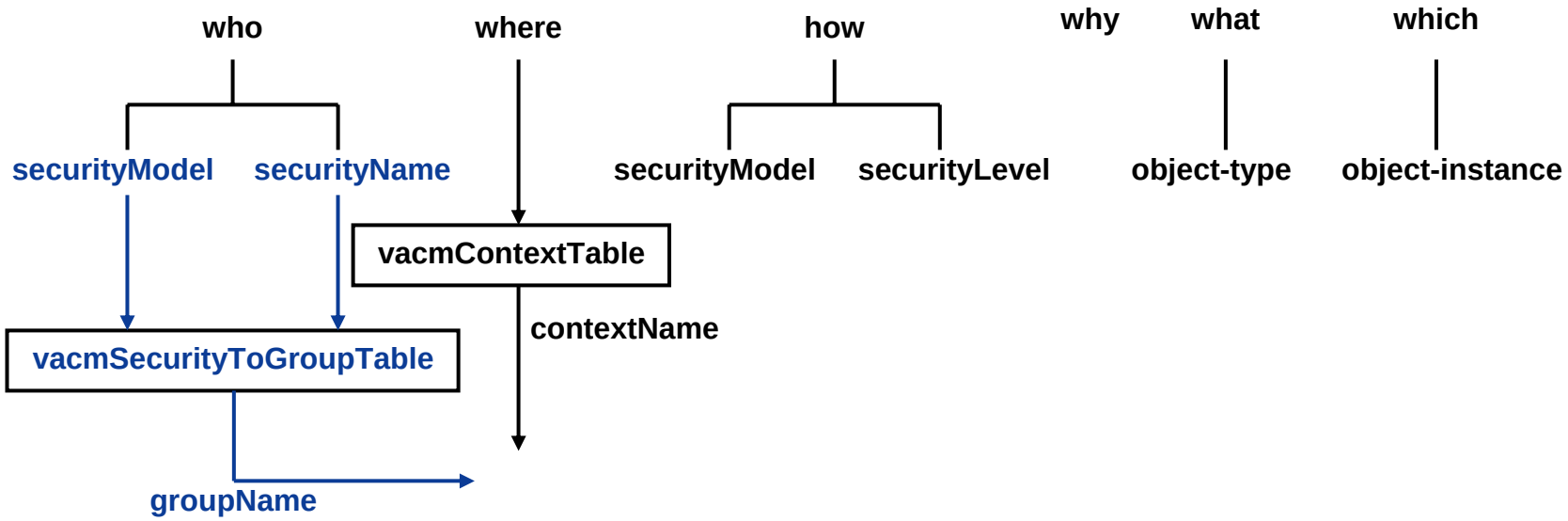
Wenn die Funktion isAccessAllowed aufgerufen wird, führt die VACM Zugriffskontrolle einige Schritte durch.

Das sichtenbasierte Zugriffkontrollmodell
→ Zugriffskontroll-Logik (4/9) → Entscheidungsbaum



Die Zugriffskontrolle überprüft, ob der Kontext contextName in der vacmContextTable enthalten ist.
Falls nicht wird der Fehler noSuchContext zurückgegeben.

Das sichtenbasierte Zugriffkontrollmodell
→ Zugriffskontroll-Logik (5/9) → Entscheidungsbaum

