

Security-Choreografie von Web Services

Detlef Sturm

Chief Compliance Architect

Beta Systems Software AG

BITKOM SOA Forum 2009

Frankfurt, 28.05.2009



Beta Systems Software AG

- ▶ Softwareprodukte und -lösungen zur sicheren, effizienten und flexiblen Verarbeitung größter Informationsmengen
- ▶ Produkte: Data Processing, Document Processing, Security, Compliance
- ▶ Weltweit: 1.400+ Kunden, 3.300+ Installationen
- ▶ Branchen: Finanzwirtschaft, IT-Dienstleister und Industrie

Umsatz:	90,4 Mio. € (2008)
EBITDA:	10,3 Mio. € (2008)
Mitarbeiter:	> 600 weltweit
Zentrale:	Berlin

Zu meiner Person

Im Rahmen der Web Services Arbeiten / Untersuchungen (letztes Jahr):

- ▶ 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

Ab dieses Jahr:

- ▶ Chief Compliance Architect
- ▶ Verantwortlich für Design und Architektur der Compliance-Produkte
- ▶ Schwerpunkt: Compliance-Auditing im GRC-Umfeld



▶ **Motivation**

- ▶ *Bedrohungsszenarien, Anforderungen, Lösungsbausteine*
- ▶ *Grundmechanismen von WS-Security, Security-Choreografie*

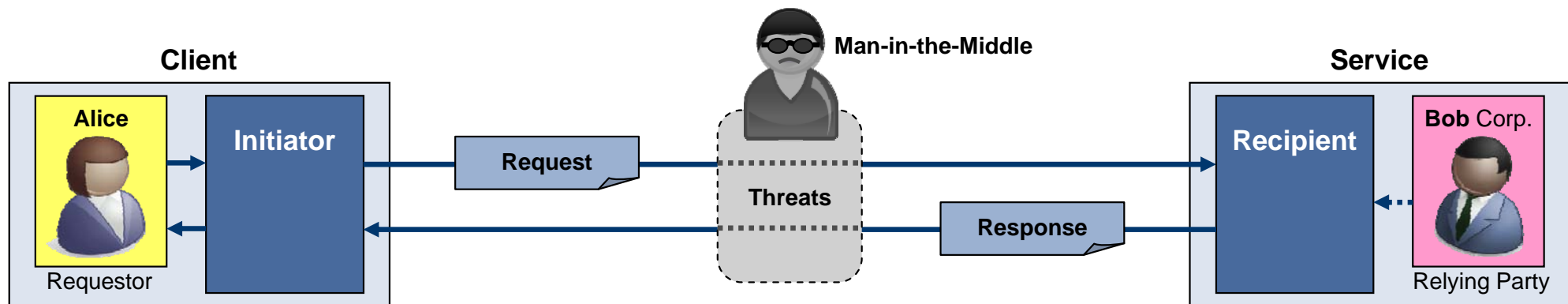
▶ **Security-Choreografie**

- ▶ *Grundsätzliche Patterns und deren Optionen*
- ▶ *Authentifizierungsoptionen*

▶ **WS-SecurityPolicy**

- ▶ *Rolle als Beschreibungssprache für Security-Anforderungen*
- ▶ *Assertion-Struktur und Abbildung der Choreografien*





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
 - Wickelt im Auftrag von Alice die Kommunikation ab
 - Requester: Service-Benutzer (z.B. Alice)

3. Nachrichtenaustausch

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

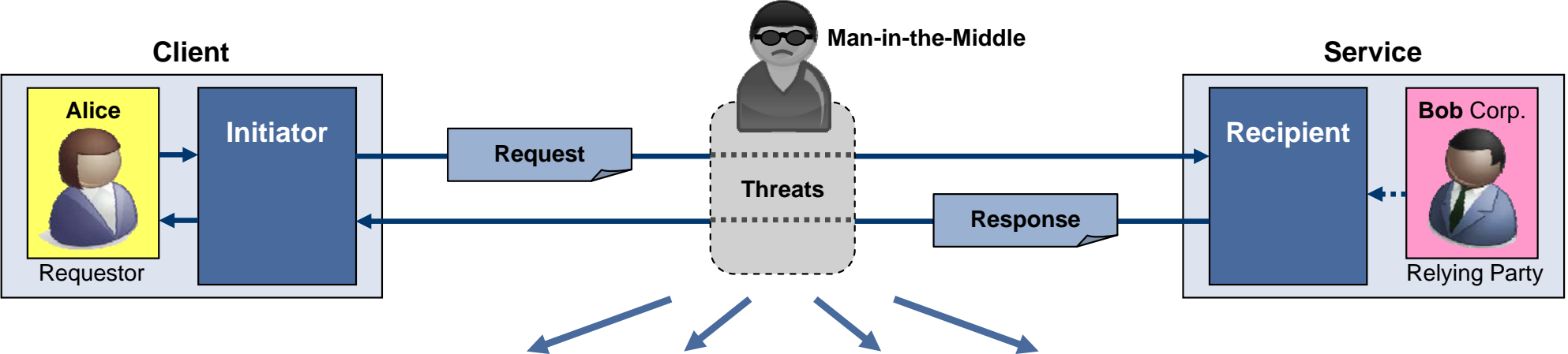
4. Man-in-the-Middle

- Es gibt nicht die heile Welt...

Bringing Civilization to its Knees

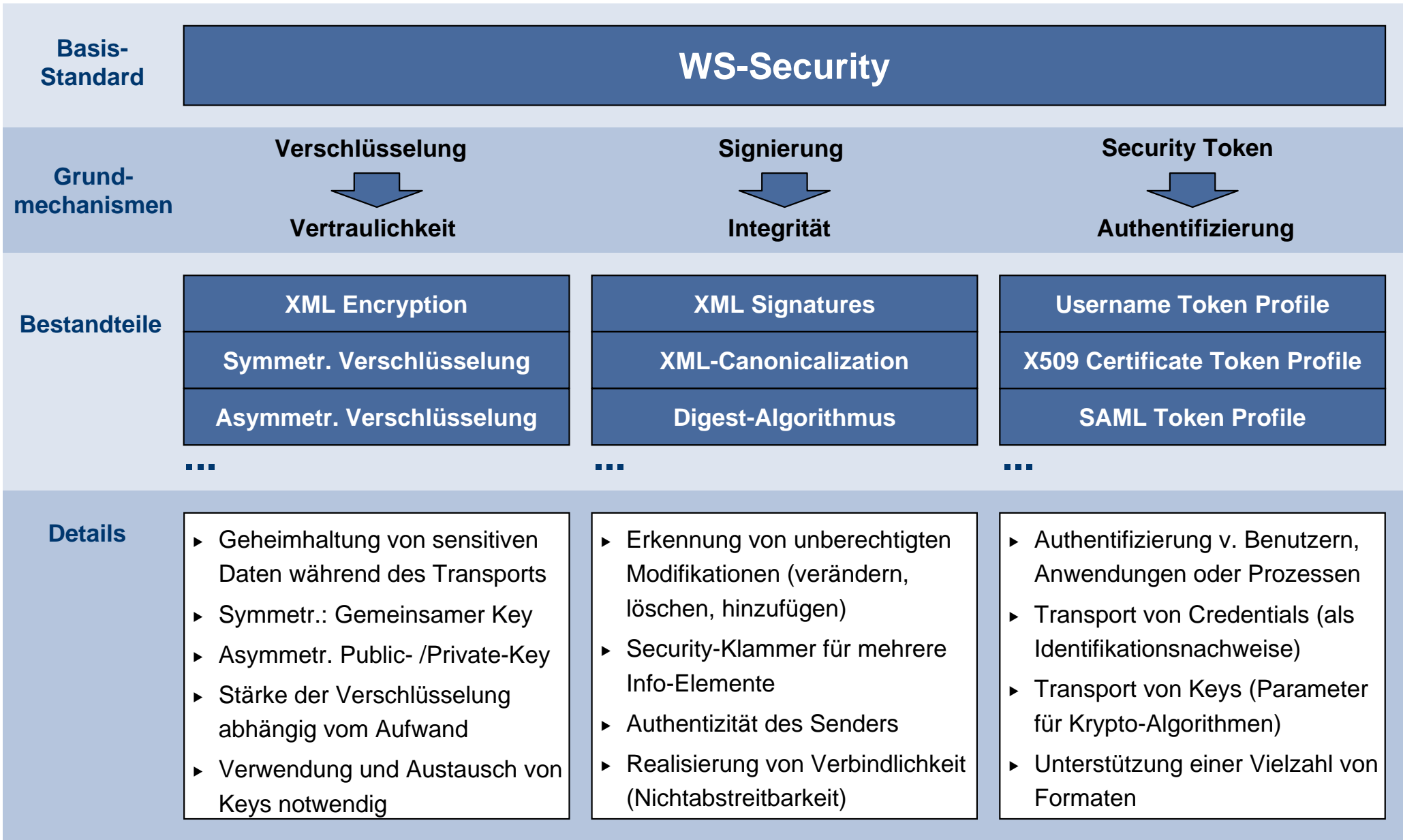


Bedrohungen, Anforderungen, Lösungsbausteine



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) ...

Grundmechanismen von WS-Security



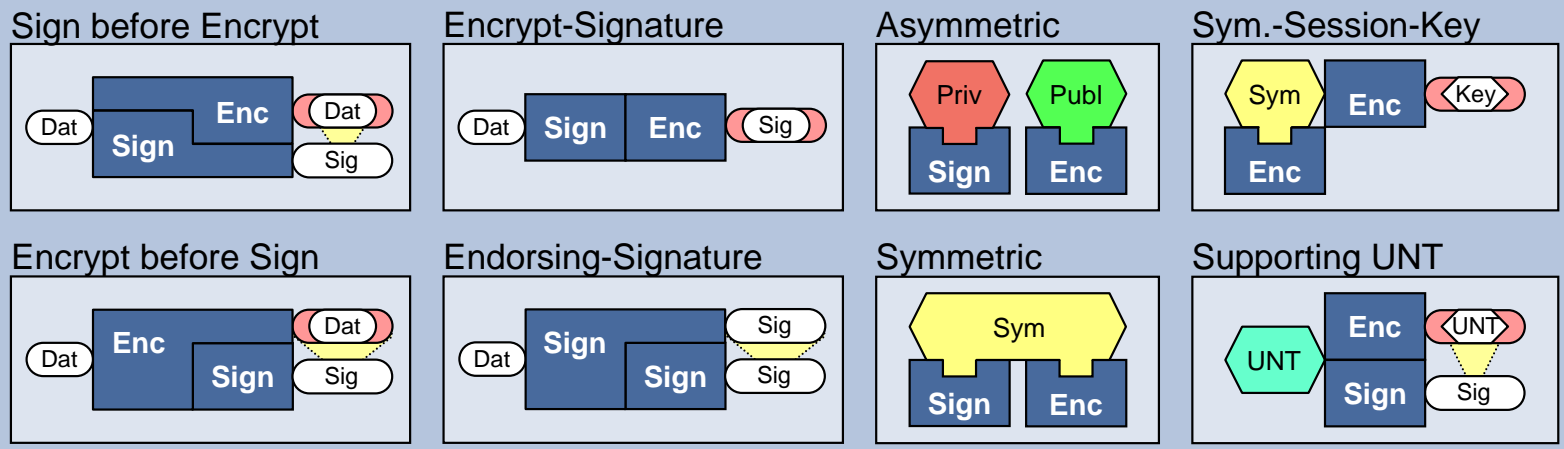
(Timestamp ist ein weiterer Mechanismus im Rahmen von WS-Security: Verhinderung von Replay-Attacken)

WS-Security

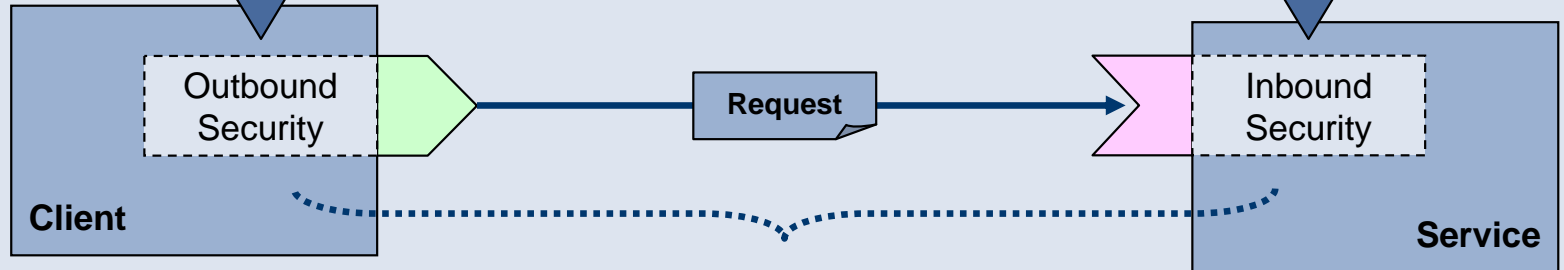
Grundmechanismen für Security



Security = Kombination der einzelnen Mechanismen

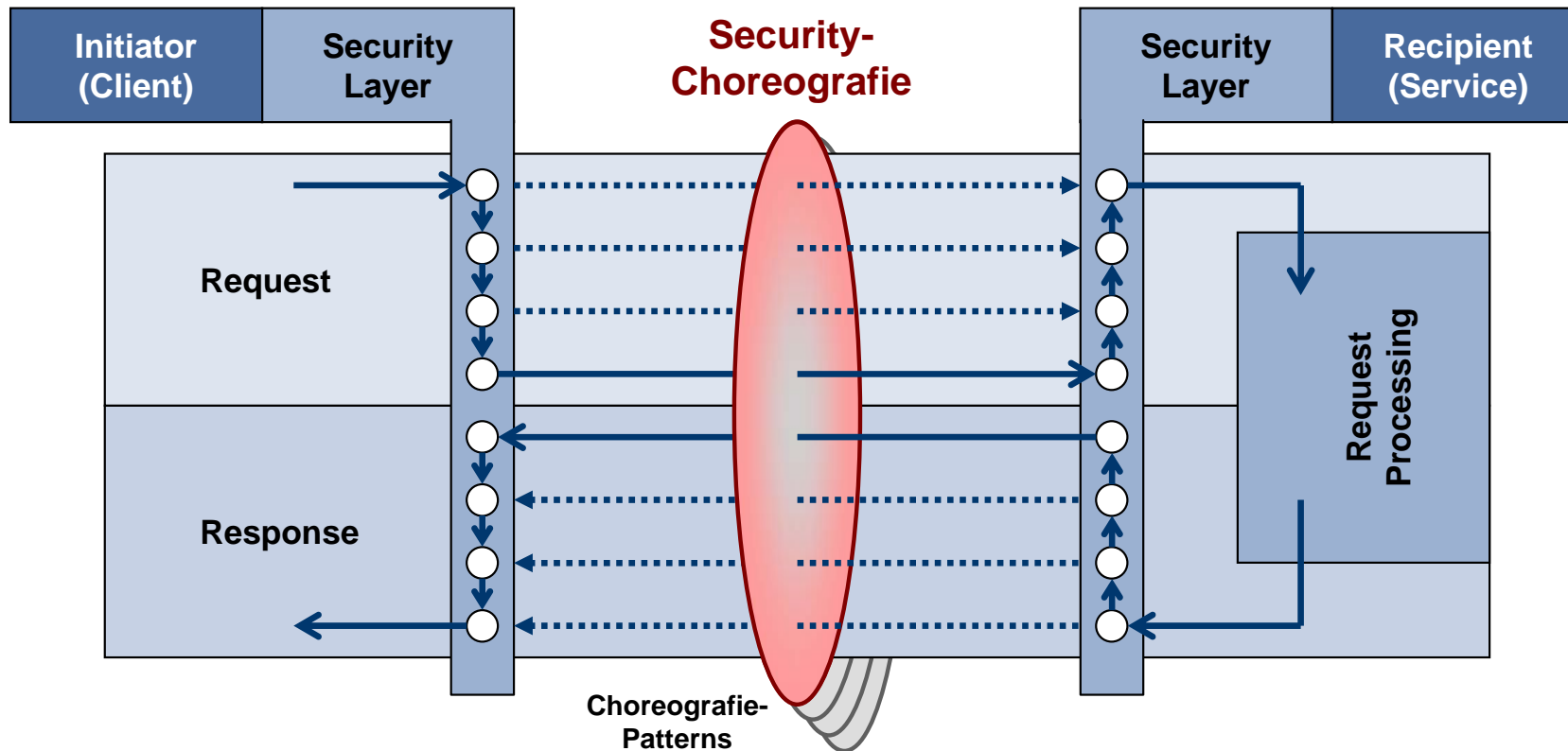


Interoperabilität benötigt korrespondierendes Verhalten



Security-Choreografie

WS-SecurityPolicy



Security-Choreografie:

