

Web Service Security & Policy

Basis für eine sichere SOA

Detlef Sturm
Senior Manager
Product Technology
Beta Systems Software AG

SOA-Kongress 2008

Mainz, 29.10.2008



Beta Systems Software AG

- ▶ Softwareprodukte und -lösungen zur sicheren, effizienten und flexiblen Verarbeitung größter Informationsmengen
 - ▶ Data Processing,
Document Processing,
Security, Compliance
- ▶ Weltweit 1.300+ Kunden aus den Bereichen Banken, Versicherungen, Industrie, Handel und IT-Outsourcer mit besonderem Focus auf Financial Services

Umsatz:	96,6 Mio. € (2006)
Mitarbeiter:	> 630 weltweit
Headquarters:	Berlin

Zu meiner Person

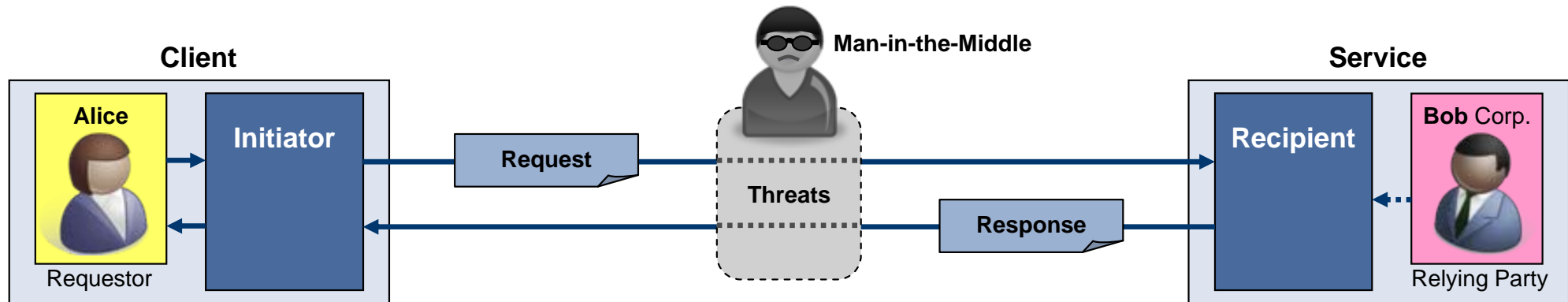
- ▶ Senior Manager in der Querschnittsabteilung „Product Technology“
- ▶ Verantwortlich für produktübergreifende Architektur- und Technologiethemen
- ▶ Betreuung der Web Service Aktivitäten
- ▶ Beratung bezüglich Design und Implementierung
- ▶ Untersuchung der Web Service Plattformen diverser Hersteller
- ▶ Schwerpunkt: WS-Security und deren Interoperabilität



Agenda

- ▶ **Motivation**
 - ▶ *Szenario, Bedrohungen, Anforderungen, Lösungsbausteine*
- ▶ **WS-Security**
 - ▶ *Grundmechanismen, Signaturen, Authentifizierungsunterstützung*
 - ▶ *Design-Prinzipien, Architekturpatterns*
- ▶ **WS-SecurityPolicy**
 - ▶ *Prozessmodell, Policy-Provisioning*
 - ▶ *Choreografie-Patterns, Policy-Konfiguration, Policy-Management*





1. Service-Bereitstellung

- Komponenten:
 - Recipient: Empfangsmodule für die Nachrichten
 - Relying Party: Service-Anbieter (z.B. Bob Corporation)

2. Service-Konsument

- Komponenten:
 - Initiator: Modul zum Aufbereiten und Senden der Nachrichten
 - Requester: Service-Benutzer (z.B. Alice)

3. Nachrichtenaustausch

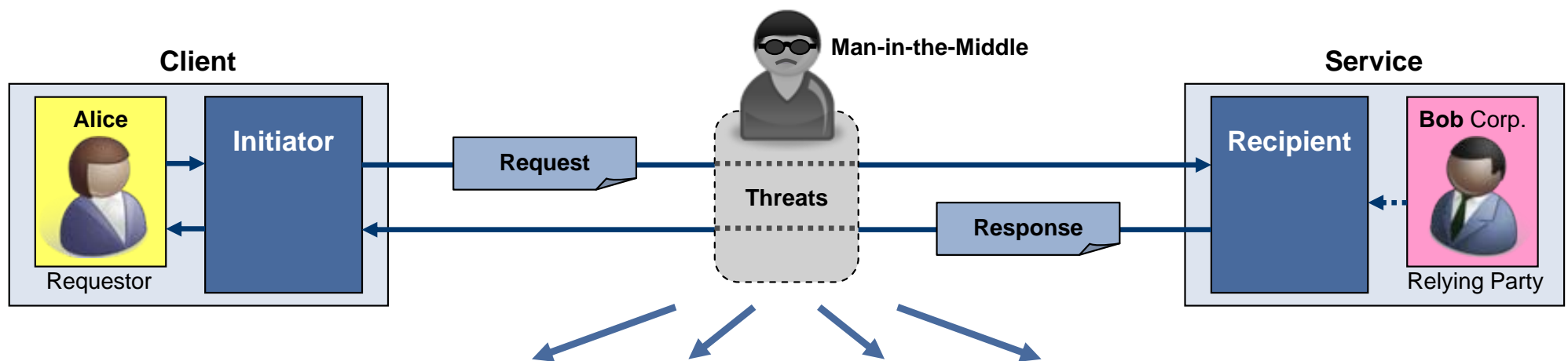
- Request- und Response-Nachricht (SOAP-Format)

4. Man-in-the-Middle

- Es gibt nicht die heile Welt...

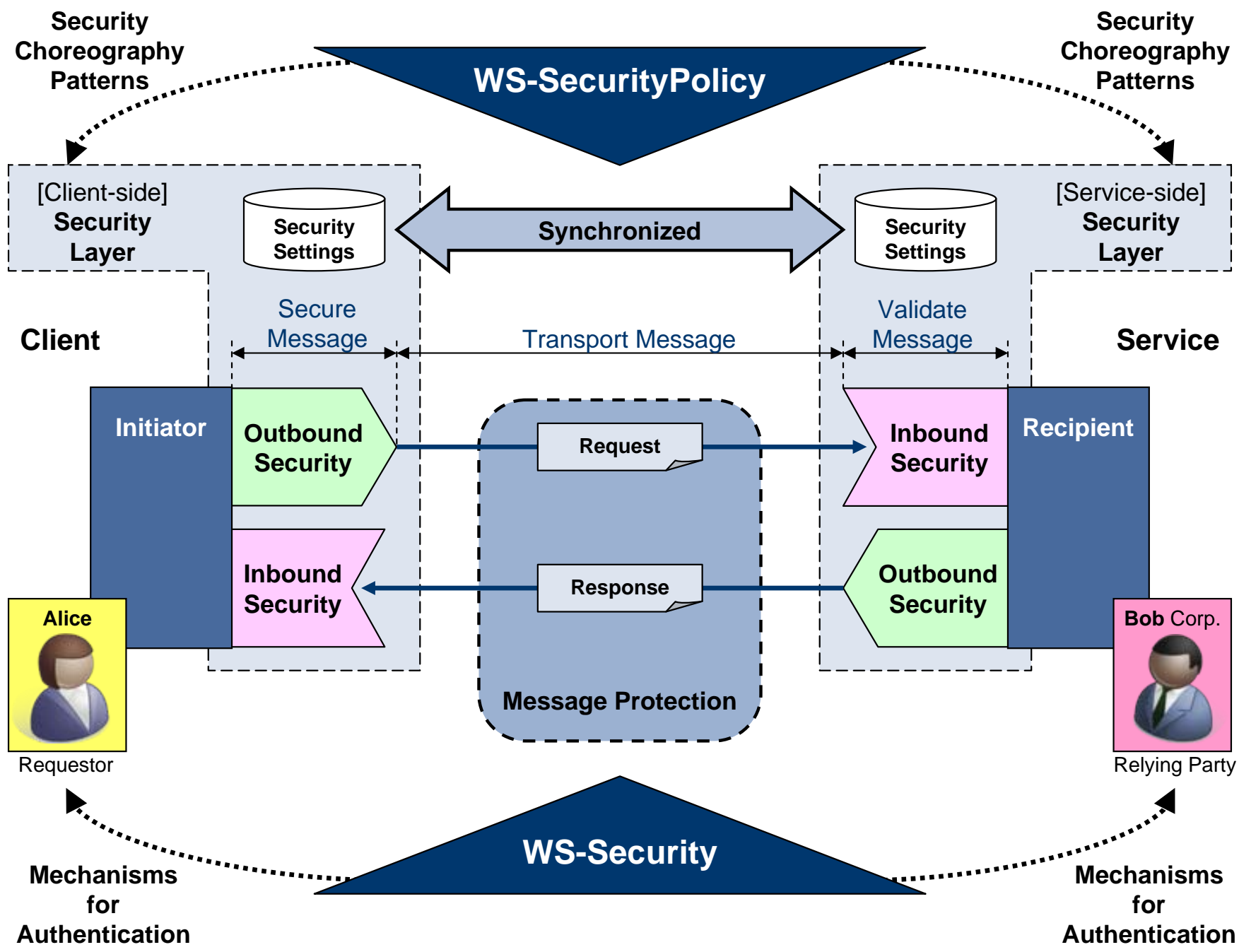


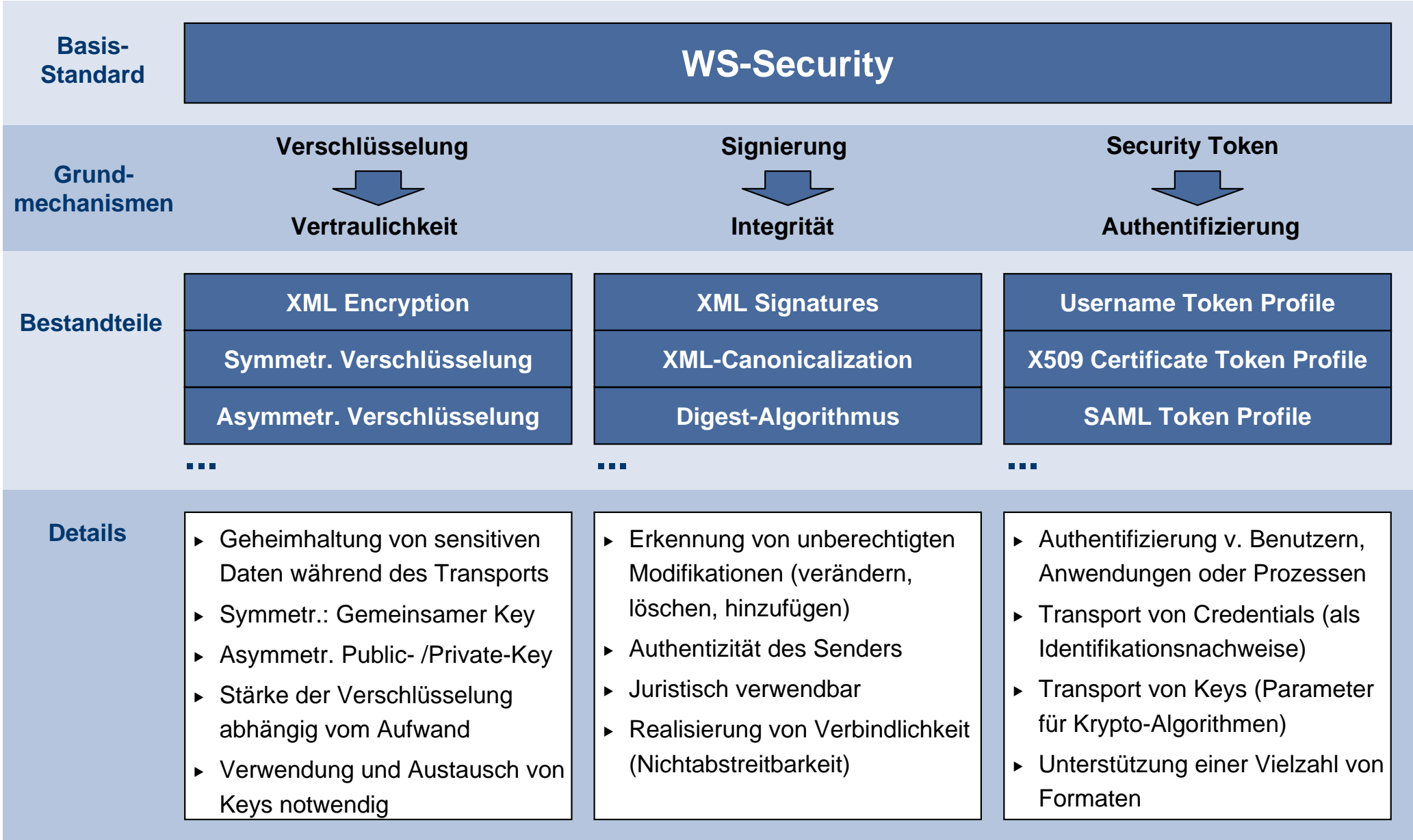
Bedrohungen, Anforderungen, Lösungsbausteine



Bedrohungen	Lauschangriff (Eavesdropping)	Manipulation (Falsification)	Falsche Identität (Masquerade)	Nachrichtenwiederholung (Replaying) ...
Anforderungen	Vertraulichkeit (Confidentiality)	Unverfälschtheit (Integrity)	Authentifizierung (Authentication)	Einmaligkeit (Uniqueness) ...
Lösungsbausteine	Verschlüsselung (Encryption)	Digit. Unterschrift (Signatures)	Identifikationsnachweise (Credentials, Token)	Nachrichten-ID (Timestamp, Nonce) ...