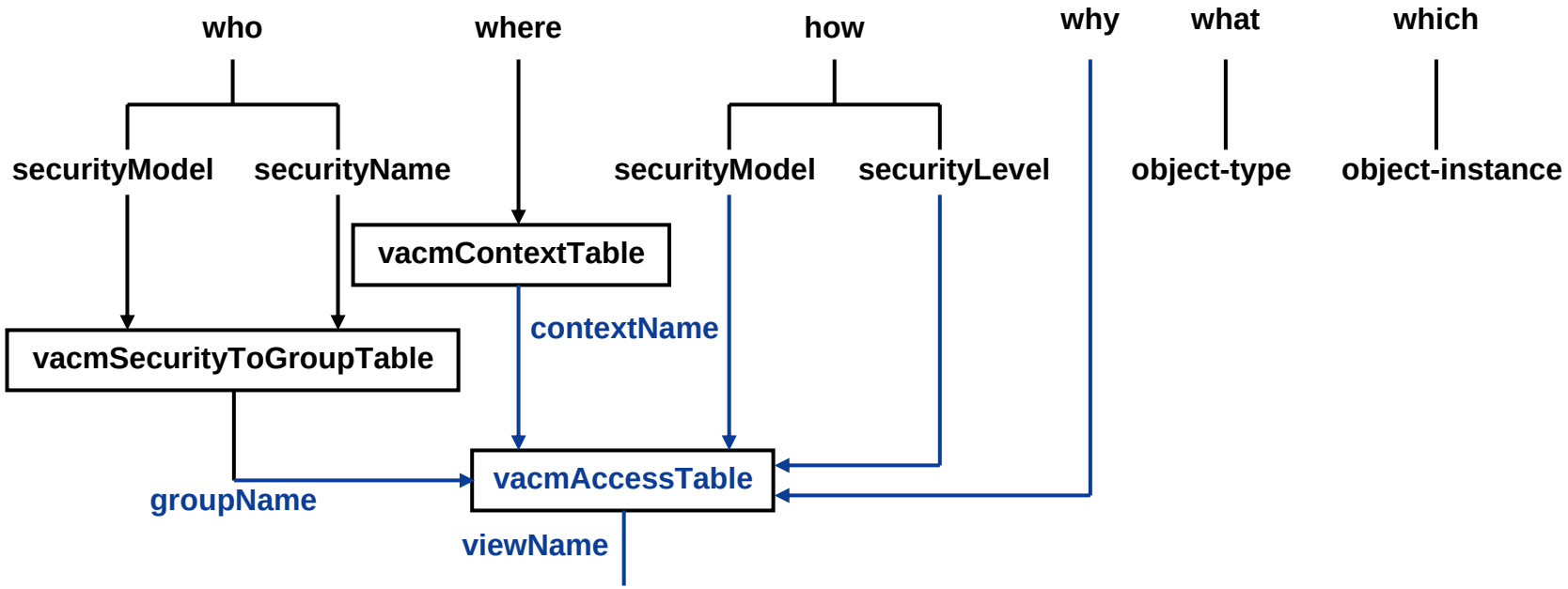


Die Zugriffskontrolle überprüft, ob für die beiden Parameter securityModel und SecurityName ein Eintrag in der vacmSecurityToGroupTable Tabelle vorhanden ist.

Falls ja, bedeutet dies, dass der Anwender unter dem angegebenen Sicherheitsmodell Teil einer Gruppe des Agenten ist.

Falls nicht, wird der Fehler noGroupName zurückgegeben und die Zugriffskontrolle abgebrochen.