- ▶ Abgestimmtes Verhalten zwischen den Kommunikationspartnern Client und Service bezüglich:
 - ▶ Arten der Security-Mechanismen (Element)
 - ▶ Reihenfolge der Abarbeitung (Chronologie)
- ▶ Service wird i.d.R. die Choreografie vorgeben
- ▶ Nicht verwechseln mit WS-Choreography (Beschreibungssprache für den Nachrichtenfluss)
- ▶ Resultierend aus den Kombinationsmöglichkeiten existieren eine Vielzahl von Choreografien
 - ▶ Etablierung von Choreografie-Patterns

Asymmetric Binding

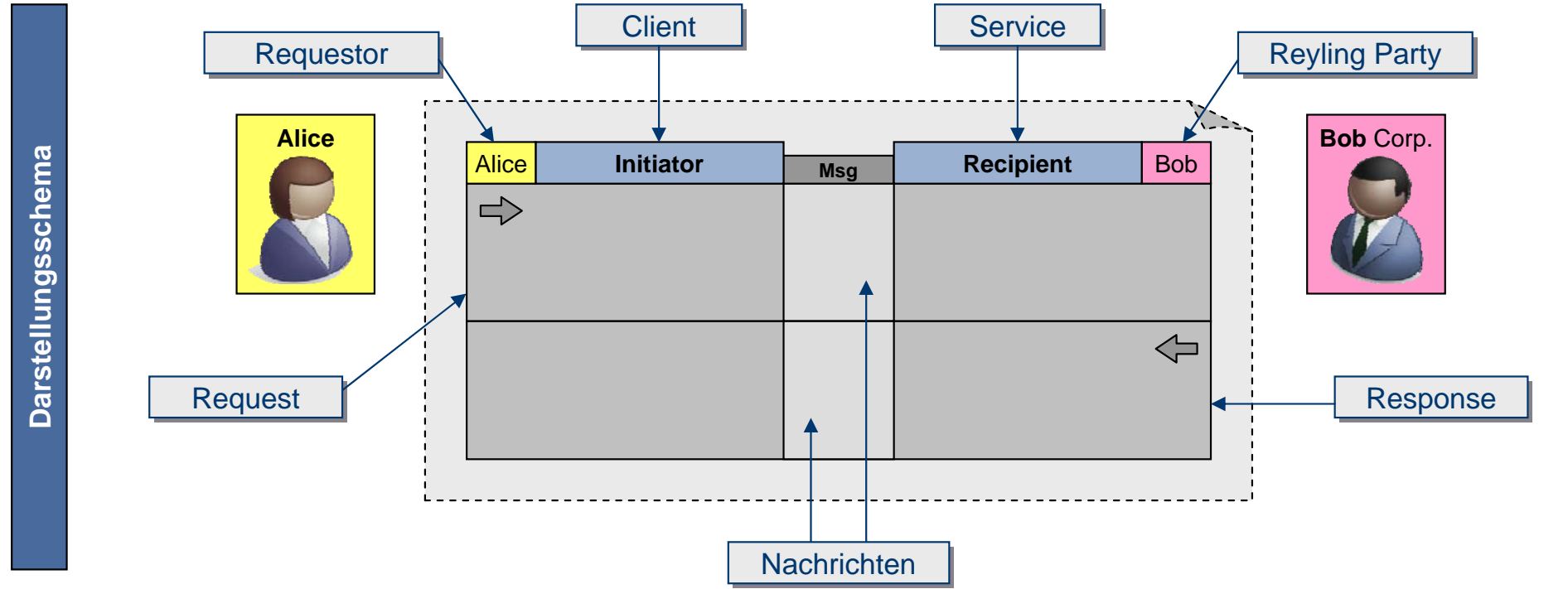
Symmetric Binding &
Endorsing Signature

Symmetric Binding

Symmetric Binding &
Username Token

- ▶ Gemeinsame Ziele:
 - ▶ Primär: Sicherstellung der Vertraulichkeit und Integrität von Daten
 - ▶ Zusätzlich: Authentifizierung des Senders und/oder des Empfängers
- ▶ Pattern unterscheiden sich bzgl. der Umsetzung:
 - ▶ Verwendung der Grundmechanismen: Encryption und Signatures
 - ▶ Unterschiedliches Key-Handling: Symmetrische oder asymmetrische Verfahren
 - ▶ Wer muss welchen Schlüssel bzw. Zertifikat besitzen?
- ▶ Kenntnisse über diese Zusammenhänge sind für das Verständnis hinsichtlich der Wirkungsweise und Konfigurationsmöglichkeiten der verschiedenen Web Service Umgebungen notwendig

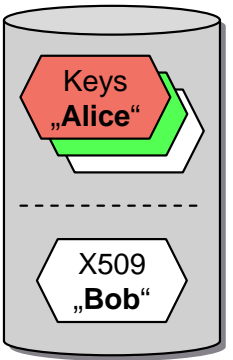
Choreografie-Pattern: Darstellungsschema und Symbole



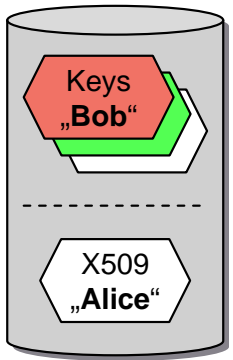
Symbole

Security Mechanisms	Token	Data
Enc Encryption	Publ Public Key	Data (Payload, Signature, Key, ...)
Dec Decryption	Priv Private Key	Encrypted Data
Sign Signature Creation	Sym Symmetric Key	Dat
Val Signature Validierung	Drvd Derived Key	Sig Signatures
	UNT Username Token	

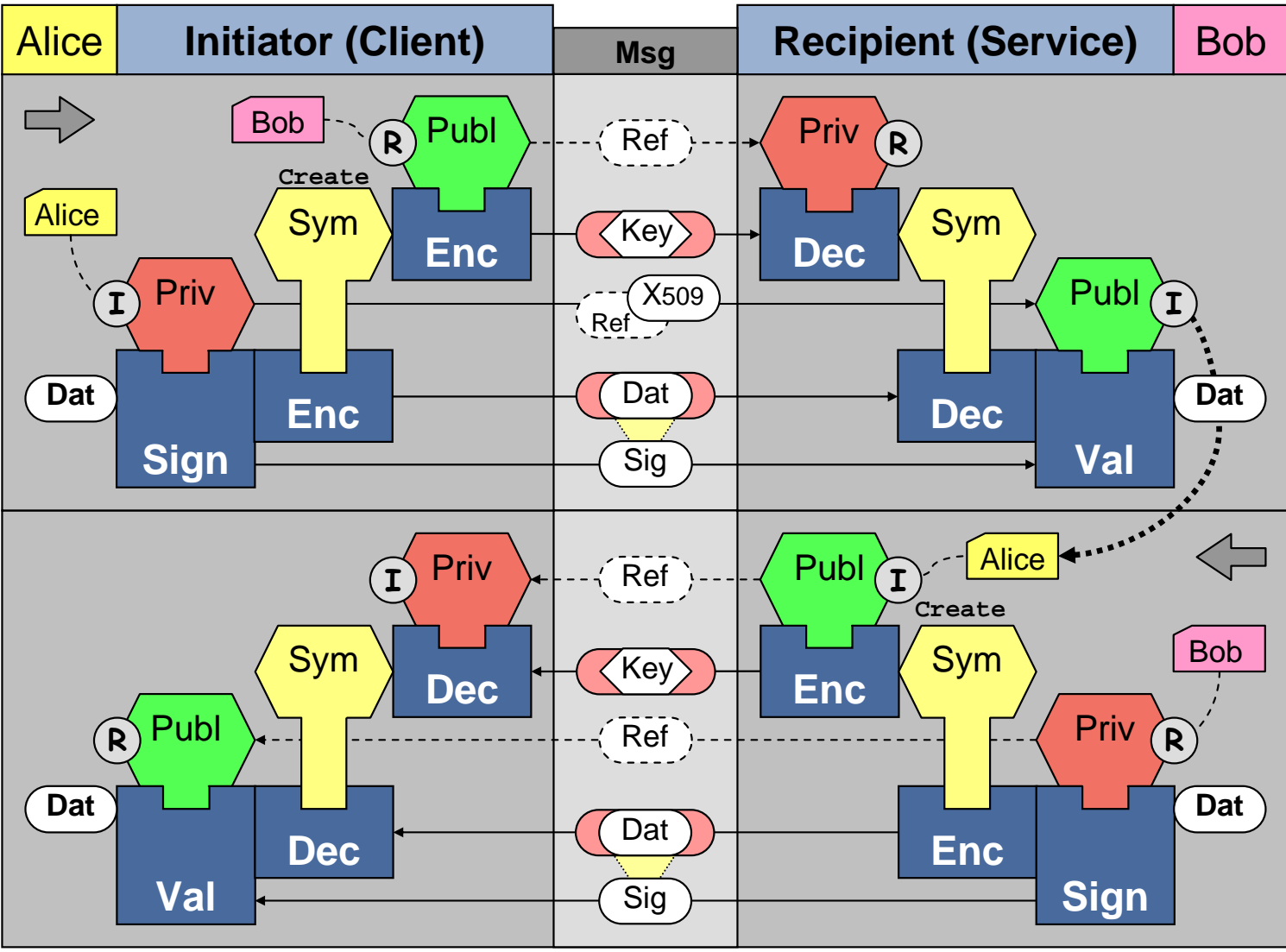
Choreografie-Pattern #1: Asymmetric Binding



Key- & Certificates-Store



Key- & Certificates-Store

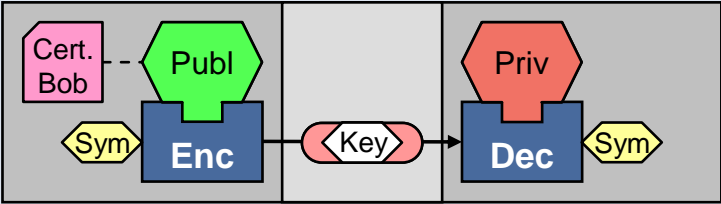
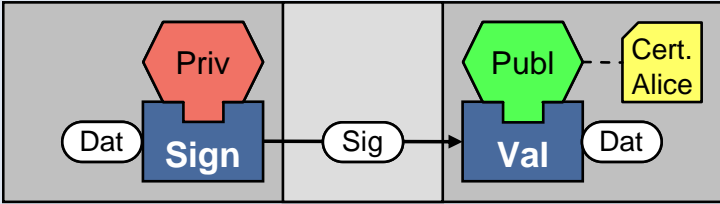


WS-SecurityPolicy:

- (I)** Initiator Token
- (R)** Recipient Token

→ AuthN-Möglichkeiten

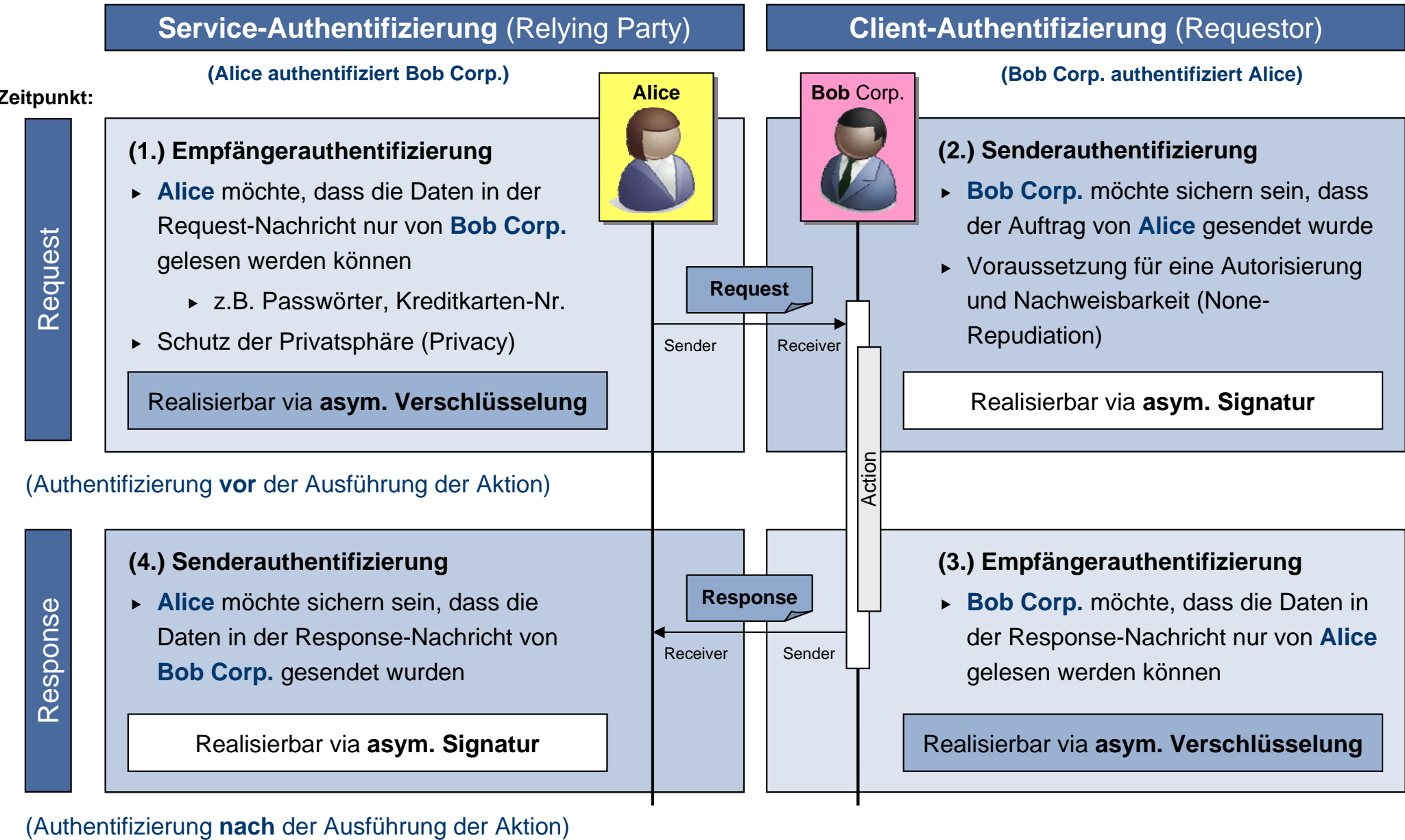
Authentifizierungsmöglichkeiten aufgrund von kryptografischen Algorithmen