Big Picture: WS-Security & WS-SecurityPolicy

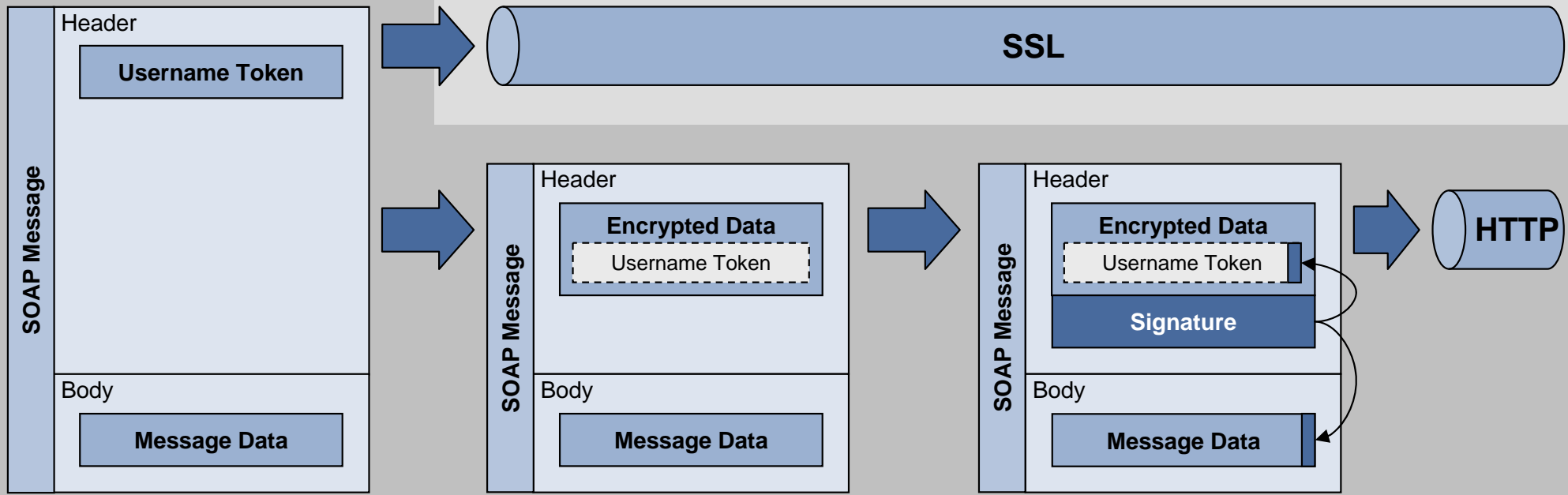




(Timestamp ist ein weiterer Mechanismus im Rahmen von WS-Security: Verhinderung von Replay-Attacken)

Message-Level Security

Transport-Level Security



Authentifizierungsdaten

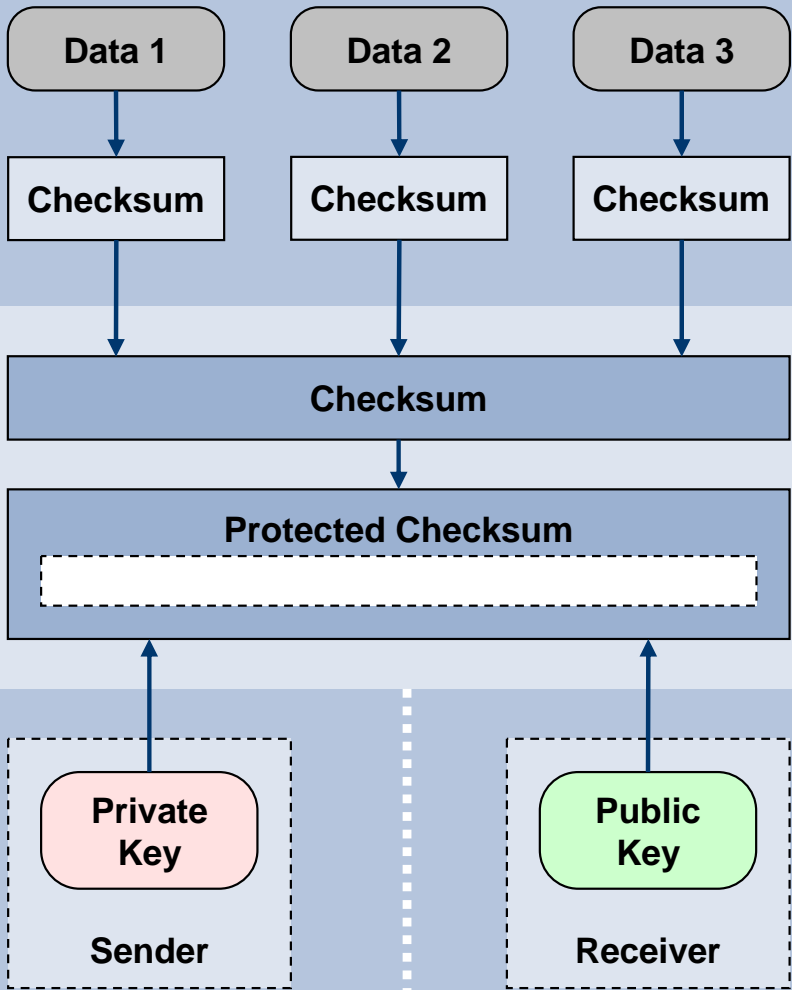
Vertraulichkeit

Integrität

- ▶ **Username-Token** als Container für Auth.-Daten
 - ▶ User-ID, Passwort
- Problem:**
 - ▶ Daten sind lesbar

- ▶ **Verschlüsselung** der Daten via WS-Security
 - ▶ Alternativ: SSL
- Problem:**
 - ▶ Username-Token ist nicht gekoppelt an die Nachricht

- ▶ **Signatur** u.a. als Klammer für mehrere Datenblöcke
- ▶ Nachweis für die Authentizität der Nachricht



Datenintegrität

- ▶ Erkennung von Datenmanipulationen (Verändern, Löschen, ...)
- ▶ Maßnahme: Bildung einer Prüfsumme (Hash, Digest)
- ▶ Algorithmus: SHA1

Nachrichtenthauthentizität

- ▶ Schutz vor Daten- und Prüfsummenmanipulationen
- ▶ Verknüpfung von mehreren Datenblöcken
- ▶ Maßnahme: Gemeinsame Prüfsumme und Verschlüsselung
- ▶ Algorithmen: SHA1, HMAC (symmetr. Verschlüsselung)

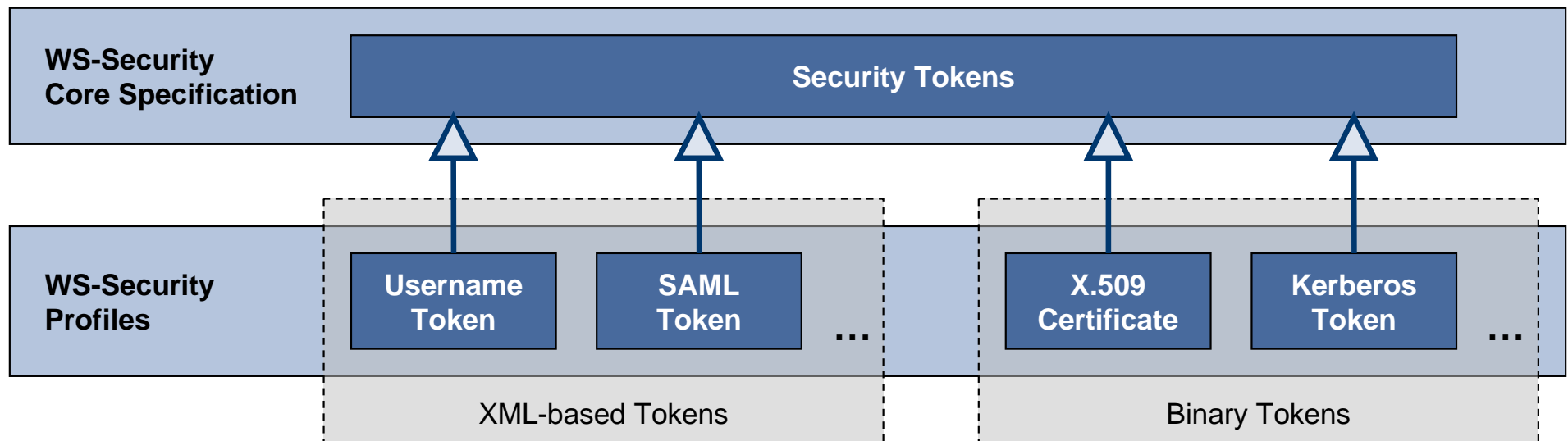
Sendersauthentizität

- ▶ Die Verwendung eines Private-Keys bietet die Möglichkeit eine Sendersauthentizität zu ermitteln
- ▶ Maßnahme: Asymmetrische (Public-Key) Verschlüsselung
- ▶ Algorithmen: DSA, RSA

Dilemma

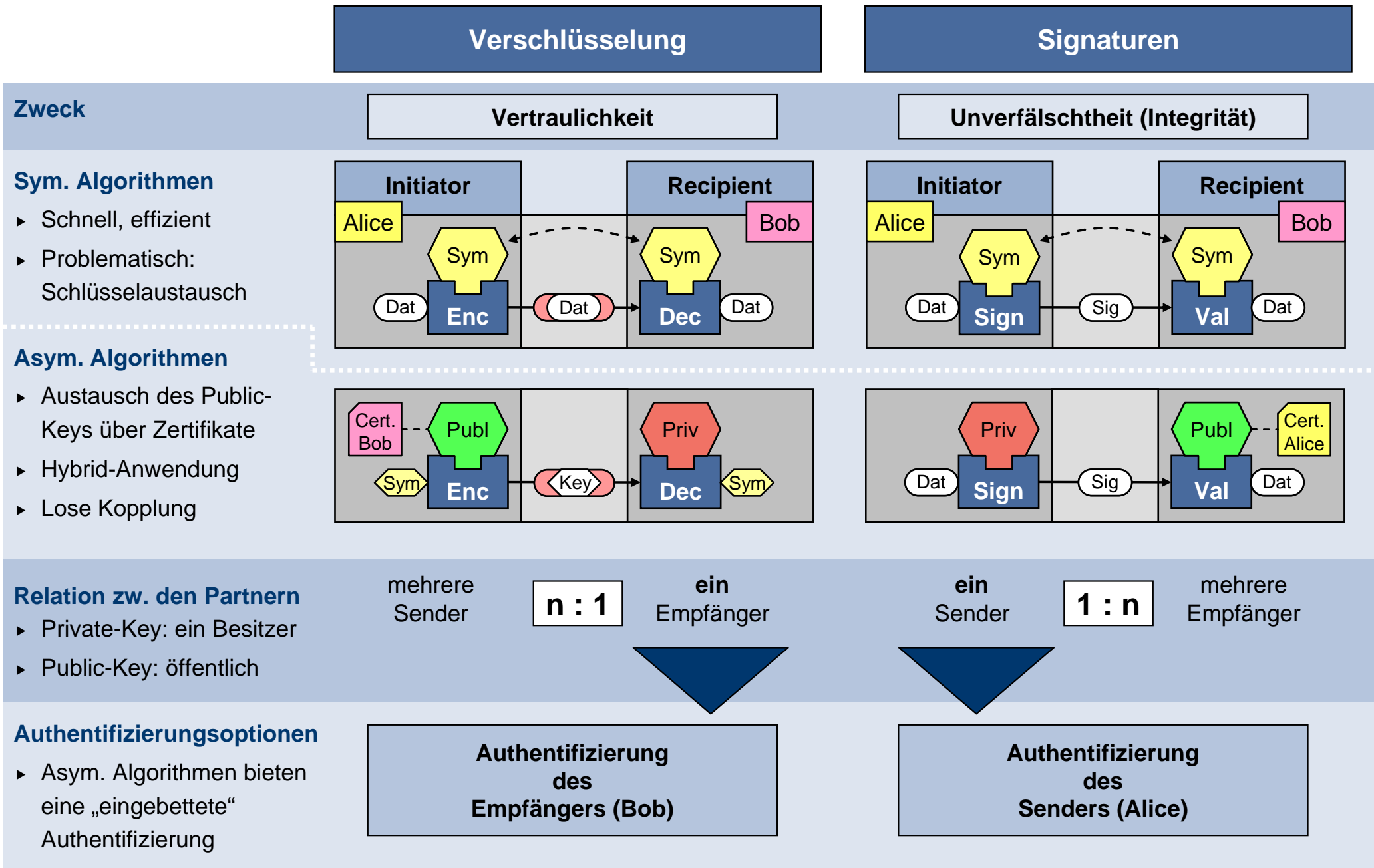
- ▶ Signaturen haben als Ergebnis nur: **Richtig oder Falsch** (Keine Details bei nicht Übereinstimmung)
- ▶ **Interoperabilität** ist die Herausforderung: Normalisierung, Digestbildung, Verschlüsselung, Abbildung

- ▶ Authentifizierung ist ein Kernbestandteil in einer Security-Infrastruktur
 - ▶ Voraussetzung für Autorisierung, Nachweisbarkeit (None Repudation) und Reporting
 - ▶ Bedingt den sicheren Transport der Informationen über die Identitäten (Credentials)
- ▶ WS-Security liefert dazu ein allgemeines Security Token Handling
 - ▶ Davon sind spezielle Token-Formate abgeleitet und in Form von zusätzlichen Profilen spezifiziert
 - ▶ Username-Token: Format für Benutzername/Passwort
 - ▶ SAML (Security Assertion Markup Language): Allg. Schema für den Austausch von Auth.-daten
 - ▶ X.509-Zertifikate: Handling für eine Public-Key-Umgebung