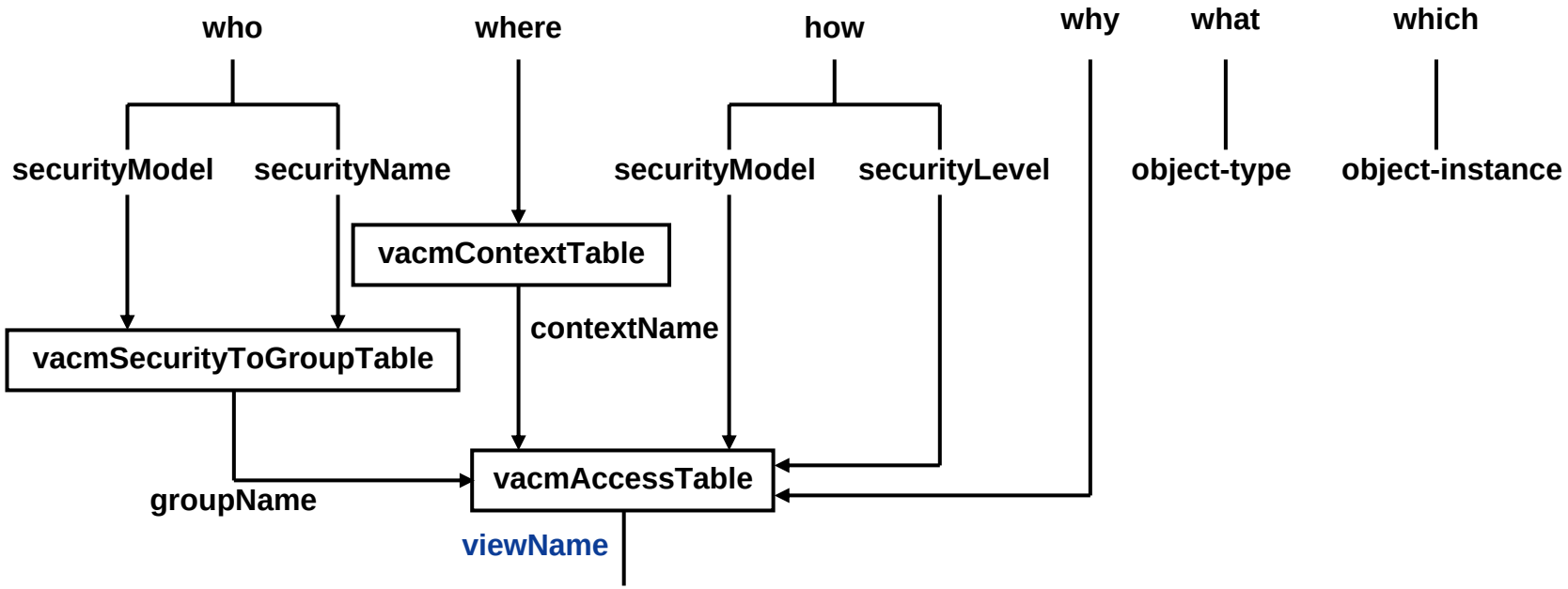
Das sichtenbasierte Zugriffkontrollmodell
→ Zugriffskontroll-Logik (6/9) → Entscheidungsbaum



Die Zugriffkontrolle überprüft die Tabelle vacmAccessTable mit den Parametern groupName, contextName, securityModel und securityLevel.

Falls kein Eintrag in der Tabelle gefunden wurde, wird der Fehler noAccessEntry zurückgegeben und die Zugriffskontrolle abgebrochen.

Das sichtenbasierte Zugriffkontrollmodell
→ Zugriffskontroll-Logik (7/9) → Entscheidungsbaum