	Verschlüsselung	Signaturen
Ziele	Vertraulichkeit	Unverfälschtheit (Integrität)
Asym. Algorithmen		
Relation zw. den Partnern	mehrere Sender $n : 1$ ein Empfänger	ein Sender $1 : n$ mehrere Empfänger
Authentifizierungsoptionen	Authentifizierung des Empfängers (Bob)	Authentifizierung des Senders (Alice)

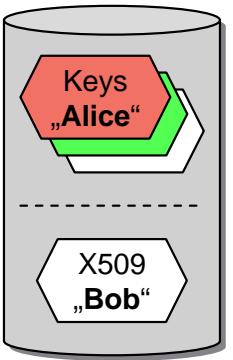
Fazit:

- ▶ Asymmetrische Algorithmen bieten von Haus aus Zertifikats-basierte Authentifizierungsmöglichkeiten
- ▶ Umkehrschluss: Zertifikats-basierte Authentifizierungen basieren auf die asymmetrischen Algorithmen für Verschlüsselung und Signaturen

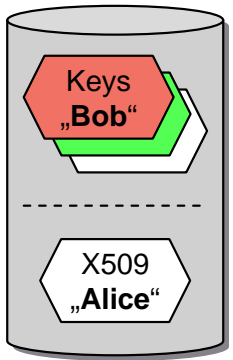
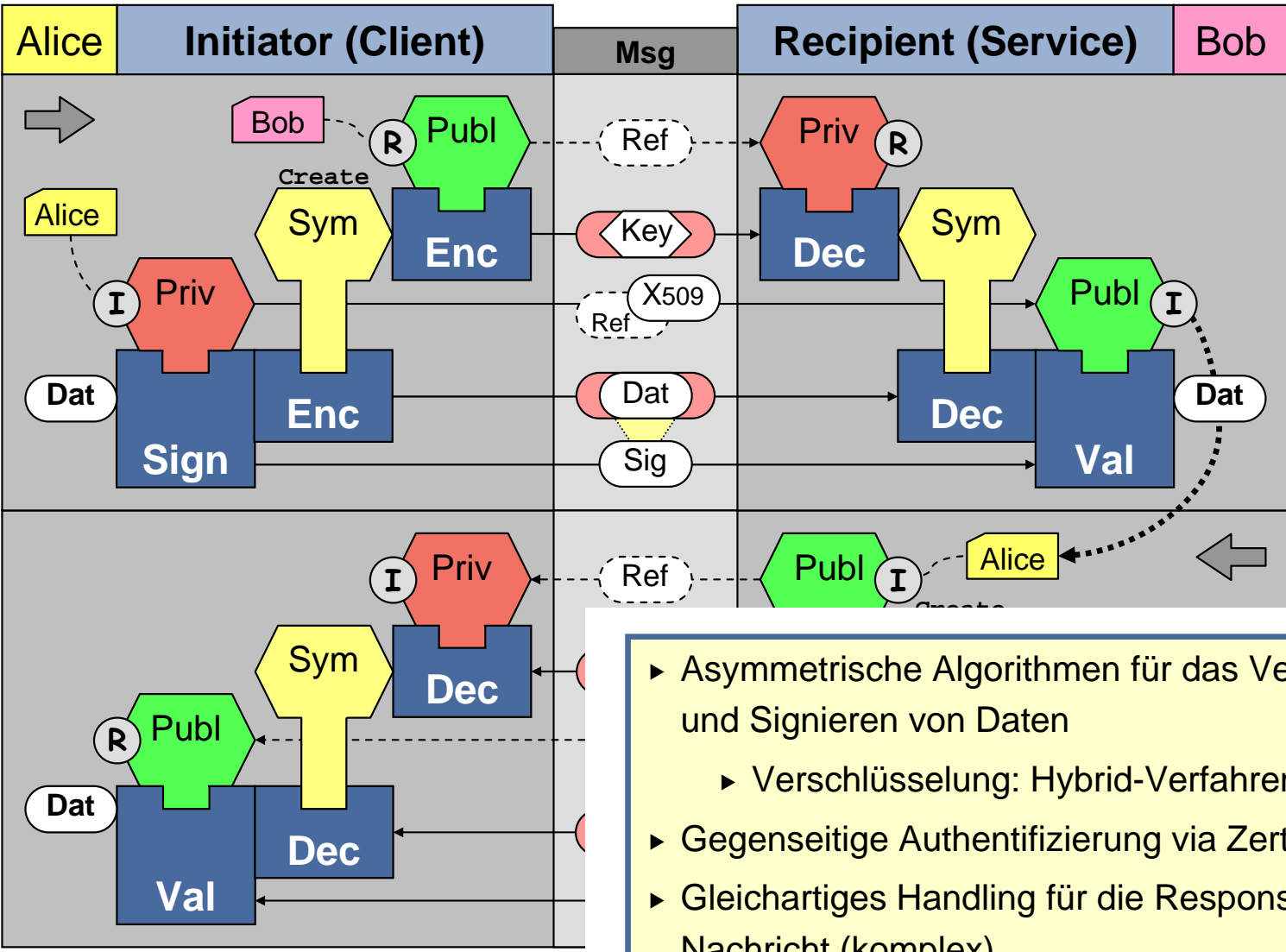
Zeitliche Betrachtung der Authentifizierungsvarianten



Choreografie-Pattern #1: Asymmetric Binding



Key- & Certificates-Store



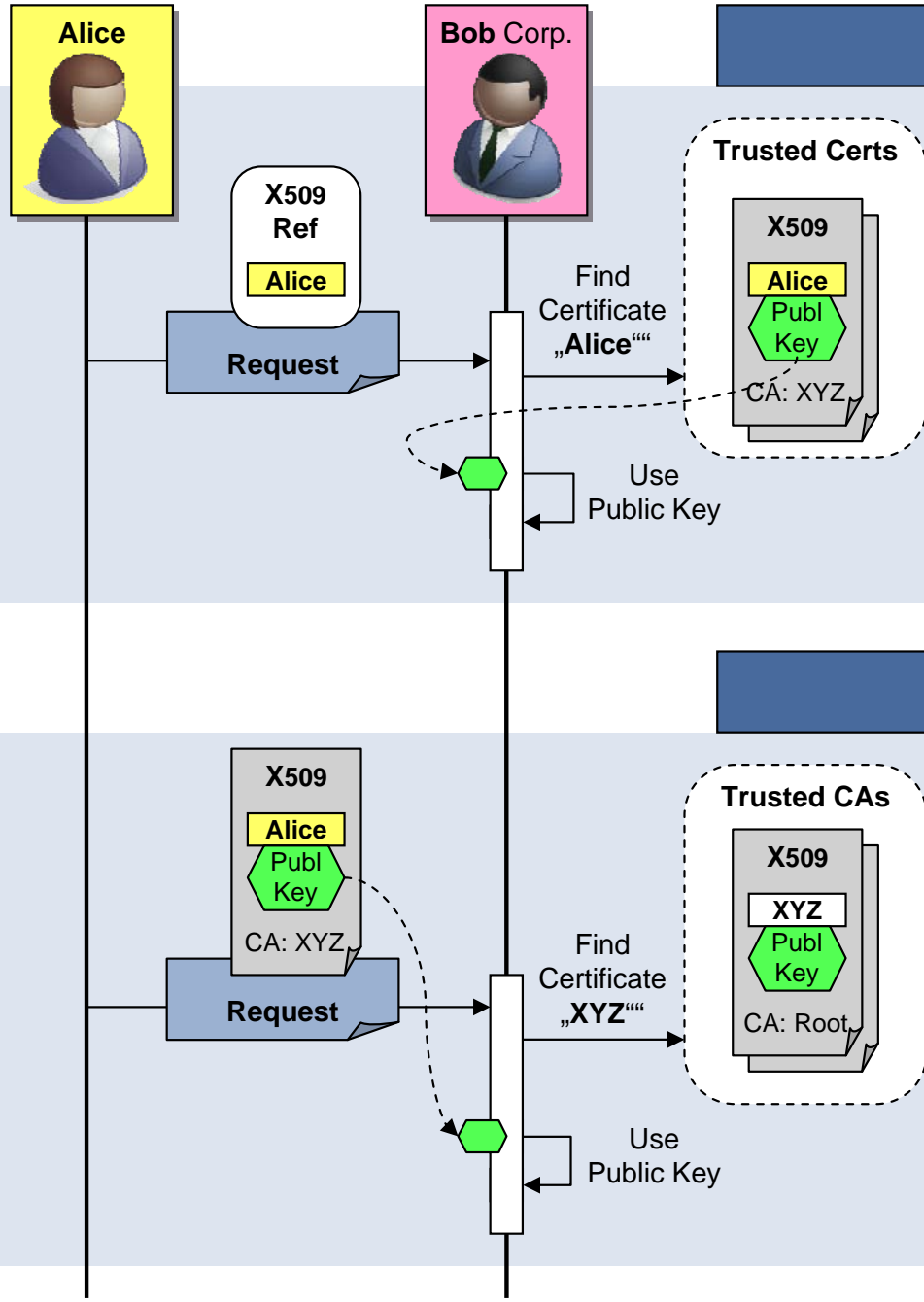
Key- & Certificates-Store

- ▶ Asymmetrische Algorithmen für das Verschlüsseln und Signieren von Daten
 - ▶ Verschlüsselung: Hybrid-Verfahren
- ▶ Gegenseitige Authentifizierung via Zertifikate
- ▶ Gleichartiges Handling für die Response-Nachricht (komplex)
 - ▶ Redundante Authentifizierung
 - ▶ Geeignet für asynchrone Kommunikationen

WS-SecurityPolicy:

(I) Initiator Token (R) Recipient Token

Arten für die Bekanntgabe des verwendeten Keys (via Zertifikate)



als Key-Info (Referenz)

- ▶ Alice schickt eine Referenzangabe über ihr Zertifikat
 - ▶ Varianten: Sub Key Identifier, Thumbprint, Issuer Name & Serial Number
- ▶ Bob muss im Besitz des Zertifikates von Alice sein, um ihren Public Key verwenden zu können
- ▶ **Bob muss Alice kennen**

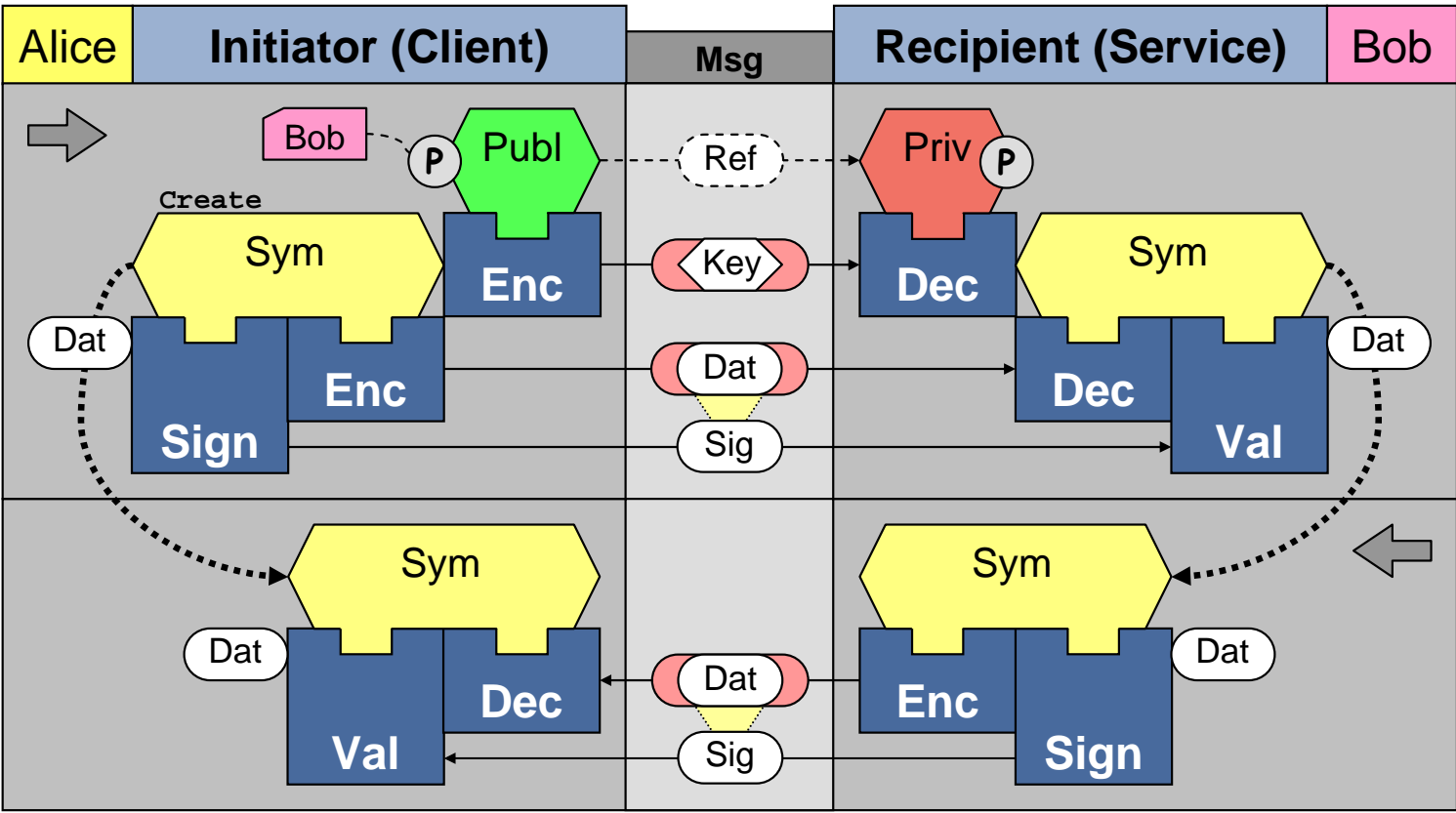
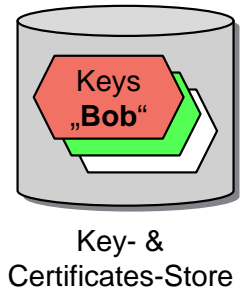
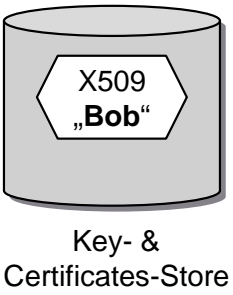
Peer Authentication

als X509-Token

- ▶ Alice schickt komplettes Zertifikat
 - ▶ Als Binary-Token
- ▶ Bob überprüft Vertrauenswürdigkeit über die Vertrauenskette der Zertifikate
- ▶ **Bob muss Security-Provider (CA) von Alice kennen**