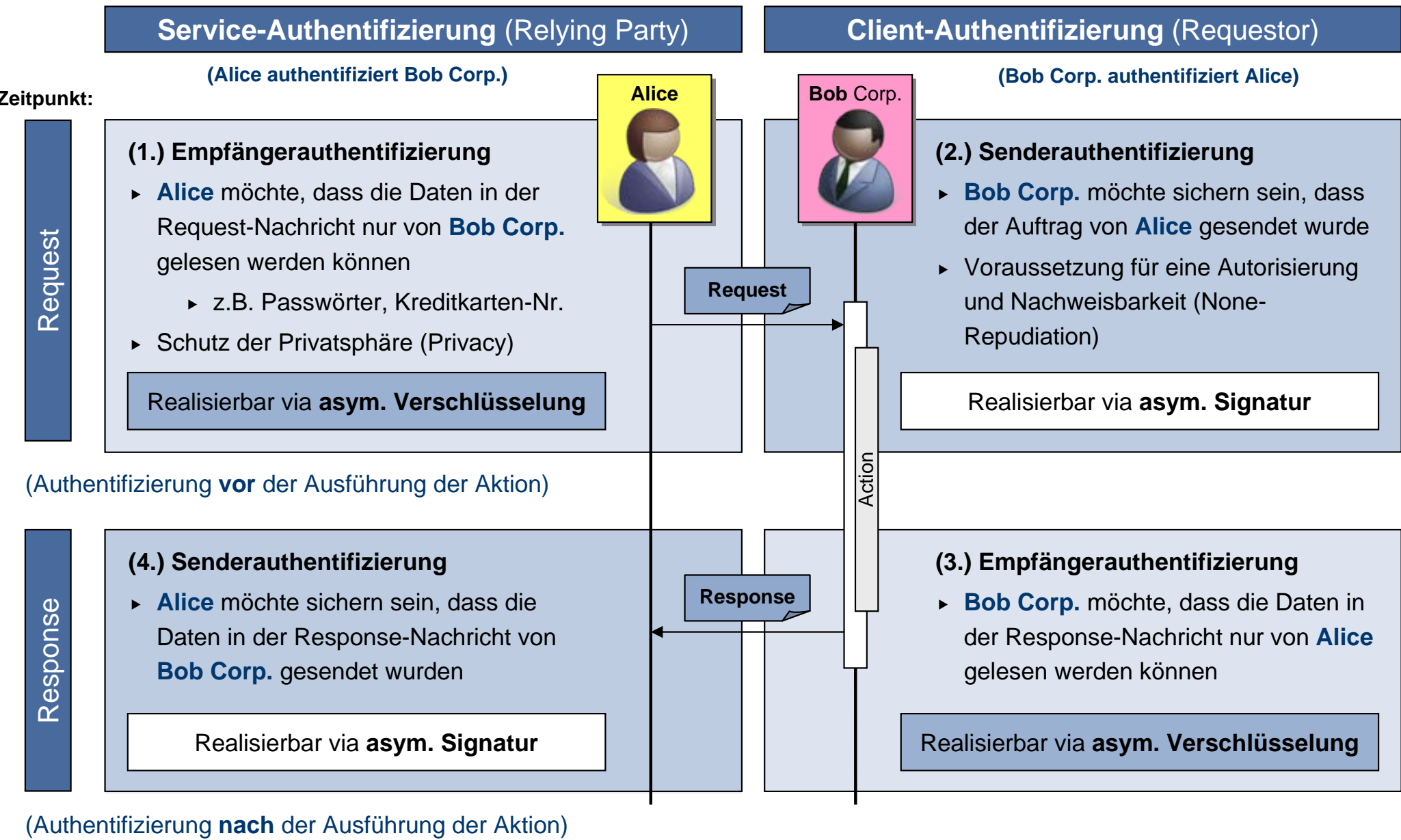


- ▶ Security-Tokens werden u.a. auch für die kryptografischen Algorithmen benötigt

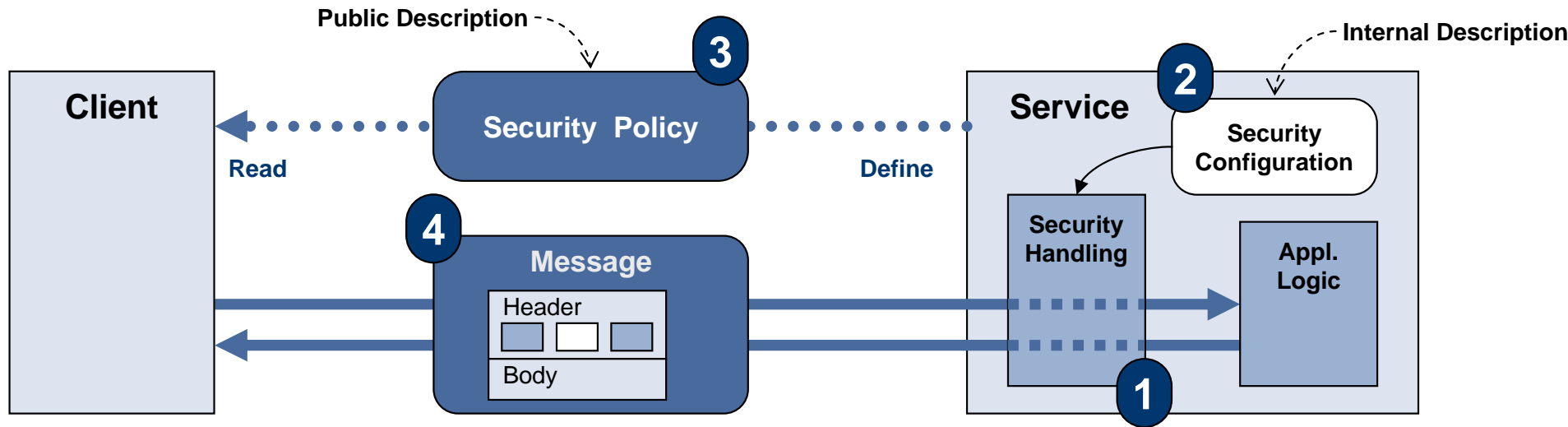
Authentifizierungsoptionen aufgrund der kryptografischen Algorithmen



Zeitliche Betrachtung der Authentifizierungsvarianten



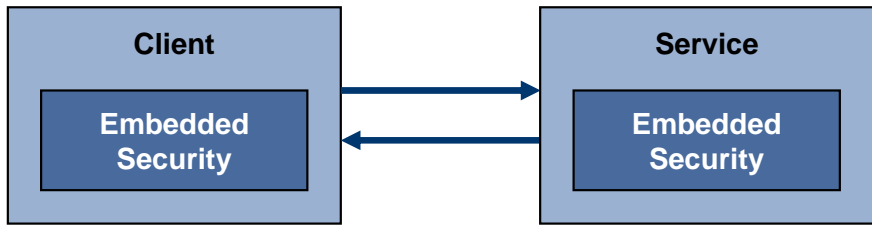
Grundsätzliche Design-Prinzipien für Security-Implementierungen



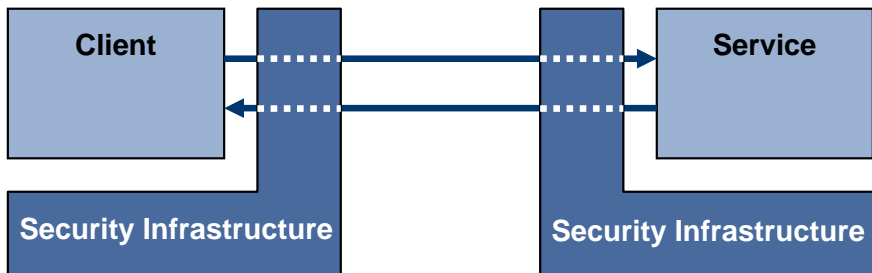
- | 1 | 2 | 3 | 4 |
|--|---|---|---|
| <h3>Self-contained-Security</h3> <ul style="list-style-type: none"> ▶ Trennung Security-Handling und Appl-Logik ▶ Unterschiedliche Lifecycles ▶ Pluggable Security ▶ Anbindung von verschiedenen Security-Frameworks | <h3>Security-by-Configuration</h3> <ul style="list-style-type: none"> ▶ Einsatzspezifische Anforderungen ▶ Anpassbare Security ▶ Keine hard-coded Security ▶ Interne Beschreibung ▶ Einstellungen für das interne Handling | <h3>Policy-driven-Security</h3> <ul style="list-style-type: none"> ▶ Service definiert die (MUSS) Anforderungen ▶ Bereitstellung in einem Standardformat (WS-SecurityPolicy) ▶ Client steuert danach sein Security-Handling | <h3>Self-describing-Security</h3> <ul style="list-style-type: none"> ▶ Beschreibung der verwendeten Security-Mechanismen ▶ Keine Abstimmungen aller Details zwischen Client und Service (KANN-Eigenschaften) ▶ Grundlage: WS-Security |

(Prinzipien gelten auch für die Client-seitige Realisierung der Security)

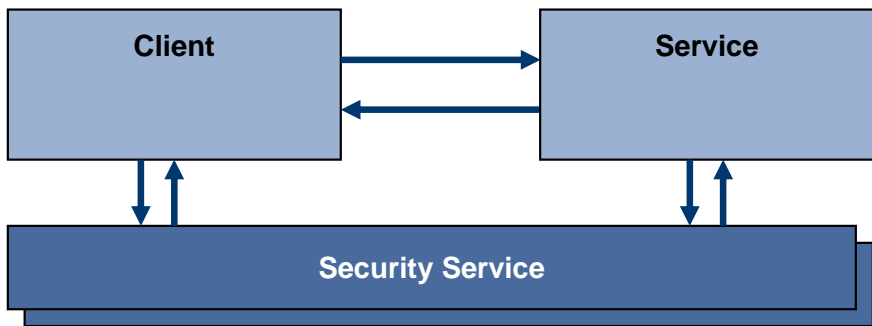
Architekturpatterns für die Realisierung von Web Service Security



- Embedded Security**
- ▶ Unabhängig, aber komplex, proprietär, monolithisch
 - ▶ Keine zentrale Administrationsmöglichkeit
 - ▶ Keine Trennung von Security-Handling und Applikationslogik

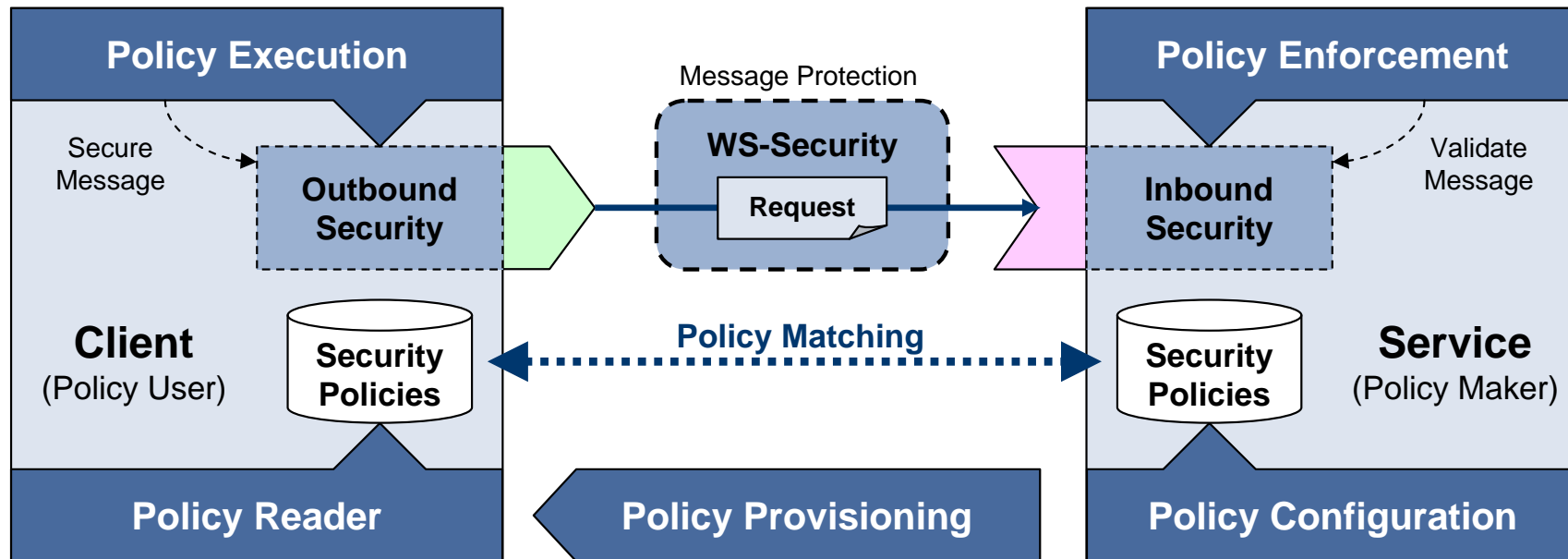


- Security as Infrastructure**
- ▶ Entkopplung der Applikationslogik vom Security-Handling
 - ▶ Security-Handling ist für Konsument und Anbieter transparent
 - ▶ Realisierung als Software oder auch als Hardware



- Security as a Service**
- ▶ Bereitstellung der Security-Funktionalität als Dienst
 - ▶ Wiederverwendung, zentrales Management
 - ▶ Konsument und Anbieter benutzen aktiv den Security-Service
 - ▶ Mehrere Services für unterschiedliche Aufgaben

Oder Mischformen: Client und Service verwenden unterschiedliche Pattern.