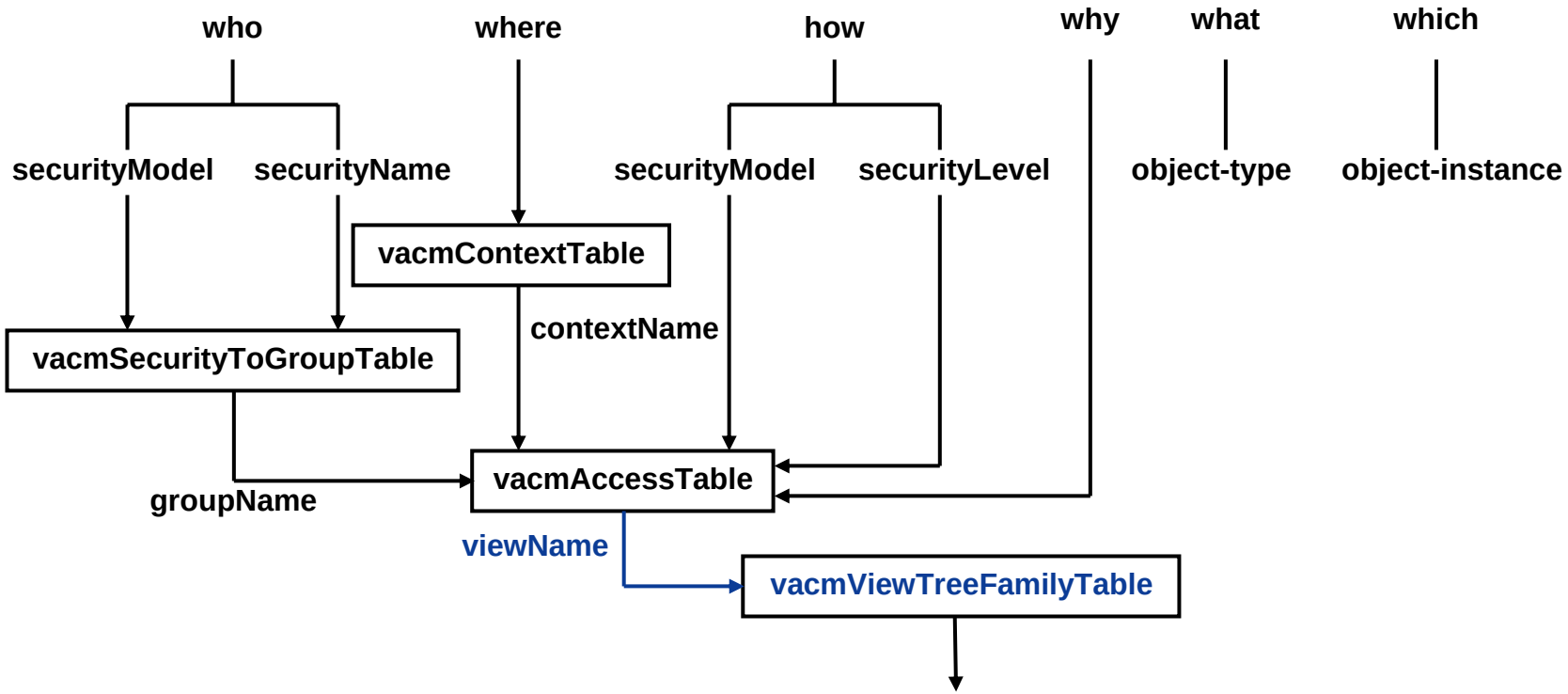


Die Zugriffskontrolle untersucht daraufhin, ob der Eintrag in der vacmAccessTable Tabelle eine Referenz auf einen MIB-Sicht vom Typ viewType enthält.

Der Parameter viewType gibt die Zugriffsart an und kann die Werte read/write/notify annehmen.

Falls der Eintrag keine entsprechende MIB-Sicht definiert, wird der Fehler noSuchView zurückgegeben und die Zugriffskontrolle abgebrochen.