Trust-Chain Authentication

Choreografie-Pattern #2: Symmetric Binding

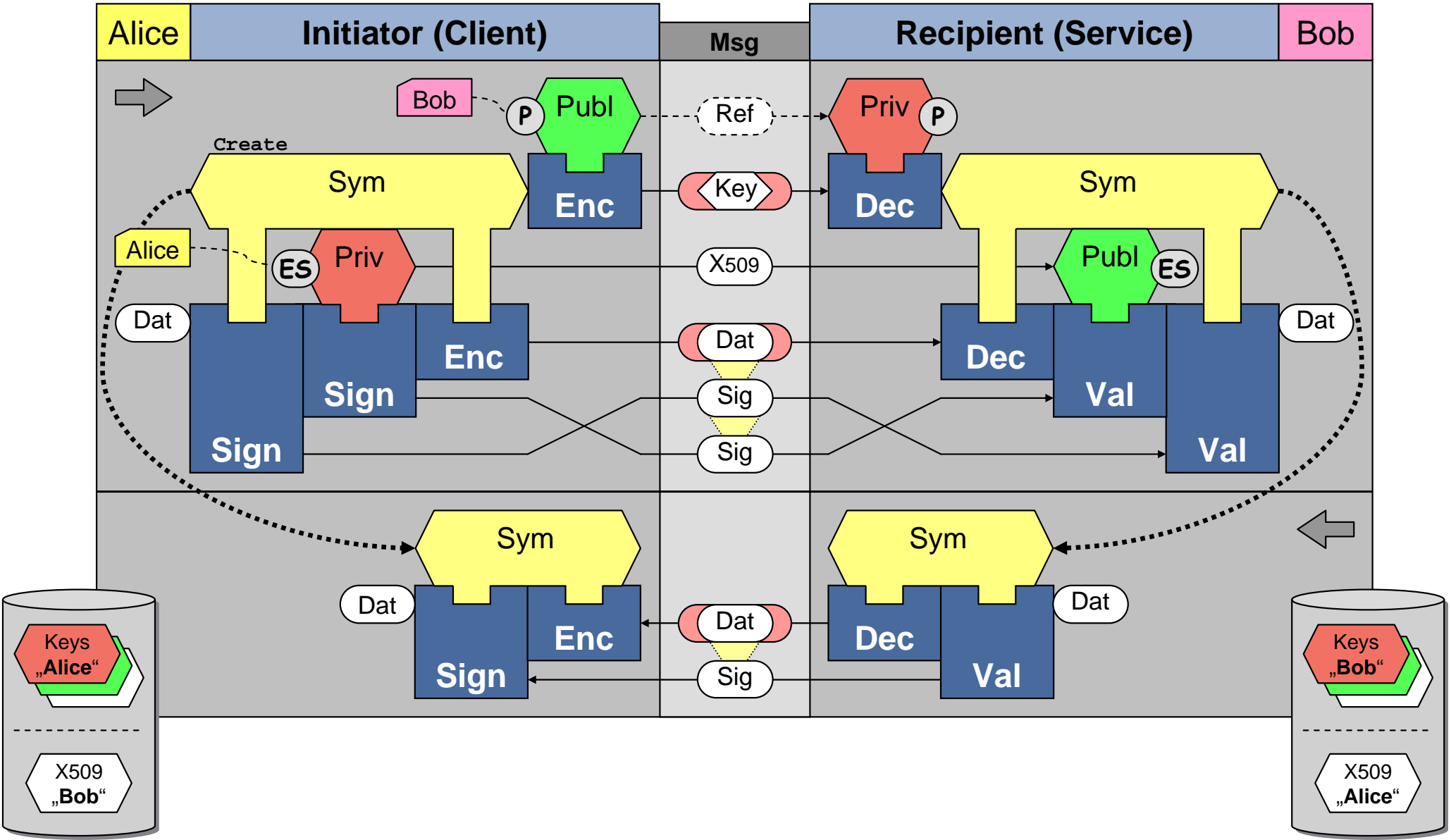


- ▶ Gemeinsamer symmetrischer Schlüssel für Verschlüsselung und Signaturen
- ▶ Keine Client-Authentifizierung (anonymer Client)
- ▶ Einfaches Handling für die Response-Nachricht
 - ▶ Erfordert die Kopplung von Request und Response (Session)
 - ▶ Dadurch keine redundanten Authentifizierungen

WS-SecurityPolicy:

(P) Protection Token

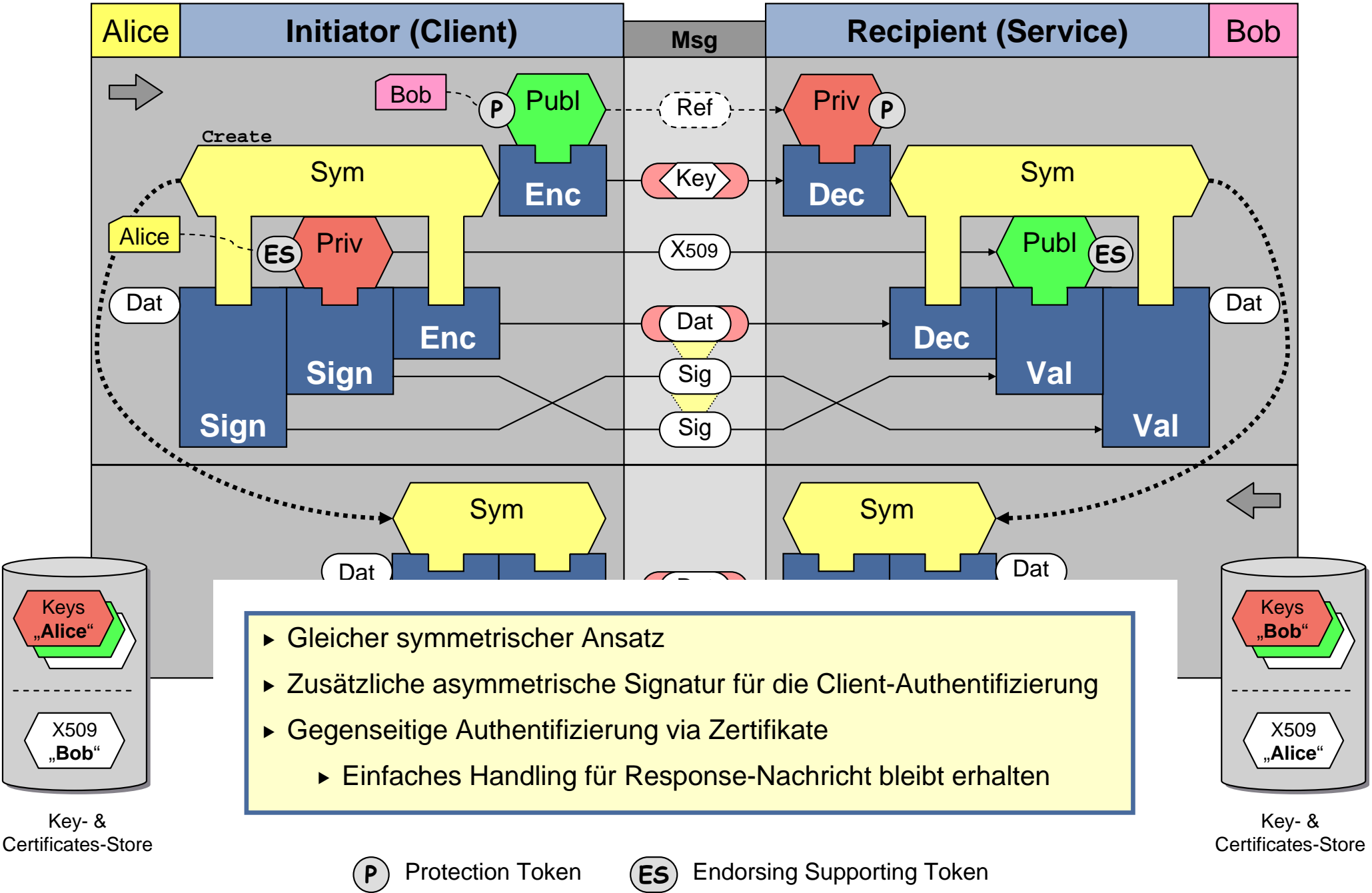
Choreografie-Pattern #3: Symmetric Binding & Endorsing Signature



WS-SecurityPolicy:

- (P) Protection Token
- (ES) Endorsing Supporting Token

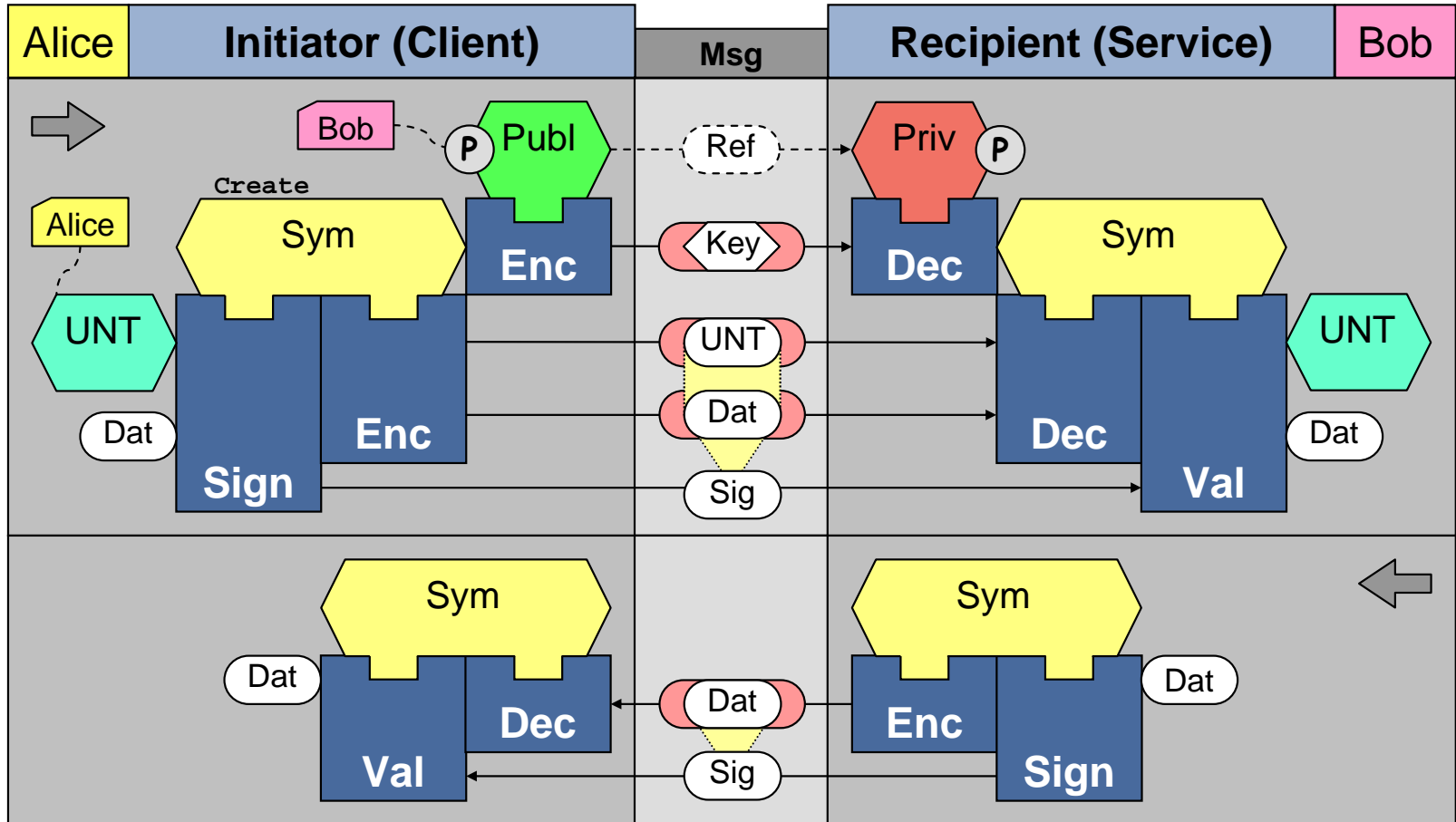
Choreografie-Pattern #3: Symmetric Binding & Endorsing Signature



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

(P) Protection Token (ES) Endorsing Supporting Token

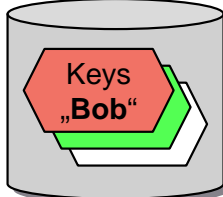
Choreografie-Pattern #4: Symmetric Binding & Username Token