1. Policy Configuration:

- ▶ Initiale Konfiguration der Policies
- ▶ Service hat die Rolle für die Vorgabe der Policies

2. Policy Provisioning:

- ▶ Adressiert die Verteilung der Policies zu den einzelnen Clients

3. Policy Reader:

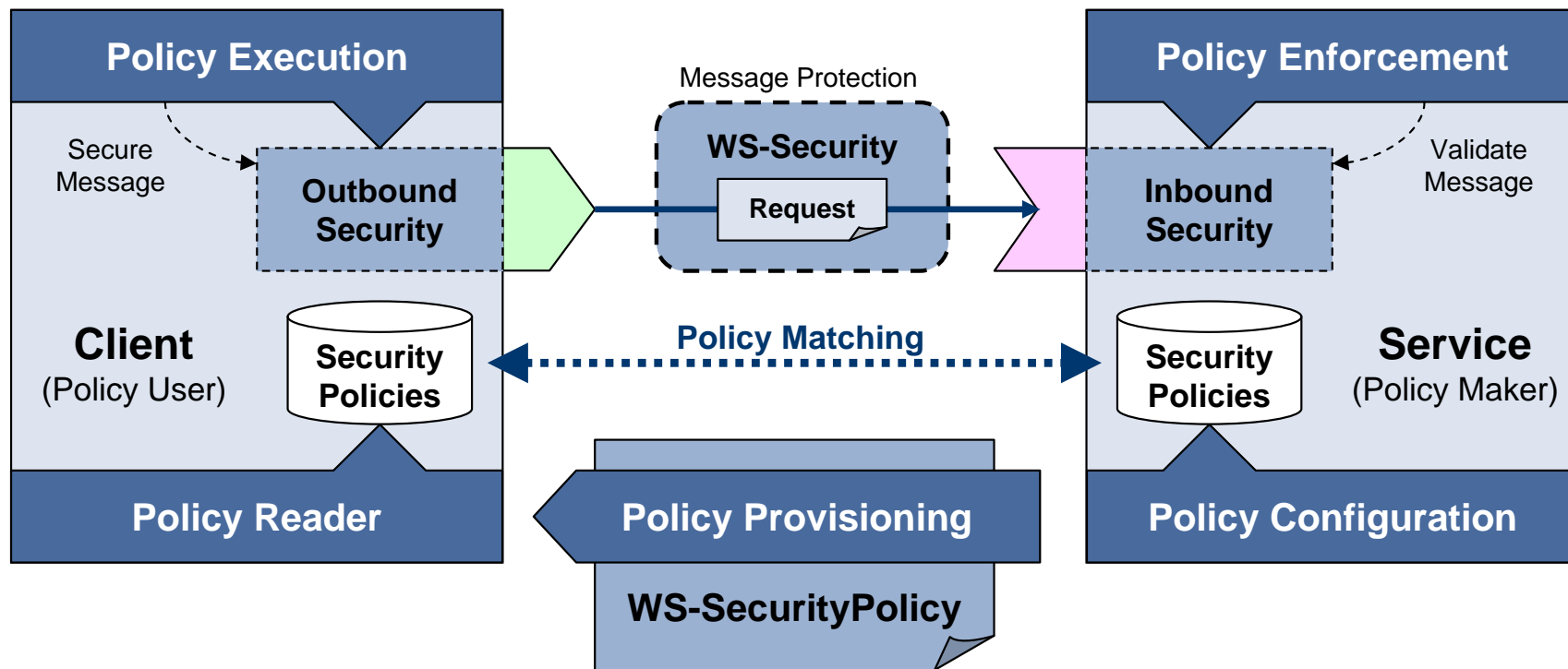
- ▶ Client muss die Policies lesen, interpretieren und (möglichst) sich selbst konfigurieren

4. Policy Execution:

- ▶ Policies werden vom Client entsprechend umgesetzt

5. Policy Enforcement:

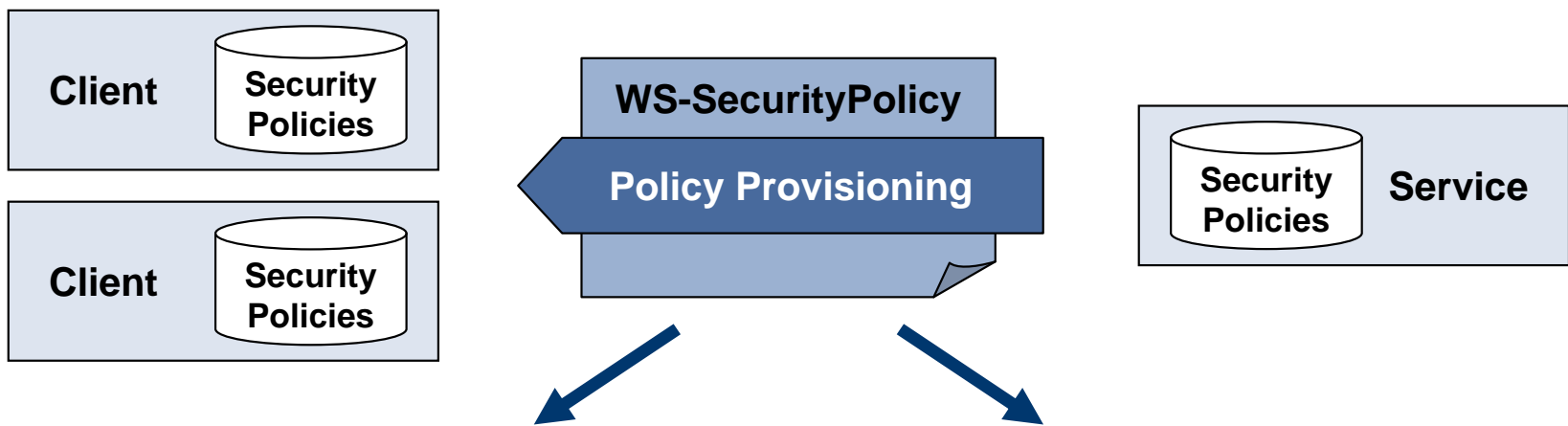
- ▶ Service kontrolliert die konforme Umsetzung der Policies



WS-SecurityPolicy:

- ▶ Bestandteil einer umfangreichen Familie aus Policy-Spezifikationen
 - ▶ WS-Policy als Basisstandard bietet eine allgemeine erweiterbare Sprache für die Beschreibung von Anforderungen und Einschränkungen in Form von Assertions
 - ▶ WS-PolicyAttachment spezifiziert die Einbettung von Policies in anderen Standards (z.B. WSDL)
- ▶ WS-SecurityPolicy liefert konkrete Beschreibungen für Security-Eigenschaften von SOAP-Nachrichten
 - ▶ Basiert auf den Mechanismen von WS-Security, WS-Trust und WS-SecureConversation
 - ▶ Beschreibung erfolgt mittels eines Basissatzes von Assertions

Ansätze für das Client-Policy-Provisioning



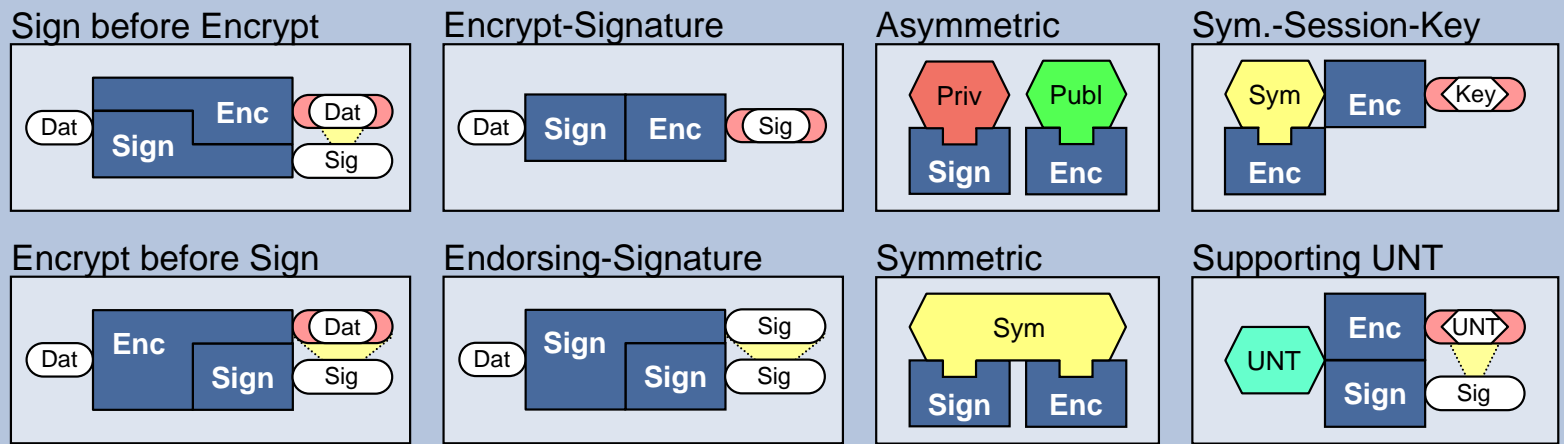
	WSDL	WS-MEX
Ansatz:	<ul style="list-style-type: none"> ▶ Web Service Description Language (W3C) ▶ Standard für die Beschreibung von Web Service, u.a. erweitert mit Policy-Assertions ▶ Beschreibungsdokument 	<ul style="list-style-type: none"> ▶ WS-MetadataExchange (W3C) ▶ Bootstrap-Mechanismus für den Austausch von Metadaten, u.a. von Policy-Daten ▶ Protokoll für die Anforderung von Daten
Transportierte Daten:	<ul style="list-style-type: none"> ▶ Funktionale Beschreibung (Operationen) ▶ Deployment Daten (Ports und Protokoll) ▶ Policy-Assertions 	<ul style="list-style-type: none"> ▶ Client oder Service können steuern, welche Daten ausgetauscht werden <ul style="list-style-type: none"> ▶ z.B. nur die Policy-Daten
Zeitpunkt des Aufrufs:	<ul style="list-style-type: none"> ▶ Typischerweise zur Entwicklungs- und Deploymentphase ▶ Kein Mechanismus, wenn sich Policy-Daten ändern (automatisches Nachladen) 	<ul style="list-style-type: none"> ▶ Zeitpunkt ist beliebig (Abstimmung zwischen Client und Service) ▶ WS-Federation: bei Policy-Mismatch liefert der Service einen SOAP-Fault mit WS-MEX-Referenz auf die gültige Policy

WS-Security

Grundmechanismen für Security

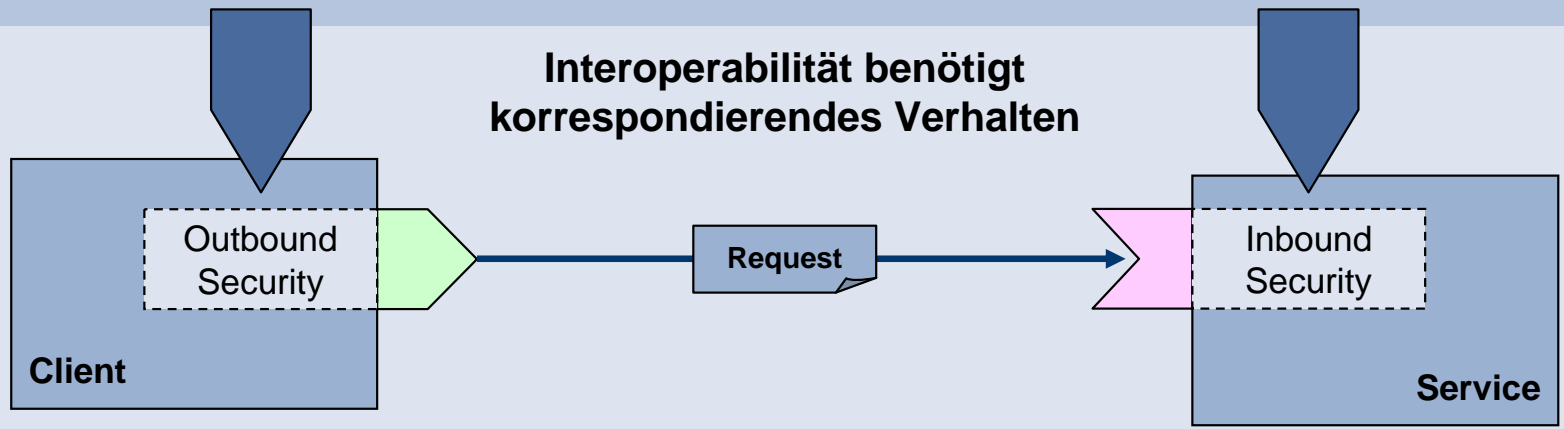


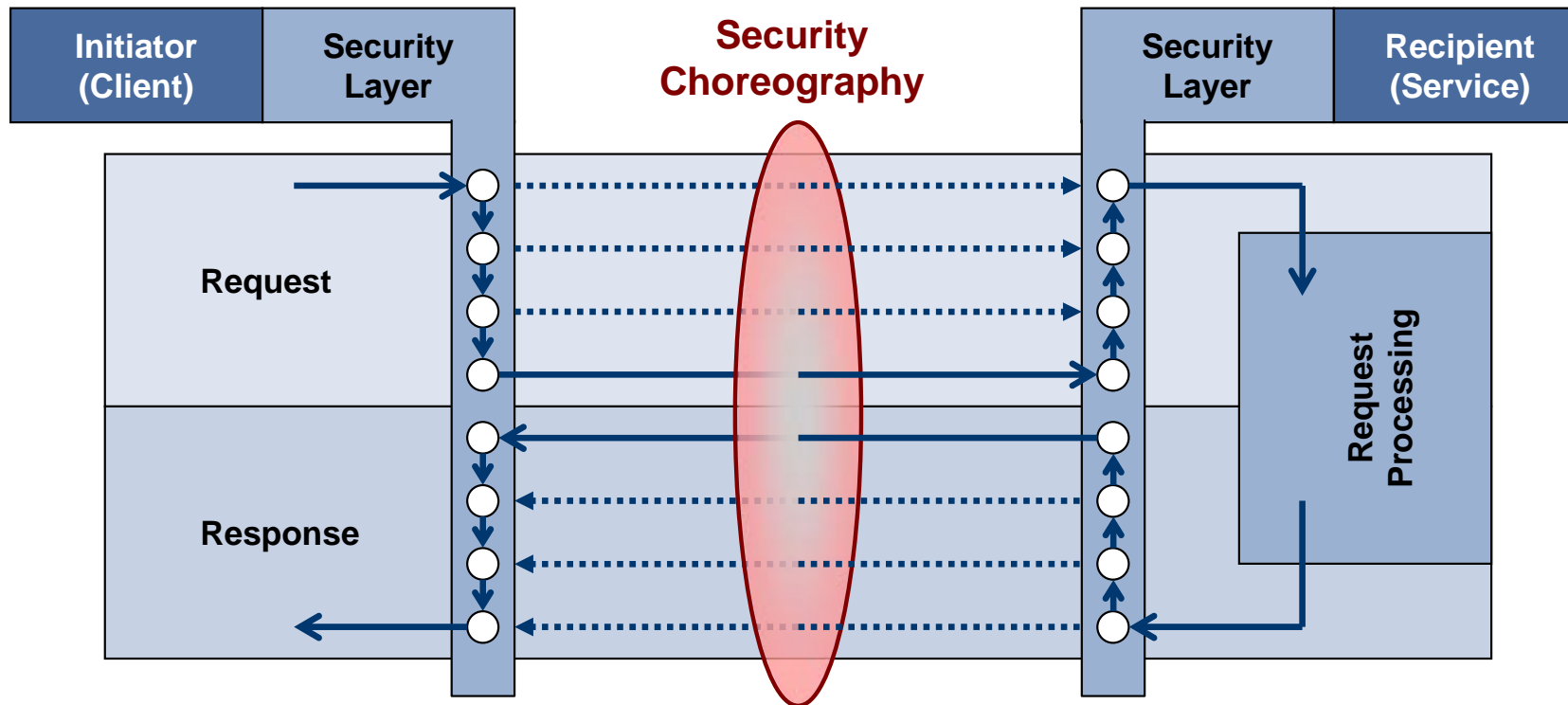
Security = Kombination der einzelnen Mechanismen