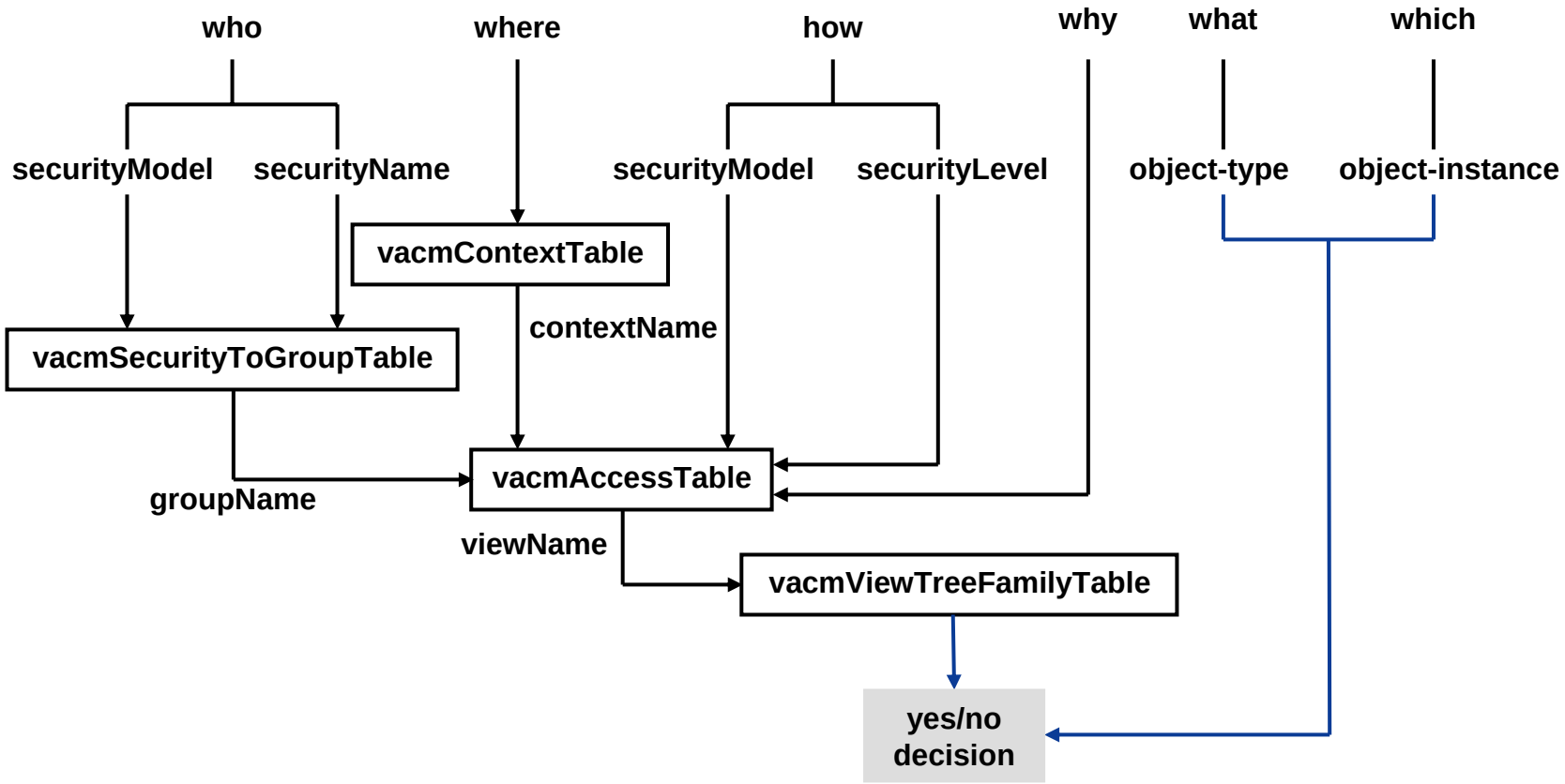
Das sichtenbasierte Zugriffkontrollmodell
→ Zugriffskontroll-Logik (8/9) → Entscheidungsbaum



Die Zugriffskontrolle benutzt den im vorhergehenden Schritt gefunden MIB-View Namen als Index für die vacmViewTreeFamilyTable Tabelle.

Falls kein Eintag gefunden werden konnte, wird der Fehler noSuchView zurückgegeben und die Zugriffskontrolle abgebrochen.

Das sichtenbasierte Zugriffkontrollmodell
→ Zugriffskontroll-Logik (9/9) → Entscheidungsbaum



Im letzten Schritt überprüft die Zugriffskontrolle, ob die Objektinstance variableName in der gefundenen View enthalten ist.

Wenn ja, wird der Wert accessAllowed zurückgegeben, andernfalls der Fehler notInView.

Netzwerkmanagement mit SNMP

→ Teil 4:SNMPv3

Vielen Dank für Ihre Aufmerksamkeit

Fragen ?

norbert.pohlmann@informatik.fh-gelsenkirchen.de

Fachhochschule Gelsenkirchen