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



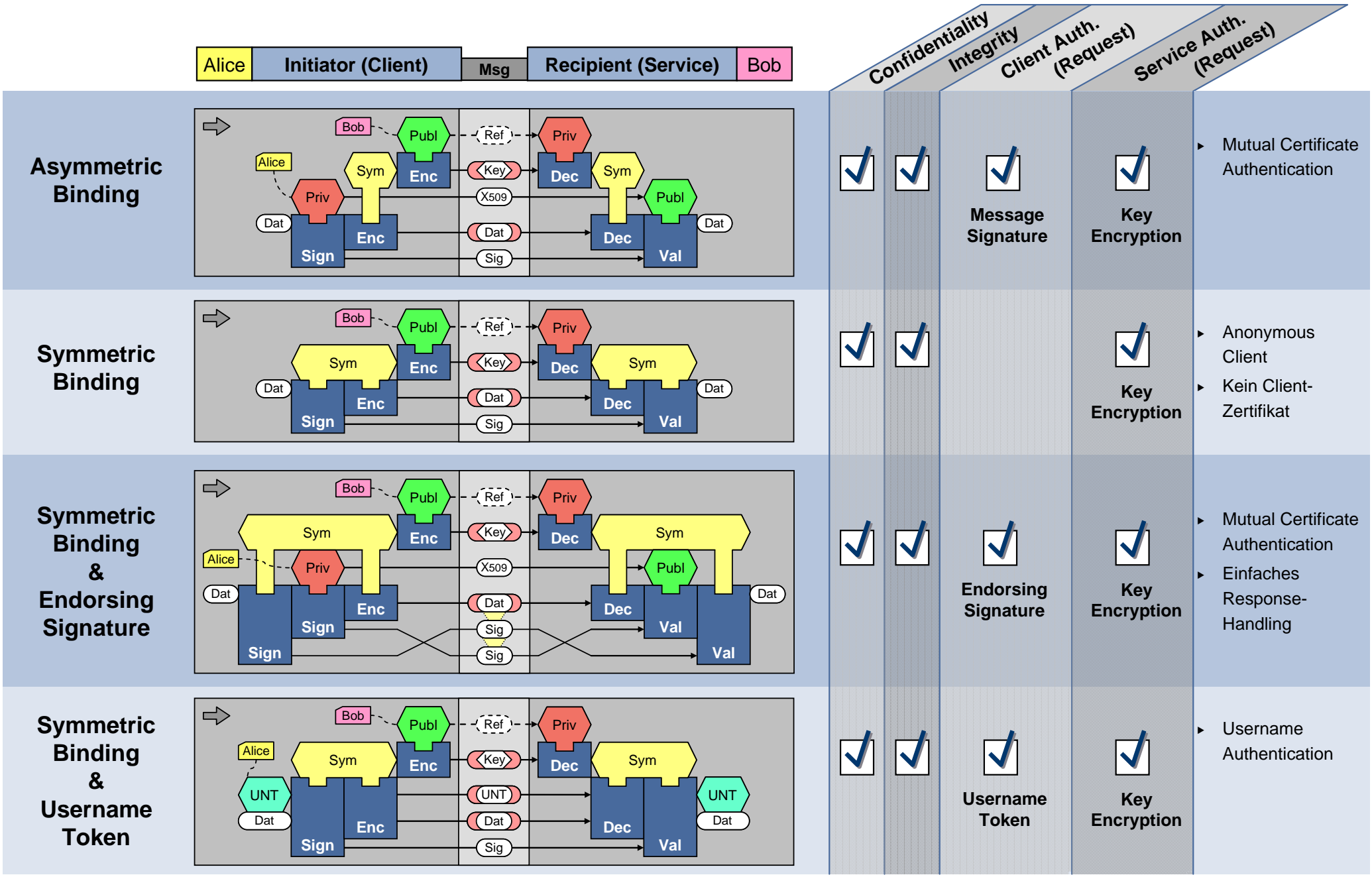
Key- & Certificates-Store



Key- & Certificates-Store

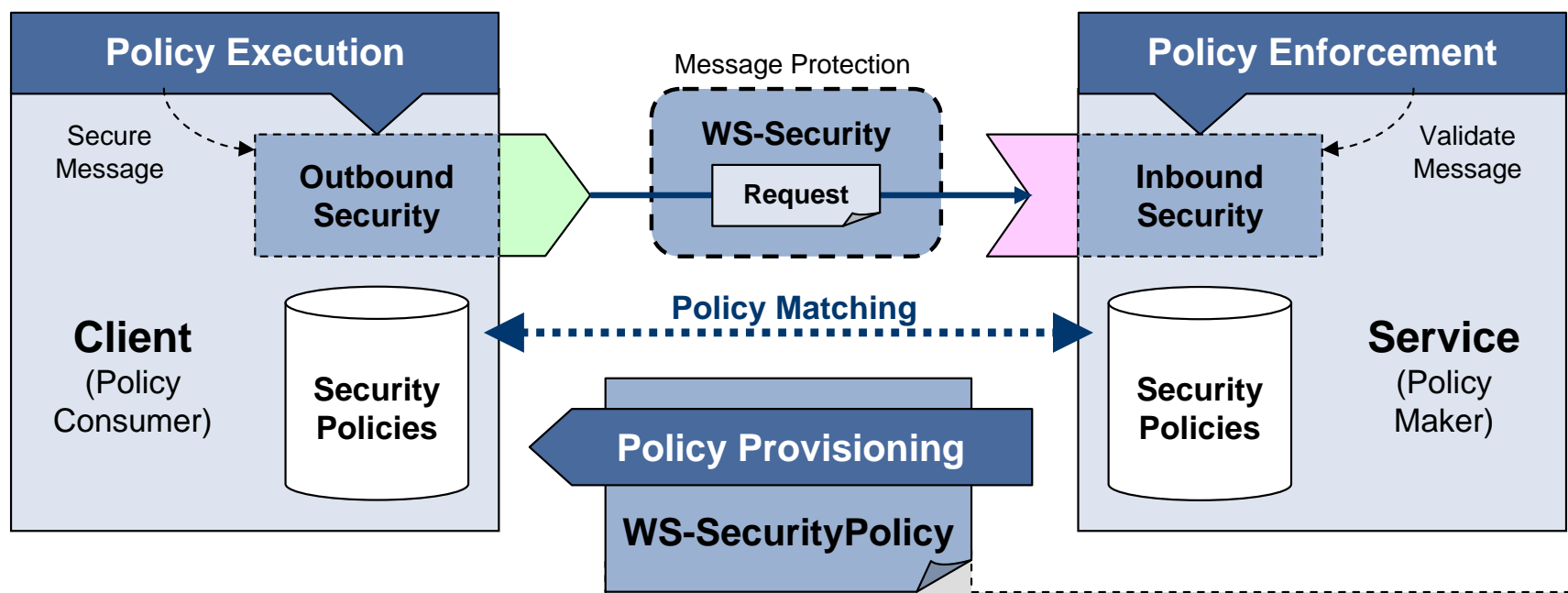
(P) Protection Token (UNT) Username Token

Gegenüberstellung: Choreografie-Patterns



→ Policy-Beschreibung

Policy-Beschreibungen: Grundlage für eine korrespondierende Security-Choreografie



Policy Provisioning:

- ▶ Adressiert die Verteilung der Policies zu den einzelnen Clients

Policy Execution:

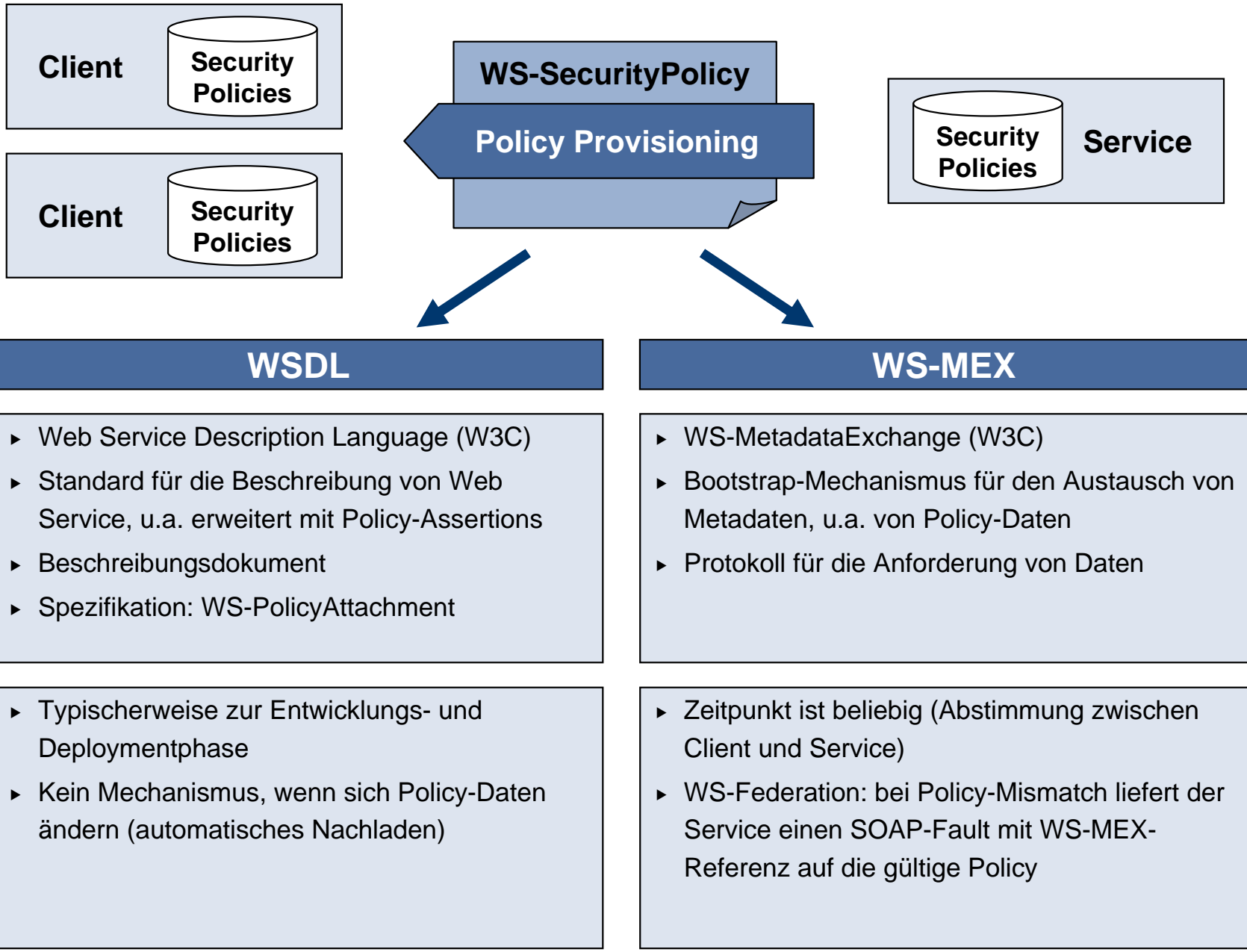
- ▶ Policies werden vom Client entsprechend umgesetzt

Policy Enforcement:

- ▶ Service kontrolliert die konforme Umsetzung der Policies

- ▶ Bestandteil einer umfangreichen Familie bestehend aus mehreren Policy-Spezifikationen
 - ▶ WS-Policy als Basisstandard bietet eine allgemeine Beschreibungssprache für die Beschreibung von in Form von Assertions
- ▶ WS-SecurityPolicy: Beschreibung der Security-Eigenschaften von SOAP-Nachrichten
 - ▶ Basiert auf den Mechanismen von WS-Security, WS-Trust und WS-SecureConversation

Ansätze für das Client-Policy-Provisioning



Ansatz:

- ▶ Web Service Description Language (W3C)
- ▶ Standard für die Beschreibung von Web Service, u.a. erweitert mit Policy-Assertions
- ▶ Beschreibungsdokument
- ▶ Spezifikation: WS-PolicyAttachment

- ▶ WS-MetadataExchange (W3C)
- ▶ Bootstrap-Mechanismus für den Austausch von Metadaten, u.a. von Policy-Daten
- ▶ Protokoll für die Anforderung von Daten

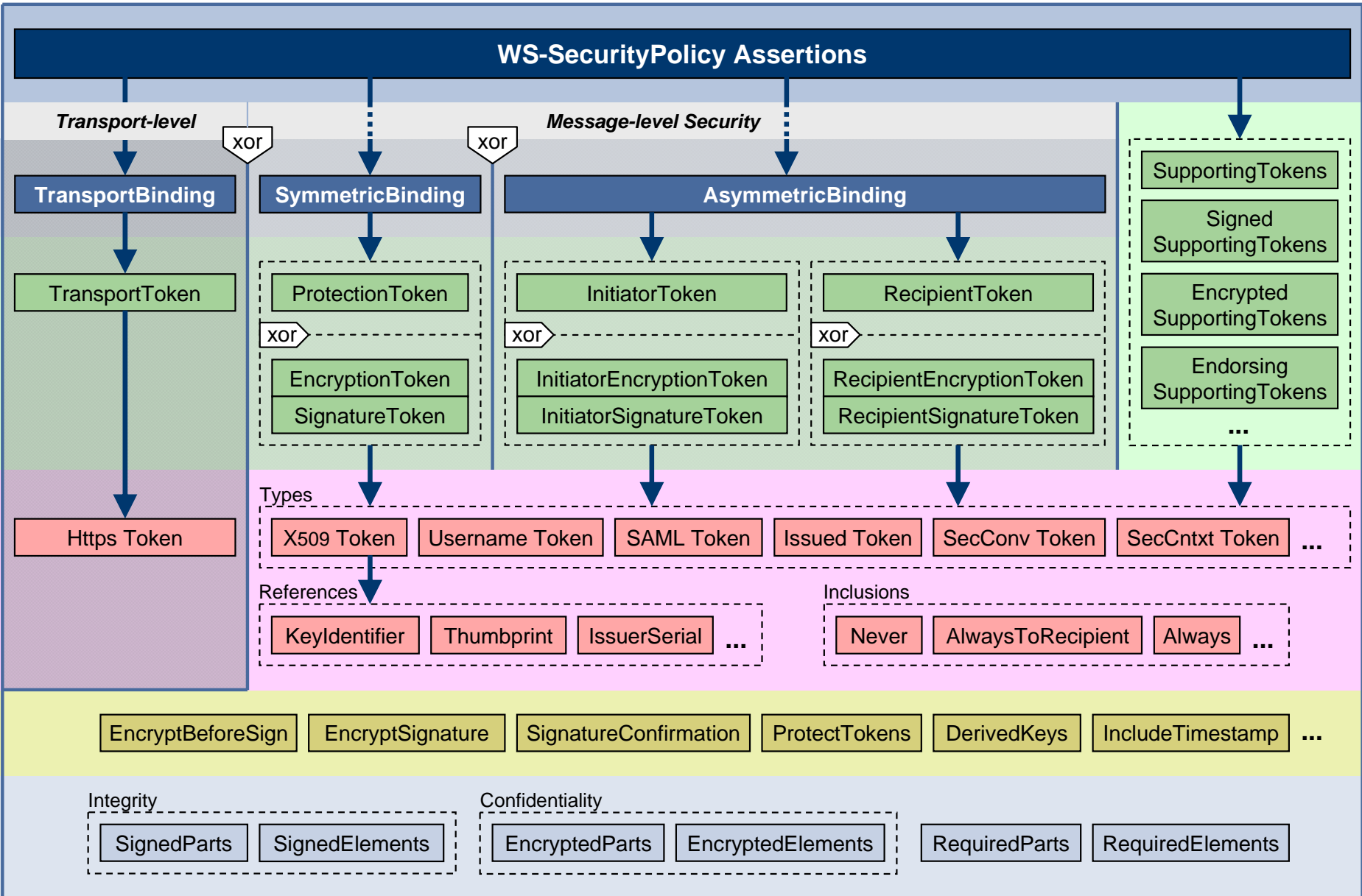
Zeitpunkt des Austausches:

- ▶ Typischerweise zur Entwicklungs- und Deploymentphase
- ▶ Kein Mechanismus, wenn sich Policy-Daten ändern (automatisches Nachladen)