WS-SecurityPolicy

Interoperabilität benötigt korrespondierendes Verhalten





Security-Choreografie:

- ▶ Abgestimmtes Verhalten zwischen den Kommunikationspartnern Client und Service bezüglich:
 - ▶ Arten der Security-Mechanismen (Element)
 - ▶ Reihenfolge der Abarbeitung (Chronologie)
- ▶ Nicht verwechseln mit WS-Choreography (Beschreibungssprache für den Nachrichtenfluss)
- ▶ *Choreografie: griech. choreos = Tanz und graphos = Schrift → Beschreibung von Bewegungen*

Pattern #1

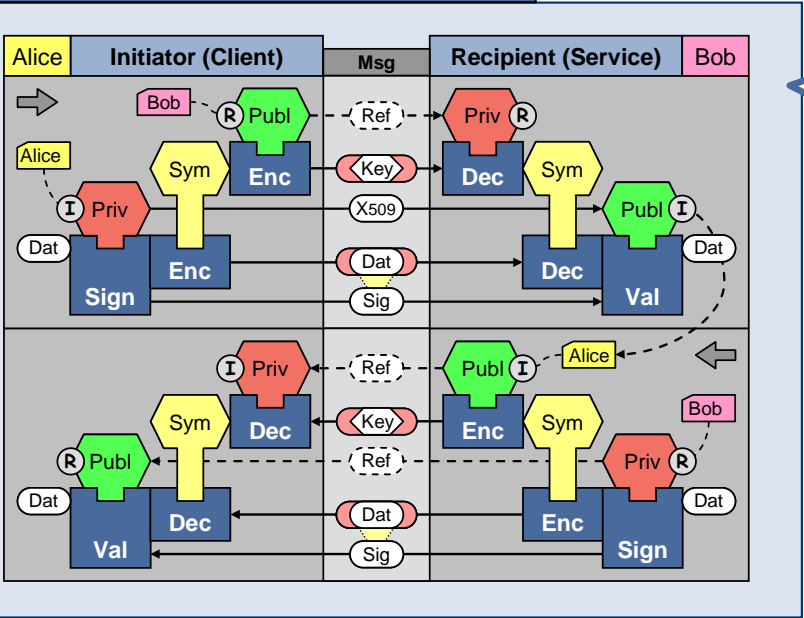
Pattern #3

Pattern #2

Pattern #4

Choreografie-Patterns von WS-SecurityPolicy (#1)

Asymmetric Binding



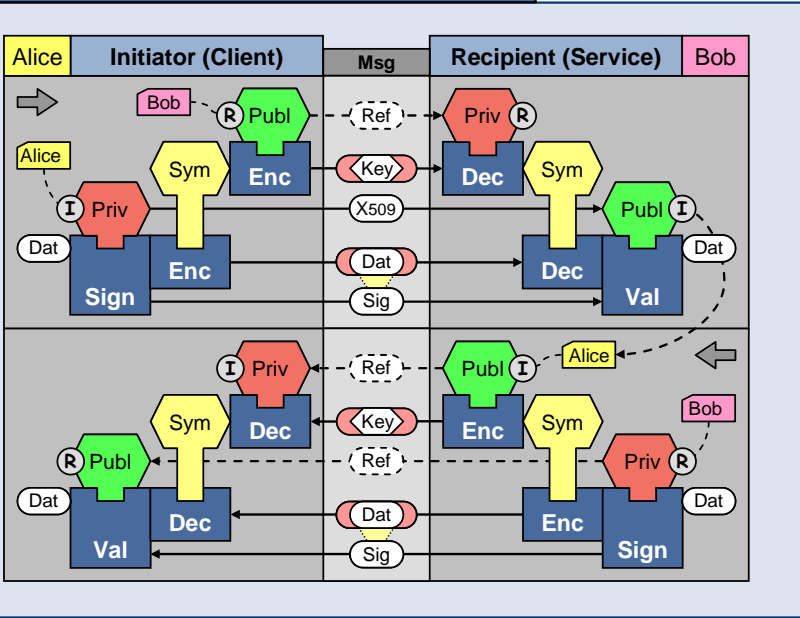
- ▶ Asymmetrische Algorithmen für das Verschlüsseln und Signieren von Daten
 - ▶ Verschlüsselung: Hybrid-Verfahren
- ▶ Gegenseitige Authentifizierung via Zertifikate
- ▶ Gleichartiges Handling für die Response-Nachricht

Pattern #2

Pattern #4

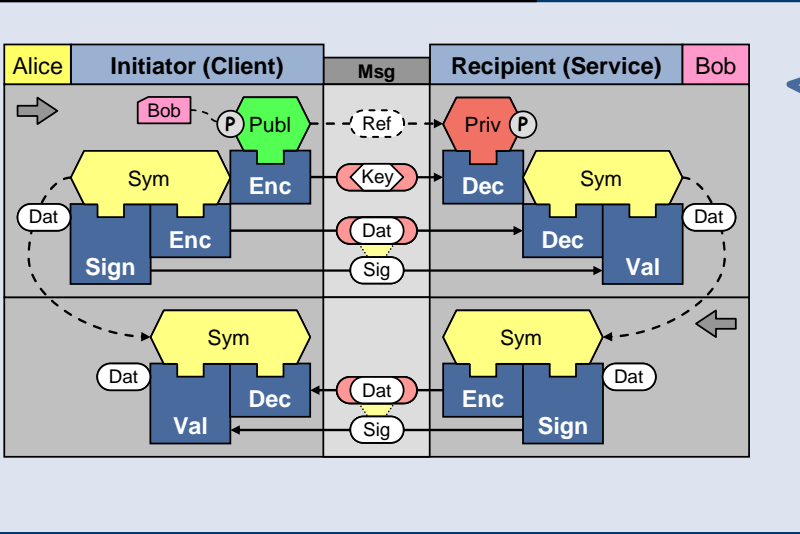
Choreografie-Patterns von WS-SecurityPolicy (#2)

Asymmetric Binding



Pattern #3

Symmetric Binding

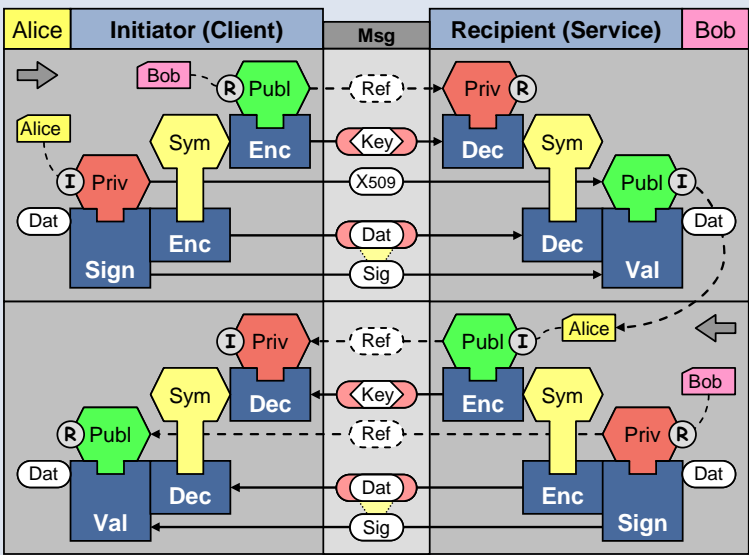


- ▶ Gemeinsamer symmetrischer Schlüssel für Verschlüsselung und Signaturen
- ▶ Keine Client-Authentifizierung (anonymer Client)
- ▶ Einfaches Handling für die Response-Nachricht

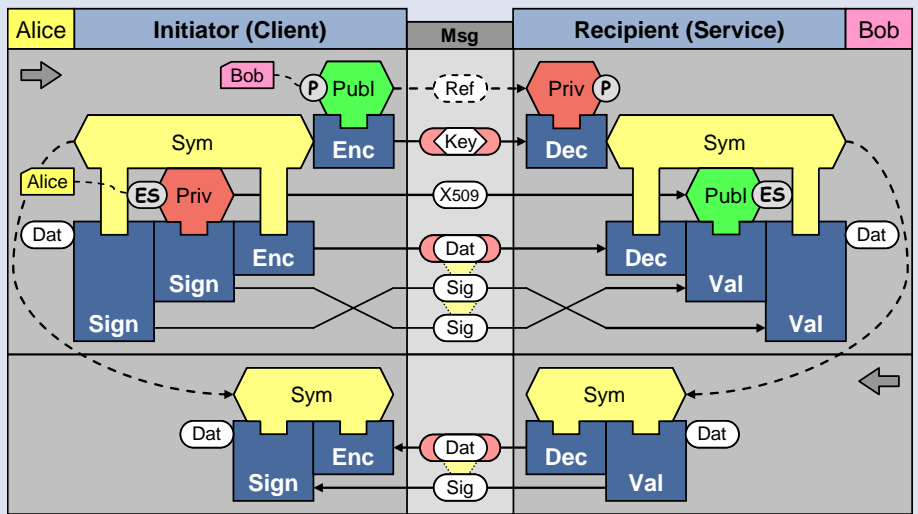
Pattern #4

Choreografie-Patterns von WS-SecurityPolicy (#3)

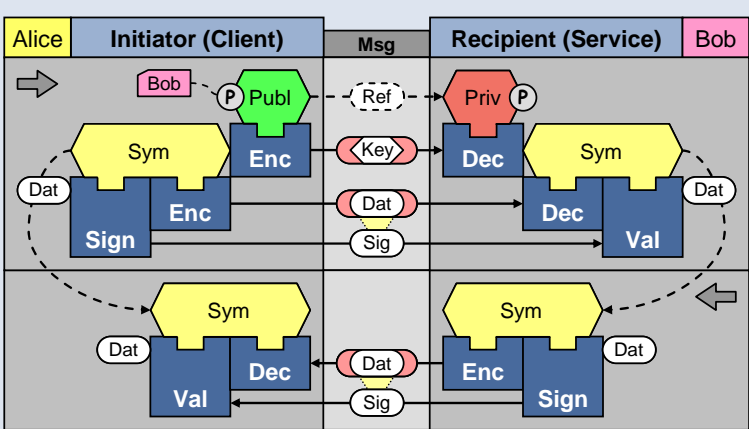
Asymmetric Binding



Symmetric Binding & Endorsing Signature



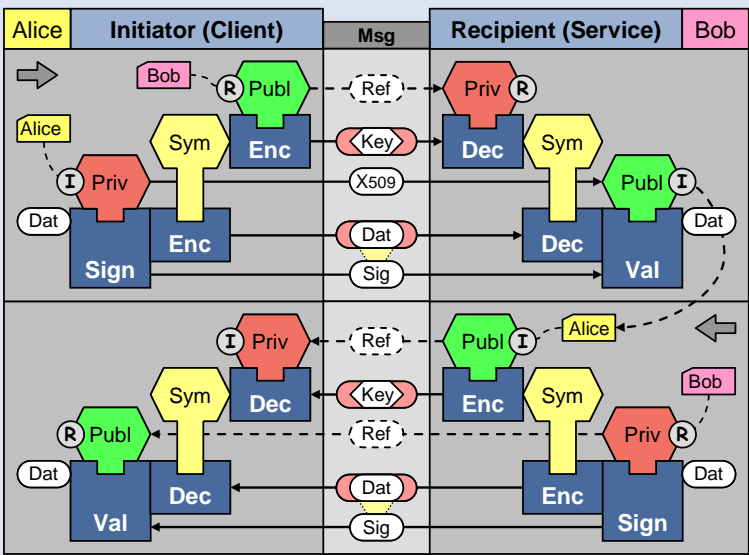
Symmetric Binding



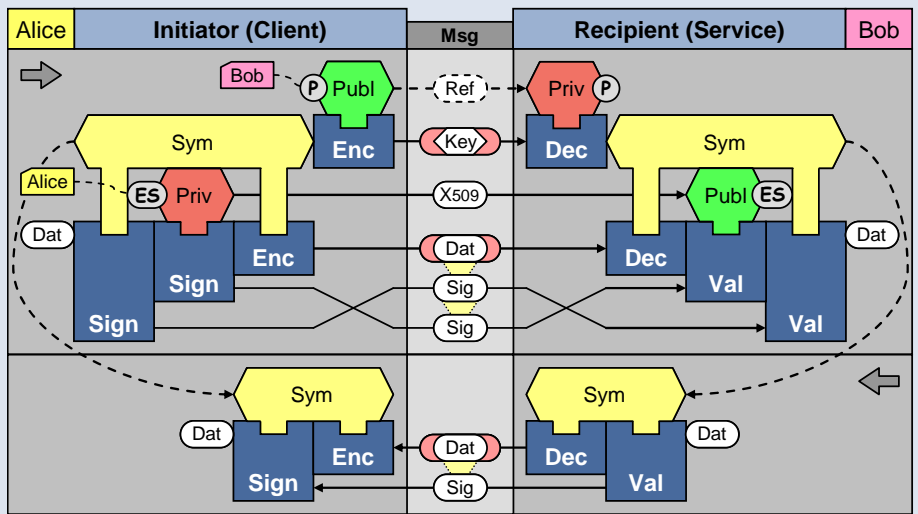
- ▶ Gleicher symmetrischer Ansatz
- ▶ Zusätzliche asymmetrische Signatur für die Client-Authentifizierung
- ▶ Gegenseitige Authentifizierung via Zertifikate
 - ▶ Einfaches Handling für Response-Nachricht bleibt erhalten

Choreografie-Patterns von WS-SecurityPolicy (#4)

Asymmetric Binding

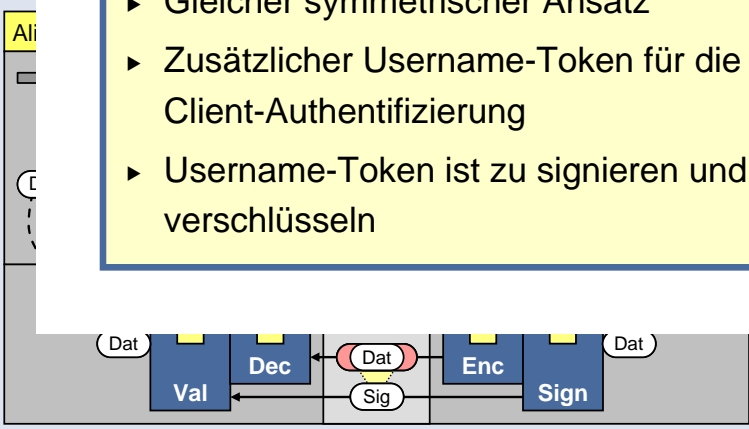


Symmetric Binding & Endorsing Signature

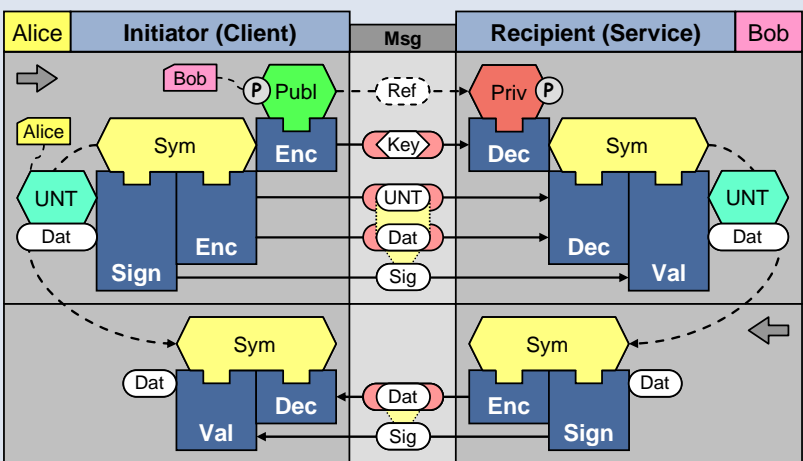


Sym

- ▶ Gleicher symmetrischer Ansatz
- ▶ Zusätzlicher Username-Token für die Client-Authentifizierung
- ▶ Username-Token ist zu signieren und zu verschlüsseln