- ▶ 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

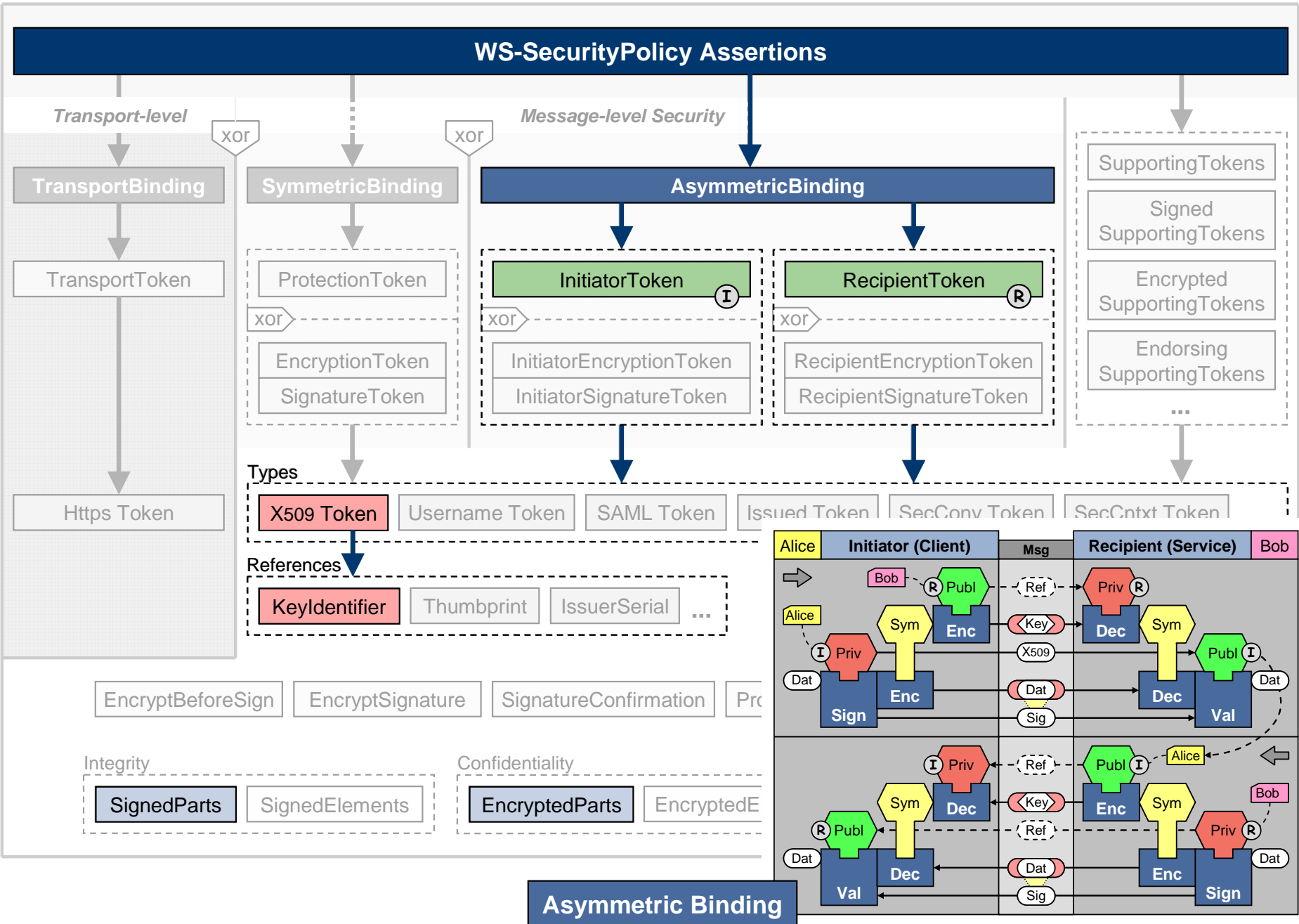
→ Assertion-Struktur

Assertion-Struktur von WS-SecurityPolicy



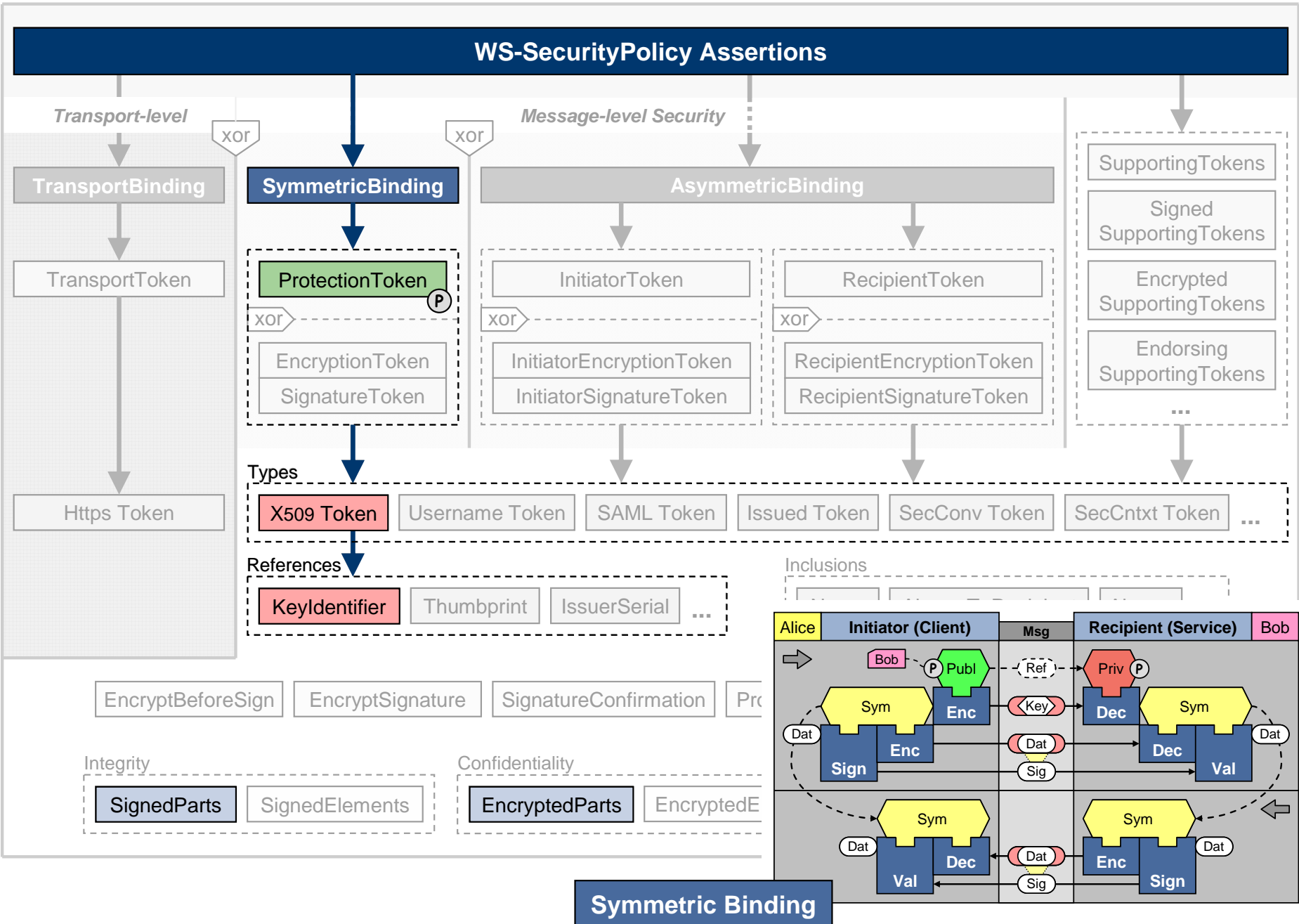
→ Assertions pro Pattern

Assertions für Choreografie „Asymmetric Binding“



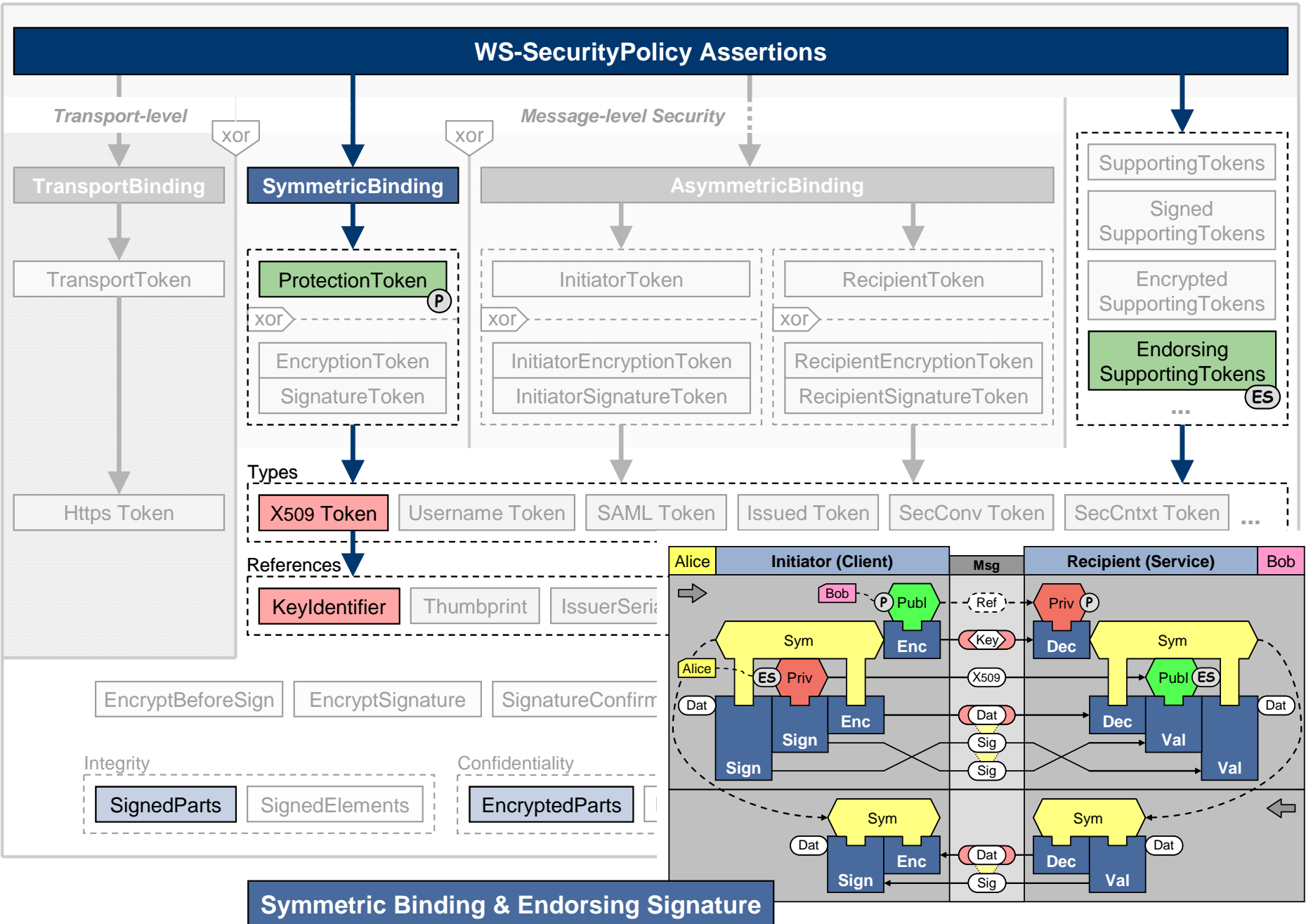
Asymmetric Binding

Assertions für Choreografie „Symmetric Binding“



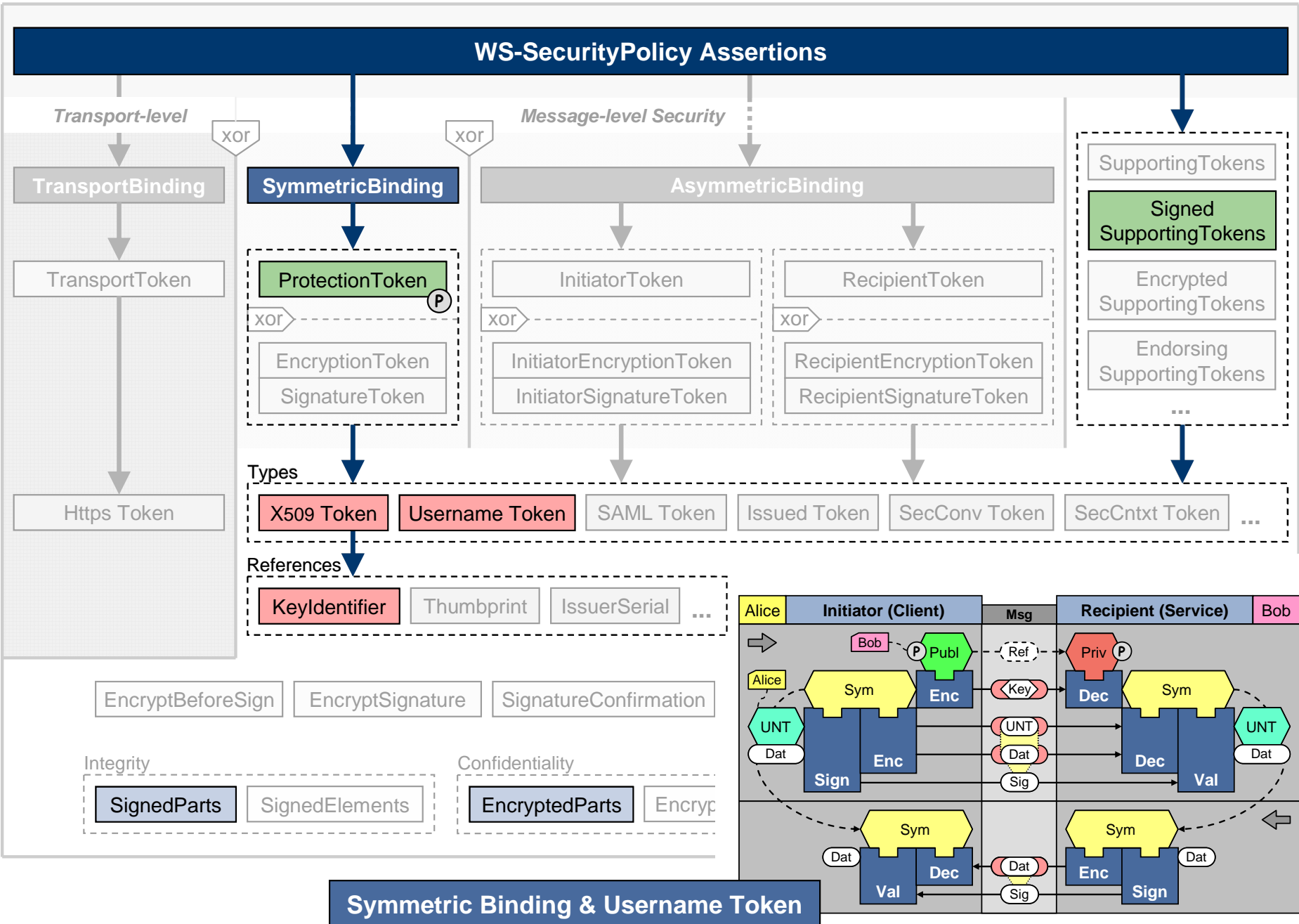
Symmetric Binding

Assertions für Choreografie „Symmetric Binding & Endorsing Signature“



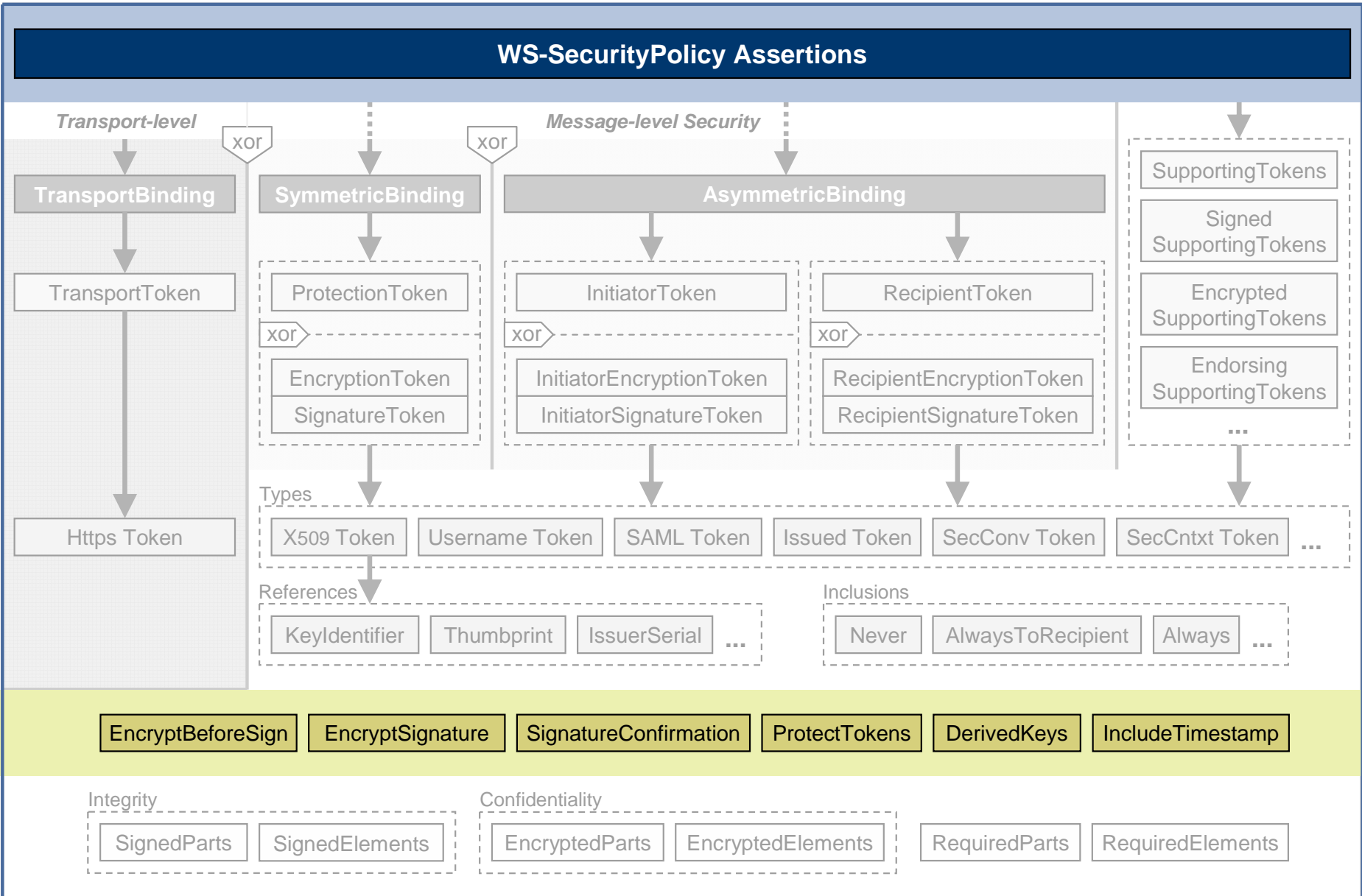
Symmetric Binding & Endorsing Signature

Assertions für Choreografie „Symmetric Binding & Username Token“



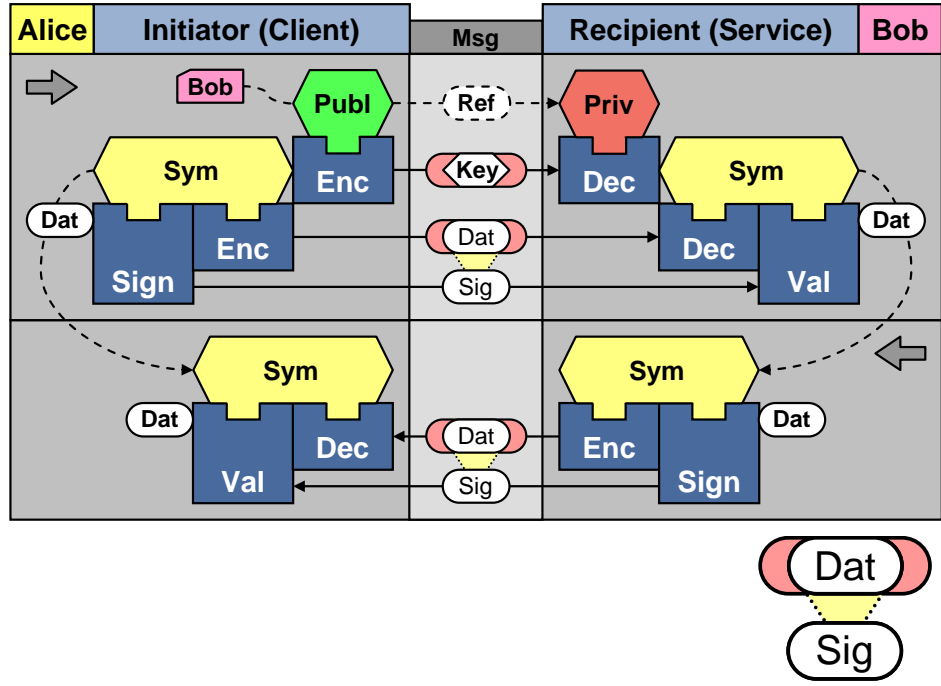
Symmetric Binding & Username Token

Assertions für Choreografie-Optionen

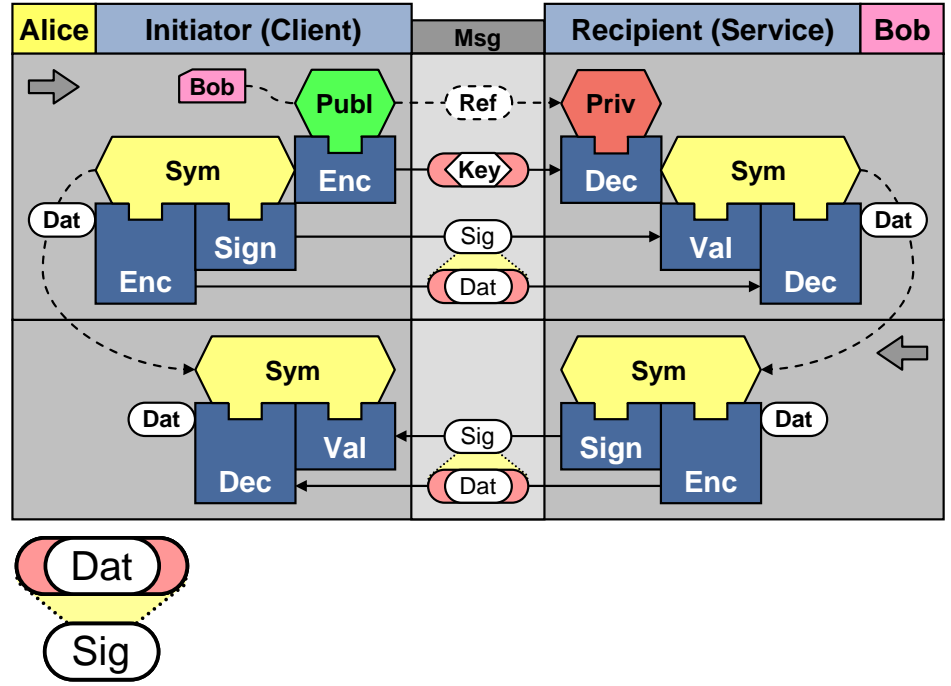


Choreografie-Option: Encrypt Before Sign (Symmetric Binding)

Default: Sign Before Encrypt



Option: Encrypt Before Sign

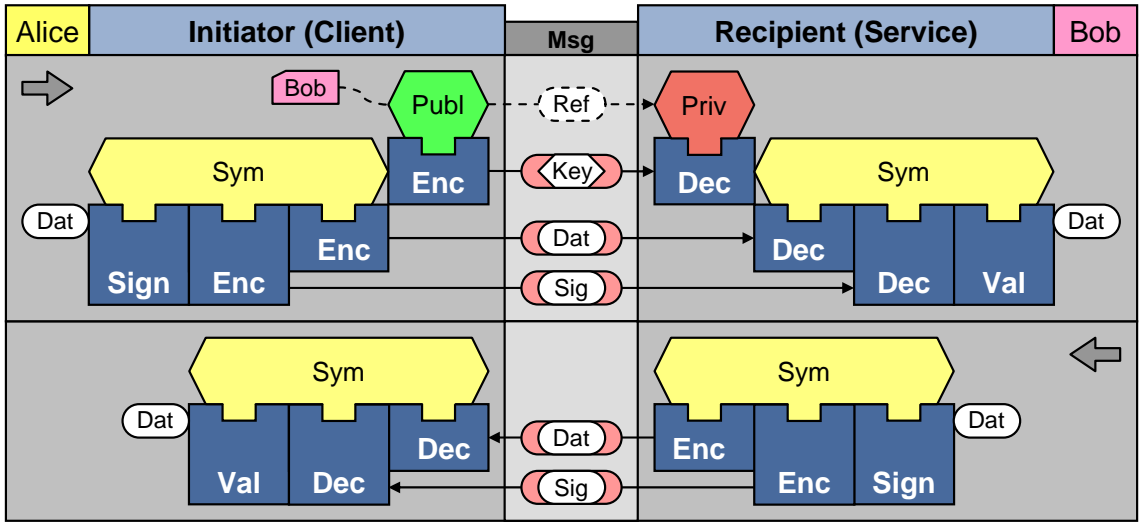


- ▶ Daten werden erst verschlüsselt und dann signiert
- ▶ Signatur kann überprüft werden, bevor die Daten entschlüsselt werden
- ▶ Gilt für Request- und Response-Nachrichten
- ▶ Auch anwendbar für Asymmetric Binding

Policy Assertions

- SymmetricBinding
 - ProtectionToken (P)
 - X509v3 (incl: Never)
 - **EncryptBeforeSigning**
- EncryptedParts, SignedParts