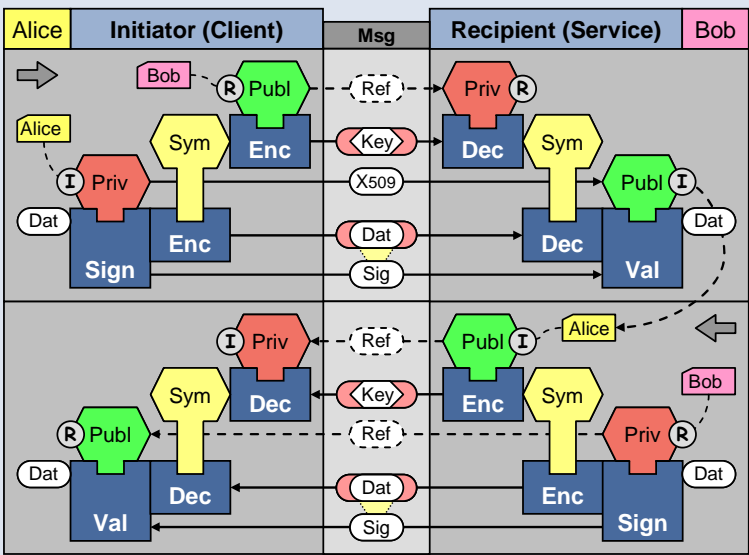


Symmetric Binding & Username Token

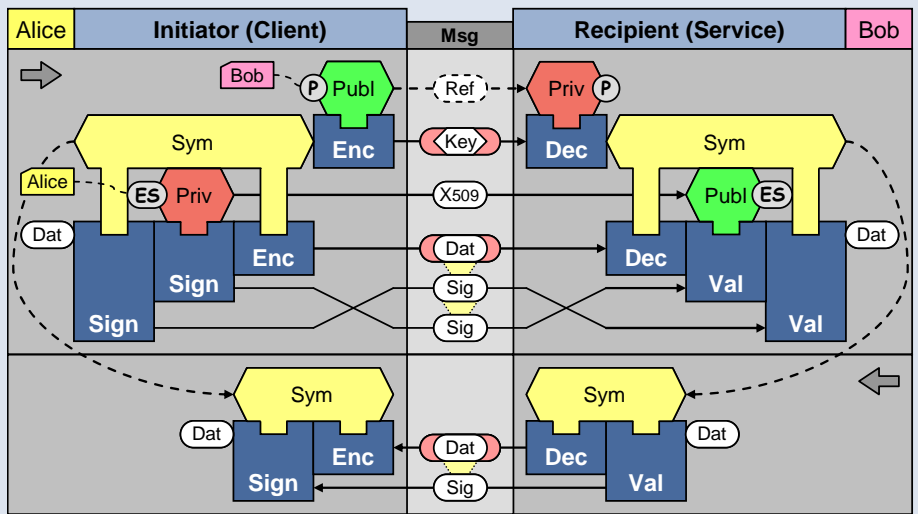


Choreografie-Patterns von WS-SecurityPolicy

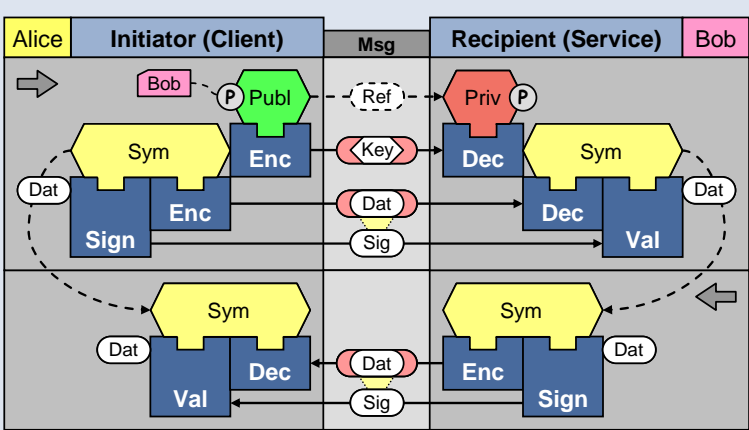
Asymmetric Binding



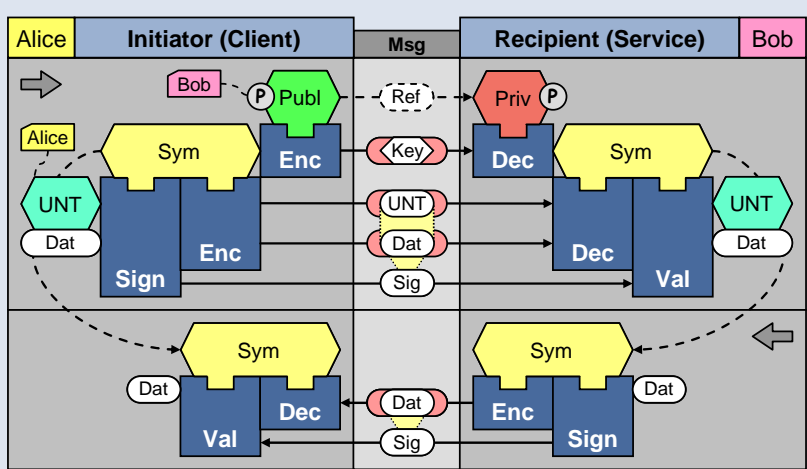
Symmetric Binding & Endorsing Signature



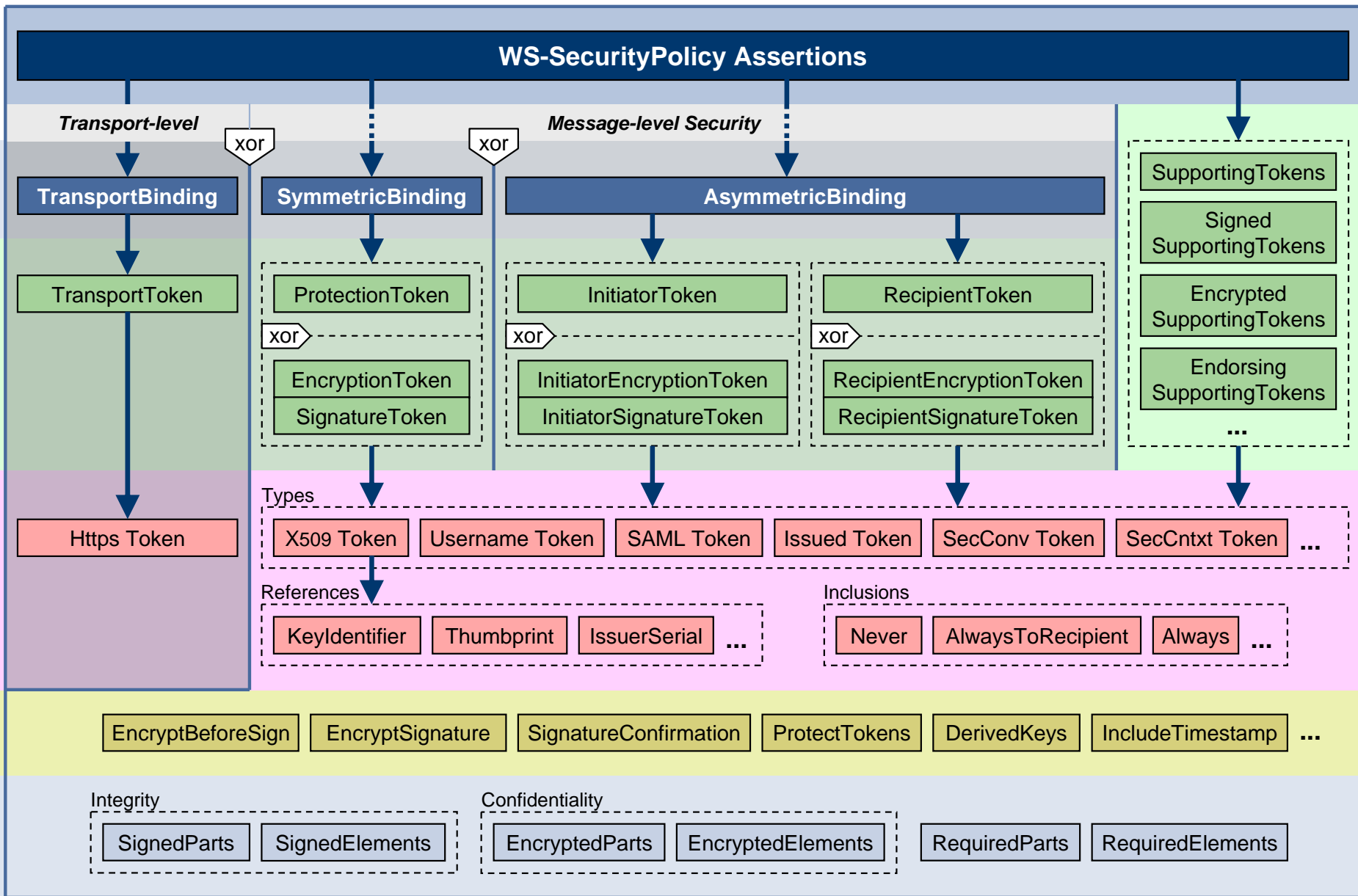
Symmetric Binding



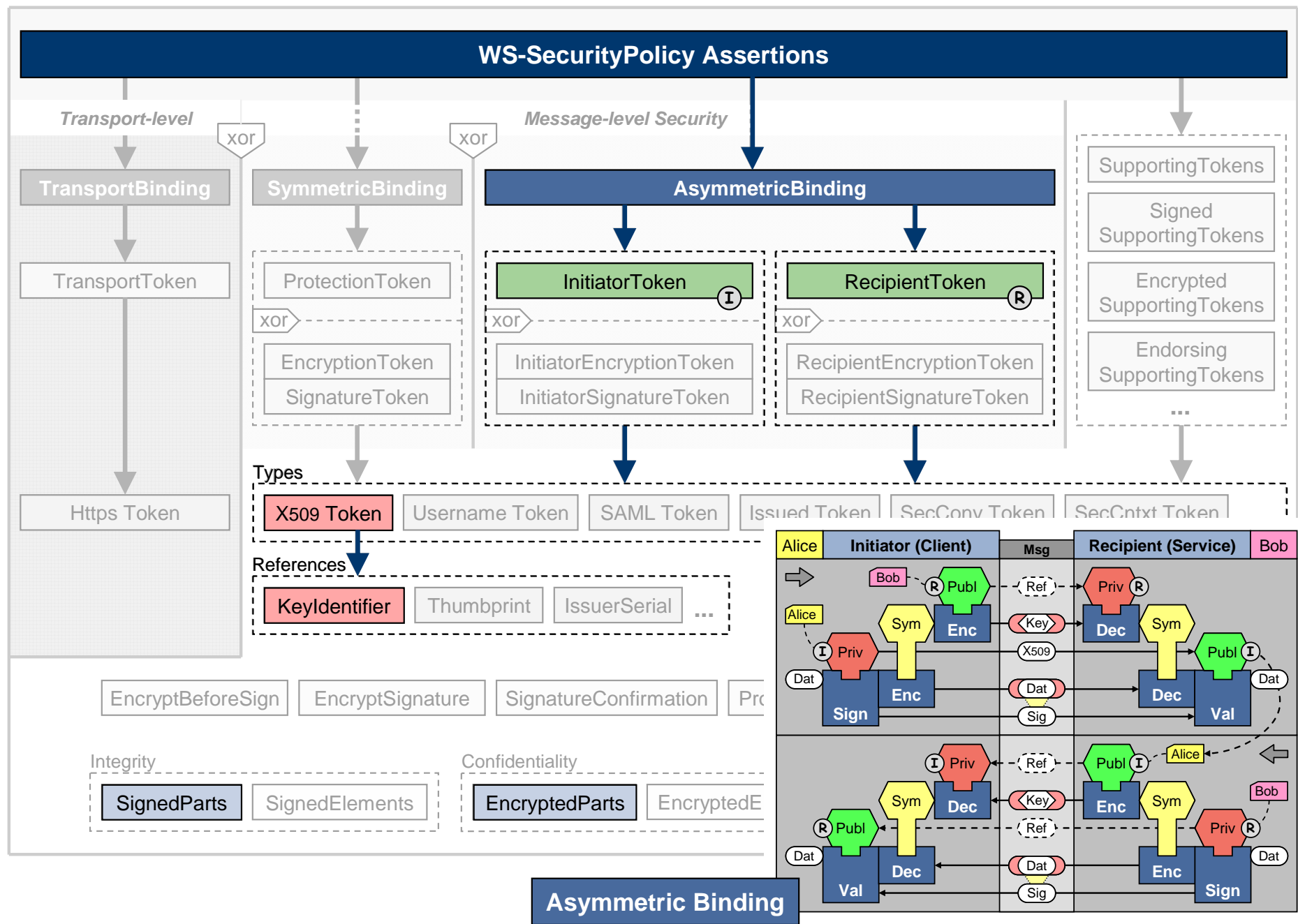
Symmetric Binding & Username Token



Assertion-Struktur von WS-SecurityPolicy

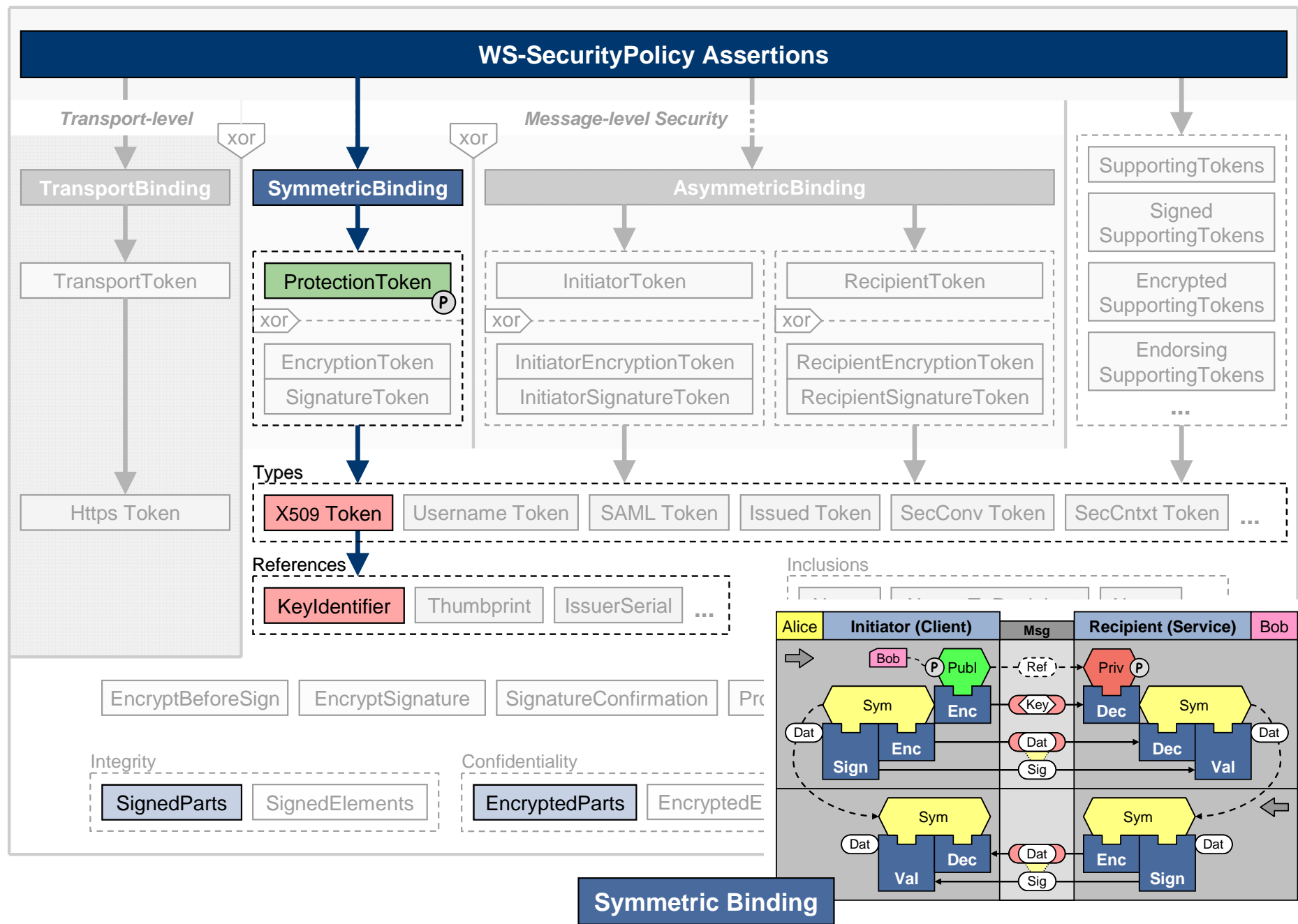


Assertions für Choreografie „Asymmetric Binding“

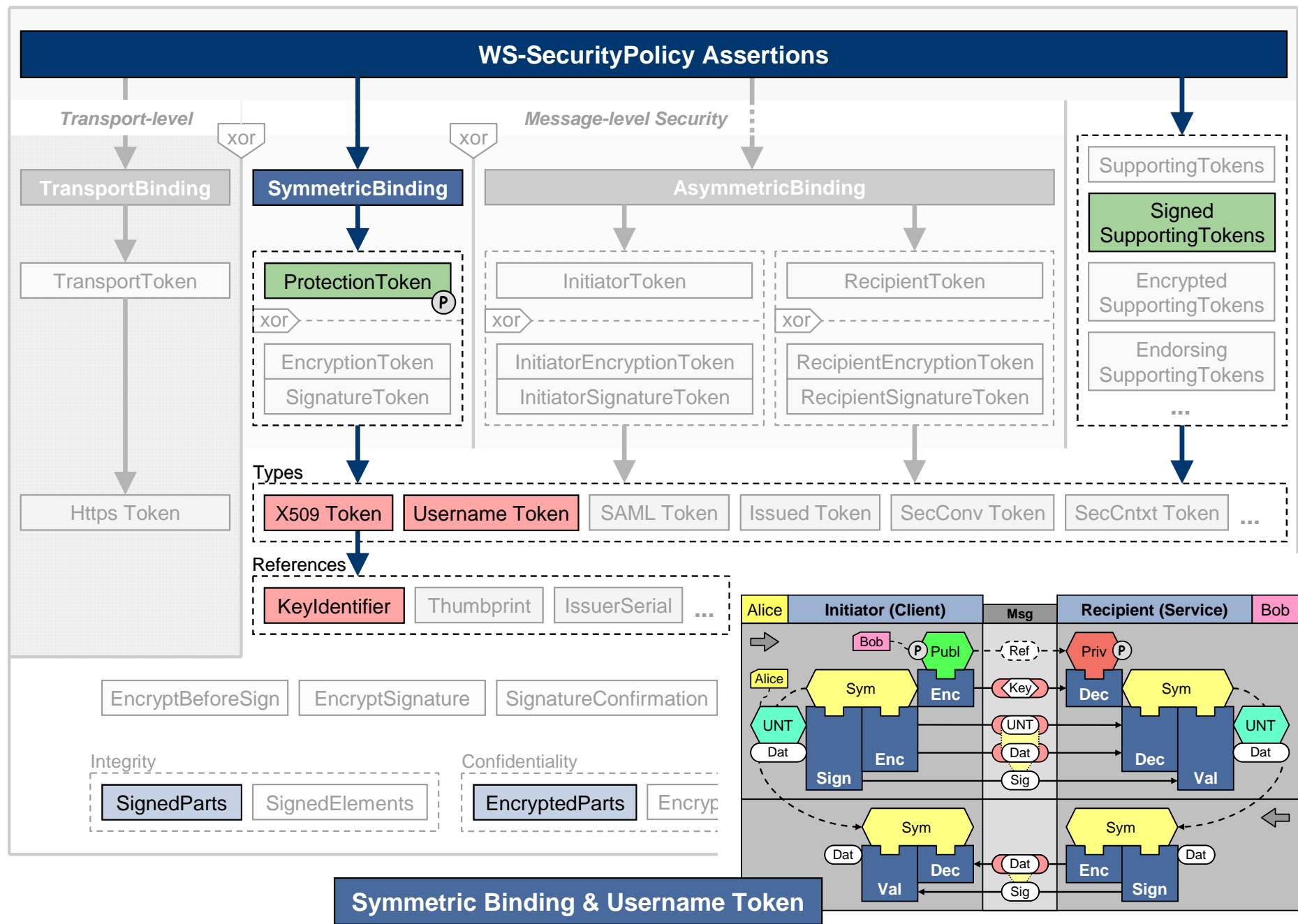


Asymmetric Binding

Assertions für Choreografie „Symmetric Binding“

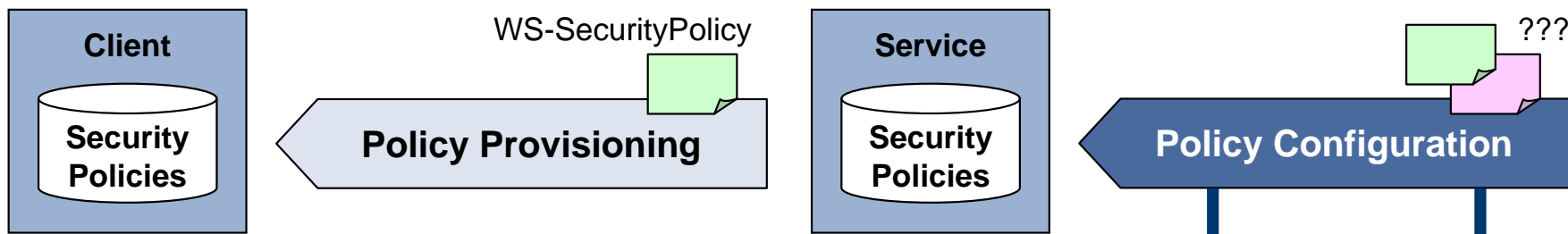


Assertions für Choreografie „Symmetric Binding & Username Token“



Symmetric Binding & Username Token

Konfigurationsansätze für die serverseitigen Policies



Ansätze:

Step-by-Step Approach

- ▶ Vordefinierte Einzelaktivitäten (=Grundmechanismen)
- ▶ Erstellung einer eigenen Security-Choreografie durch Aneinanderreihung von Aktivitäten
- ▶ Hohes Verständnis über Sec.-Prinzipien notwendig

Profile-oriented Approach

- ▶ Vordefinierte Profile (Templates)
- ▶ Vorgabe einer Basis-Choreografie mit zusätzlichen Optionen
- ▶ Erfordert („nur“) Grundwissen über Security

Formate:

Proprietär

- ▶ Plattformspezifisches (XML-) Format
- ▶ Erfordert eine Transformation in das Format von WS-SecurityPolicy für das Policy-Provisioning

- ▶ Nachteilig: Unterschiedliche Abstraktion und Bezeichnungen, Transformationsprobleme

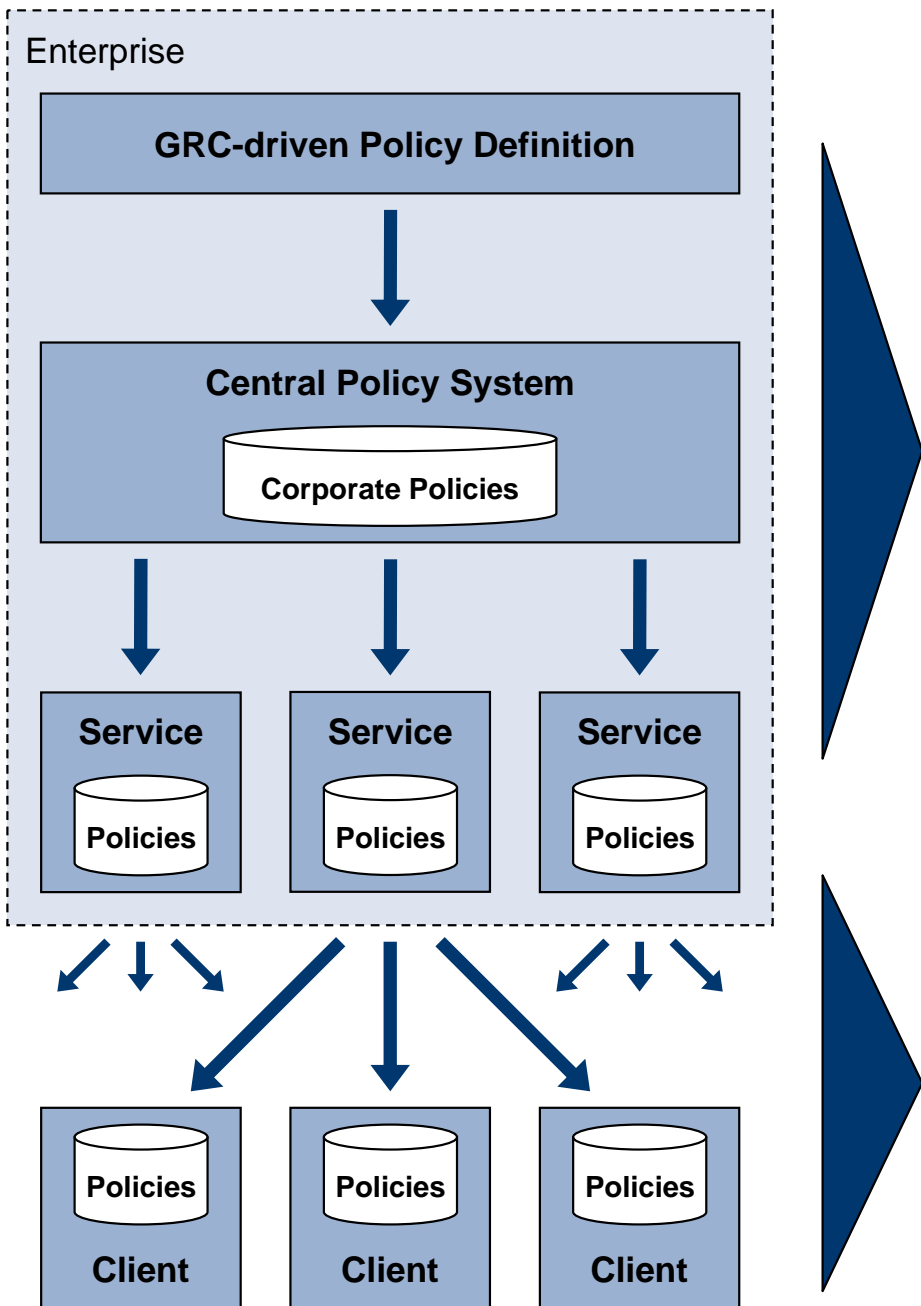
WS-SecurityPolicy

- ▶ Basiert auf dem Schema von WS-SecurityPolicy
- ▶ Entspricht der Security-Beschreibung für das Policy-Provisioning

- ▶ Prinzip: Es ist nur das konfigurierbar, was mittels WS-SecurityPolicy definierbar ist!

Interoperabilität zwischen WSIT und WCF bzgl. der Security-Choreografie

Choreografie	<div style="display: flex; align-items: center; gap: 10px;"> Alice Initiator (Client) Msg Recipient (Service) Bob </div>	WSIT [Profiles]	WCF [Authentication Mode]
Asymmetric Binding		Mutual Certificate	Mutual Certificate (SecurityVersion = WSSecurity10)
Symmetric Binding		-- kein eigenes Profile -- (Einstellbar über WS-SecurityPolicy-Assertions)	Anonymous for Certificate
Symmetric Binding & Endorsing Supporting Token		Endorsing Certificate	Mutual Certificate (SecurityVersion = WSSecurity11)
Symmetric Binding & Username Token		Username Authentication with Symmetric Key	Username for Certificate



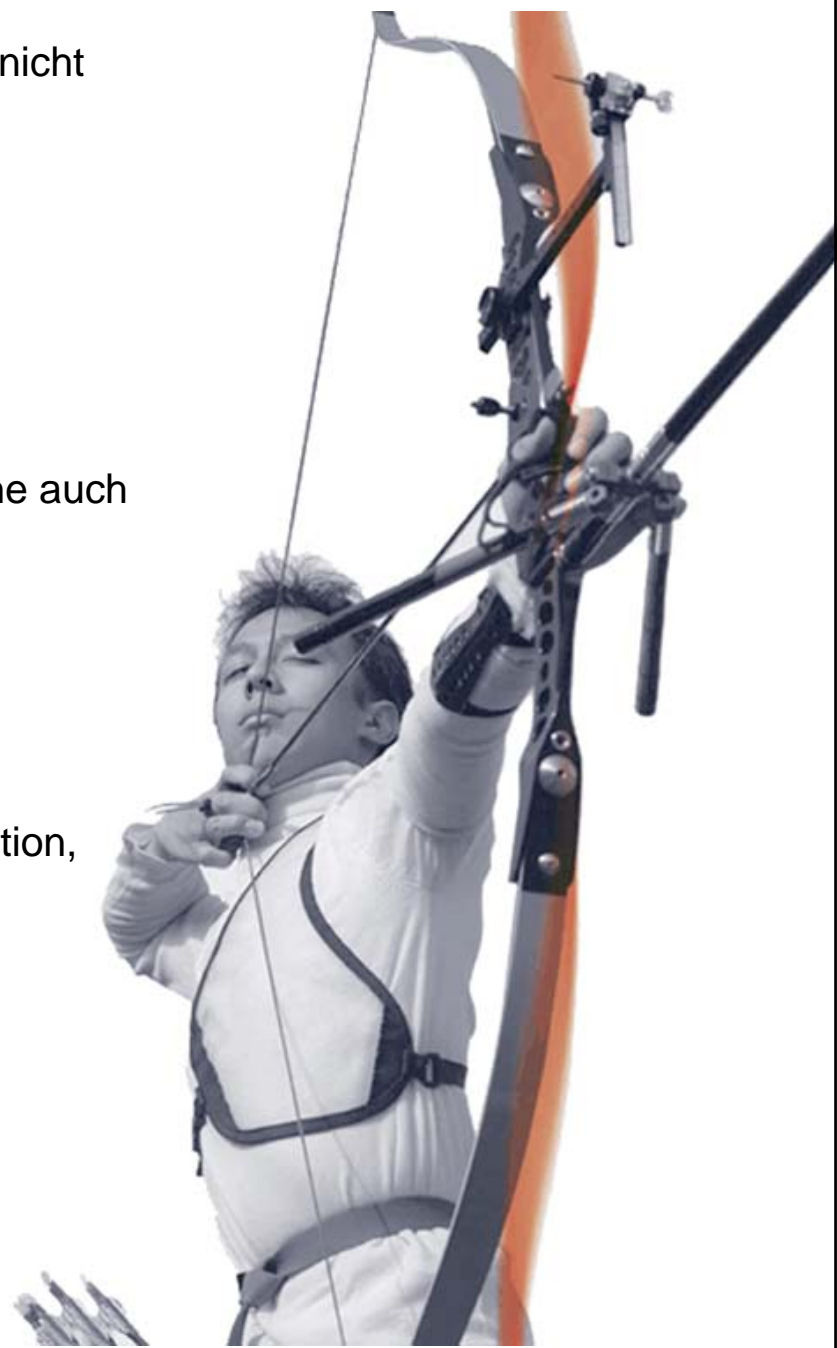
Policy Management

- ▶ Eine Enterprise-SOA erfordert einheitliche, unternehmensweite, serviceübergreifende Richtlinien
- ▶ Neben Message-Protection sind auch Authentifizierung, Zugriffskontrolle, Privacy, Auditing, ..., zu beachten
 - ▶ SAML, XACML, ...
- ▶ Erfordert hohen Standardisierungsgrad und die Wiederverwendung von Security-Lösungen
 - ▶ Security as a Service – Ansätze
 - ▶ WS-Trust, WS-Federation, SmartCards, ...
- ▶ Die Definition der Richtlinien erfolgt nach den Grundsätzen des GRC-Modells (Governance, Risk & Compliance)
 - ▶ Inkrementeller Ansatz: Allgemeingültige Aussagen müssen in konkreten IT-Richtlinien verfeinert werden

Client Policy Provisioning

- ▶ Fokus auf Message-Protection
 - ▶ WS-Security liefert die Grundmechanismen
- ▶ **WS-SecurityPolicy** liefert das Beschreibungsformat und die Choreografie-Patterns
- ▶ Via WSDL oder WS-MEX werden die Policies auf die verschiedenen Clients verteilt

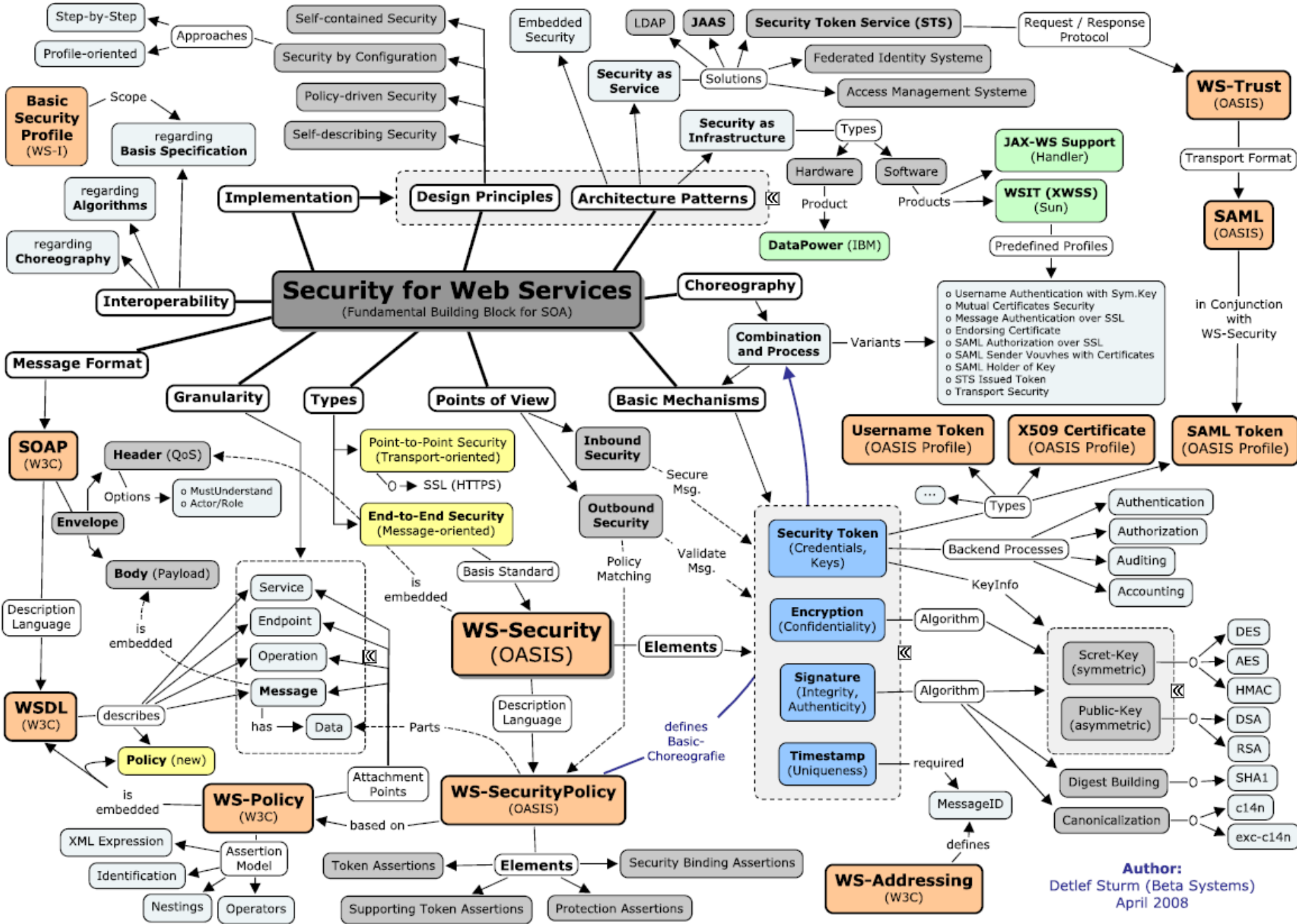
- ▶ Ohne Security kann eine SOA in geschäftskritischen Bereichen nicht betrieben werden
 - ▶ Security ist nicht nur Technologie sondern auch Business
- ▶ WS-Security (WSS) und WS-SecurityPolicy (WS-SP) sind die Basistechnologien
 - ▶ WS-Security liefert die Grundmechanismen
 - ▶ WS-SecurityPolicy liefert neben einer Beschreibungssprache auch entsprechende Choreografie-Patterns
 - ▶ Interoperabilität auf Security-Ebene umfasst nicht nur die konforme Implementierung der Algorithmen sondern auch einheitliche Choreografien
- ▶ Alle weiteren Security-Standards (SAML, WS-Trust, WS-Federation, XACML, ...) sind den mit Basistechnologien stark verzahnt
 - ▶ Ein Grundverständnis über WSS und WS-SP ist für deren Verständnis notwendig
- ▶ Für die Durchsetzung von unternehmensweiten Richtlinien ist zusätzlich ein Policy-Management erforderlich



**Vielen Dank für die
Aufmerksamkeit**



Concept Map : Security for Web Services



Author:
Detlef Sturm (Beta Systems)
April 2008