 - Body

Choreografie-Option: Encrypted Signatures (Symmetric Binding)



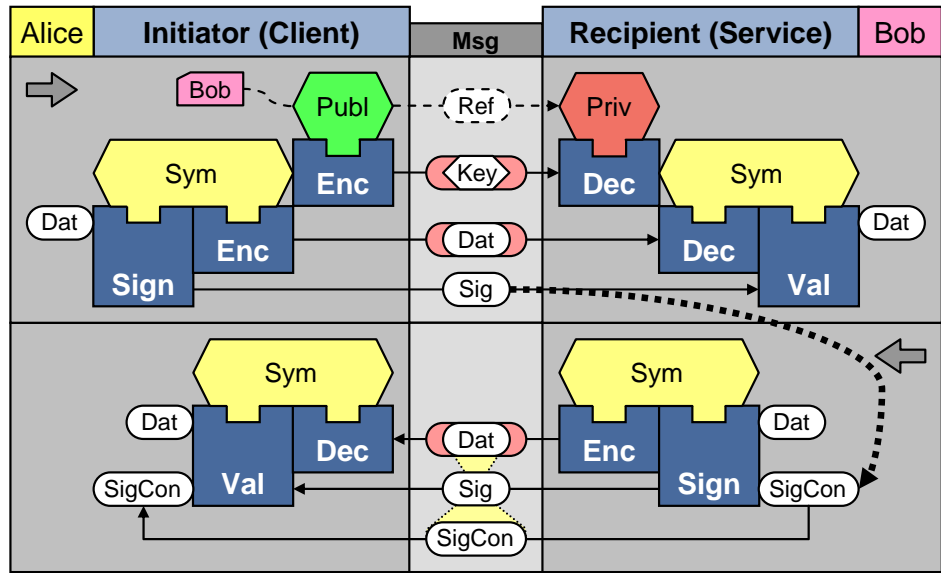
- ▶ Die Signatur wird ebenso wie die Daten verschlüsselt
- ▶ Achtung: In der Nachricht sind während des Transports keine Signaturen sichtbar
- ▶ Gilt für Request- und Response-Nachrichten
- ▶ Auch anwendbar für Asymmetric Binding
- ▶ z.B. kombinierbar mit Option „Encrypt-Before-Sign“

Policy Assertions

- SymmetricBinding
 - ProtectionToken (P)
 - X509v3 (incl: Never)
 - **EncryptSignature**
- EncryptedParts, SignedParts
 - Body

Choreografie-Option: Signature Confirmation (Symmetric Binding)

Symmetric Binding:

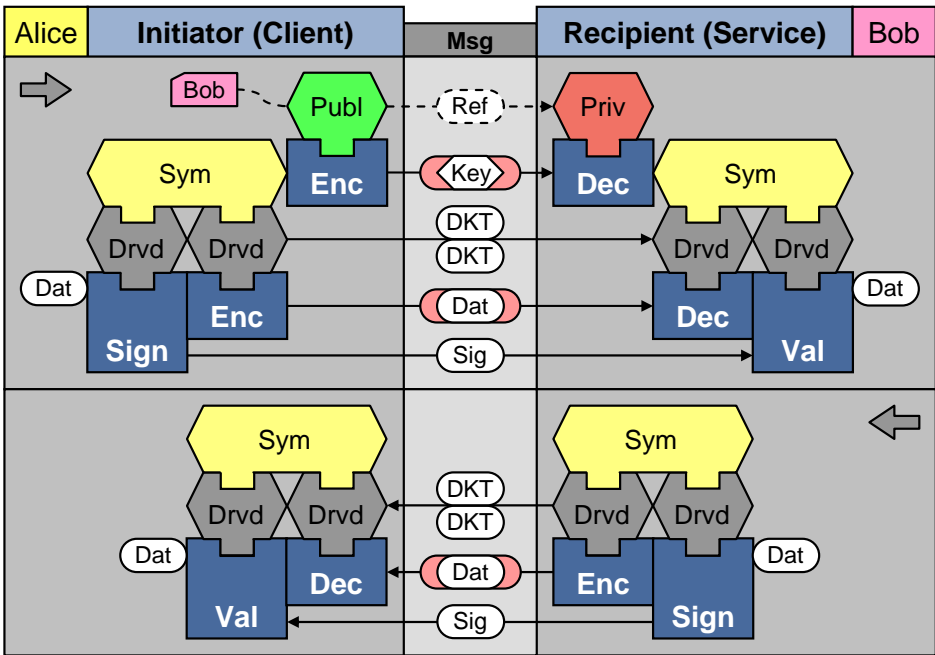


Policy Assertions
- SymmetricBinding
- ProtectionToken (P)
- X509v3 (incl: Never)
- SignatureConfirmation
- EncryptedParts, SignedParts
- Body

- ▶ Die empfangenen Signaturen müssen vom Recipient in der Response-Nachricht bestätigt werden
 - ▶ Der Initiator überprüft die Werte mit seinen gesendeten Werten
- ▶ Lösung für das Problem, das aufgrund der fehlenden Client-Authentifizierung nicht nachgewiesen werden kann, dass die Nachricht nicht manipuliert wurde (den Public-Key vom Bob kann jeder besitzen)
- ▶ Veränderungen nur in Response-Nachricht
- ▶ Auch anwendbar für Asymmetric Binding

Choreografie-Option: Derived Keys (Symmetric Binding)

Symmetric Binding:

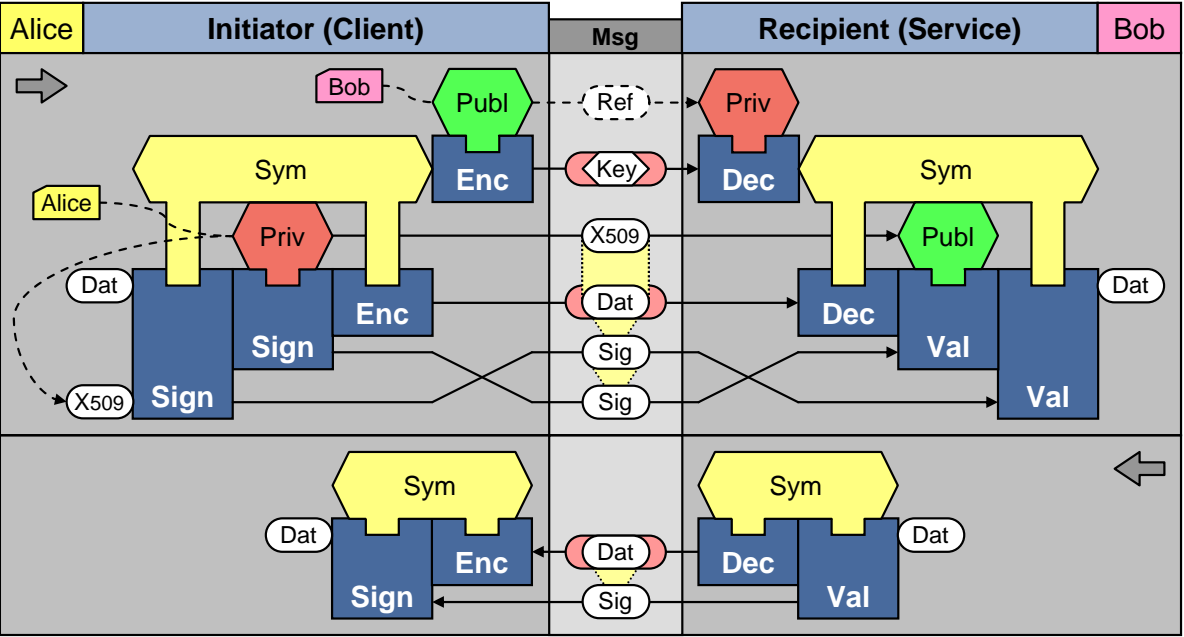


Policy Assertions

- SymmetricBinding
 - ProtectionToken (P)
 - X509v3 (incl: Never)
 - **DerivedKeys**
- EncryptedParts, SignedParts
- Body

- ▶ Jeder Krypto-Algorithmus verwendet einen eigenen Schlüsselwert, die aber von einem gemeinsamen Schlüssel abgeleitet werden
 - ▶ Vermeidung, dass von mehreren Datenblöcke mit dem gleichen Schlüssel kryptografisch behandelt sind
- ▶ Mechanismus stammt von WS-SecureConversation und kann allgemein verwendet werden
- ▶ Gilt für Request- und Response-Nachrichten
- ▶ Auch anwendbar für Asymmetric Binding

Choreografie-Option: Protect Tokens (Symmetric Binding)

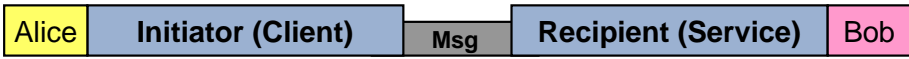


- ▶ Das Zertifikat von Alice („Token“) wird auch signiert, womit Manipulationen erkannt werden können
- ▶ i.d.R. nur innerhalb der Request-Nachricht
- ▶ Auch anwendbar für Asymmetric Binding

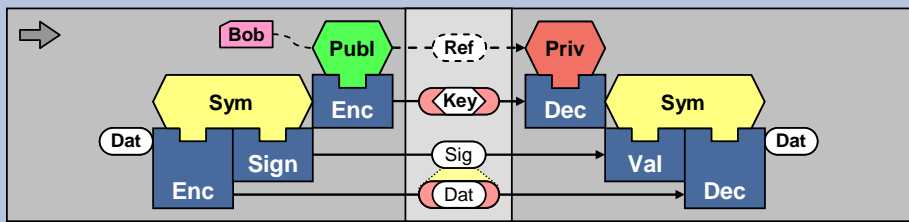
Policy Assertions

- SymmetricBinding
 - ProtectionToken (P)
 - X509v3 (incl: Never)
 - **ProtectTokens**
- EncryptedParts, SignedParts
 - Body

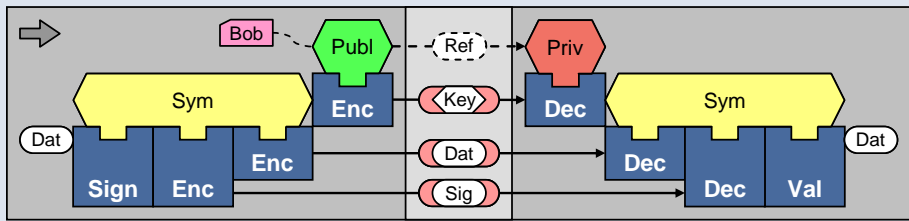
Übersicht: Choreografie-Optionen



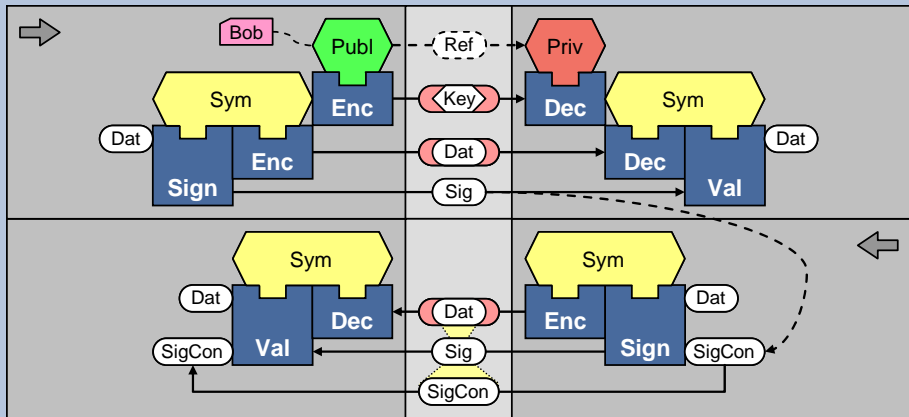
Encrypt Before Sign



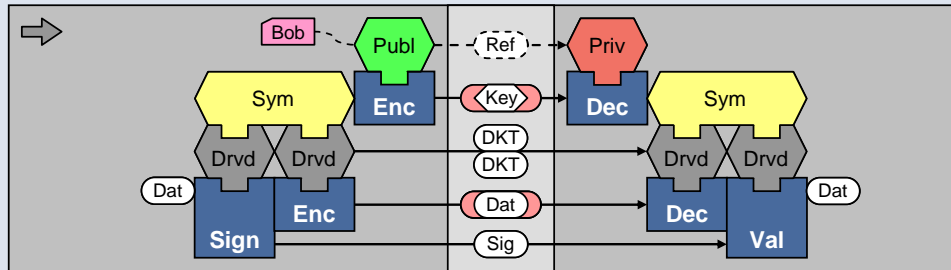
Encrypted Signature



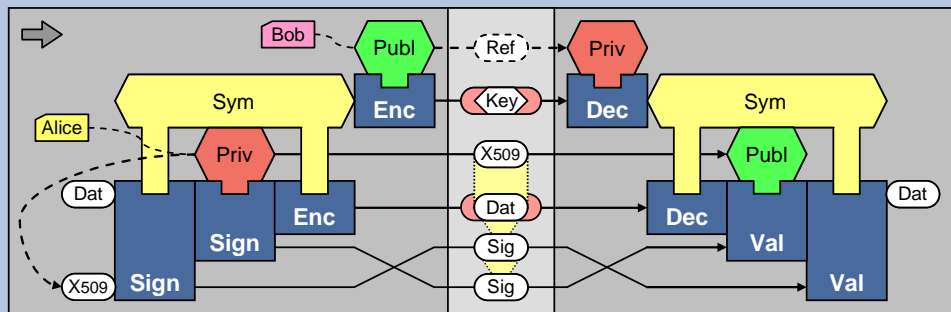
Signature Confirmation



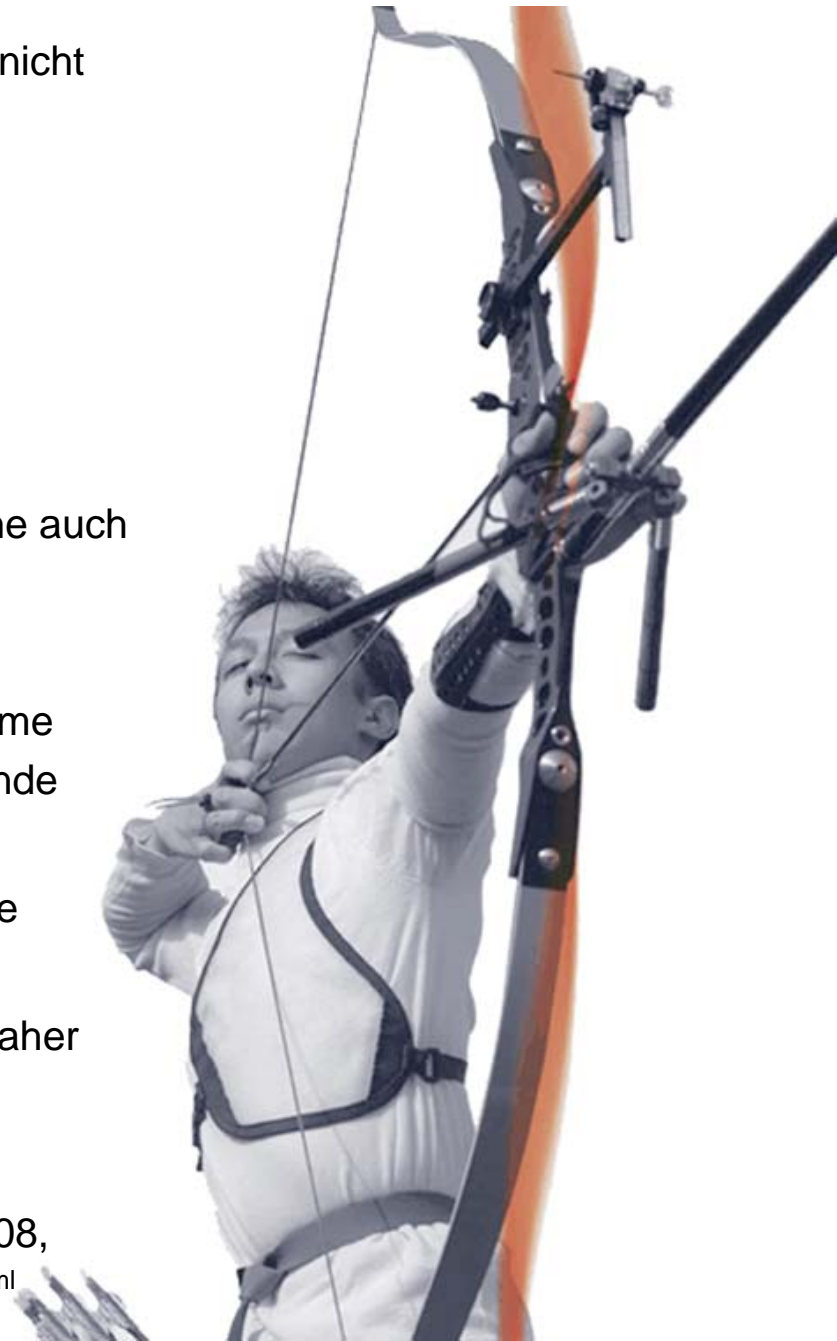
Derived Keys



Protect Tokens



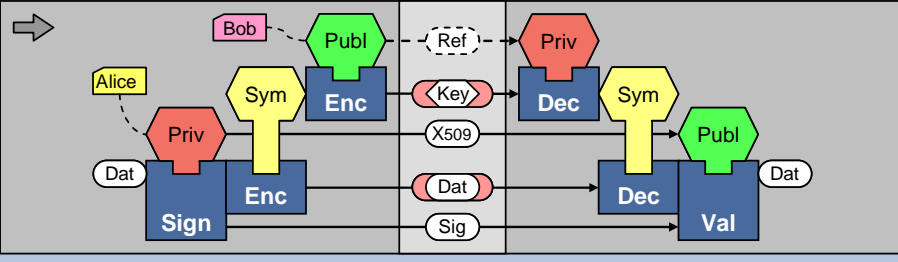
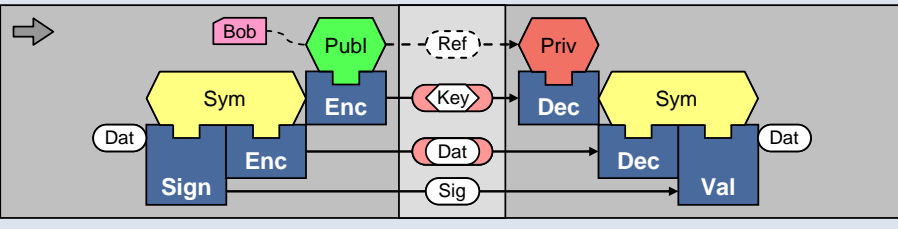
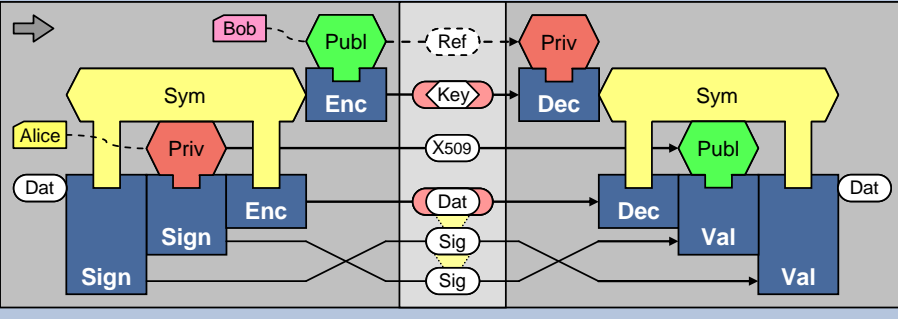
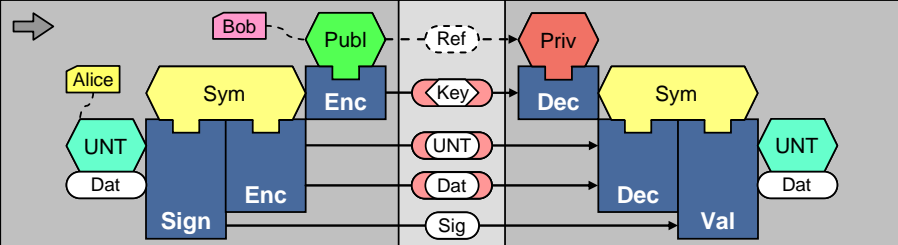
- ▶ Ohne Security kann eine SOA in geschäftskritischen Bereichen nicht betrieben werden
 - ▶ Security ist nicht nur Technologie sondern auch Business
- ▶ WS-Security (WSS) und WS-SecurityPolicy (WS-SP) sind entscheidende Basistechnologien
 - ▶ WS-Security liefert die Grundmechanismen
 - ▶ WS-SecurityPolicy liefert neben einer Beschreibungssprache auch entsprechende Choreografie-Patterns
- ▶ Interoperabilität auf Security-Ebene umfasst nicht nur die konforme Implementierung der Algorithmen sondern auch korrespondierende Choreografien
 - ▶ Die derzeitigen WS-Plattformen bieten sehr unterschiedliche Konfigurationsansätze
 - ▶ Plattform-übergreifende Security-Einstellungen benötigen daher ein hohes Maß an technischem Verständnis
- ▶ Artikel: „Choreografie der Webservice-Security“, LANLine 12/2008,
http://www.lanline.de/articles/choreografie_der_webservice-security:/2008012/31735603_ha_LL.html



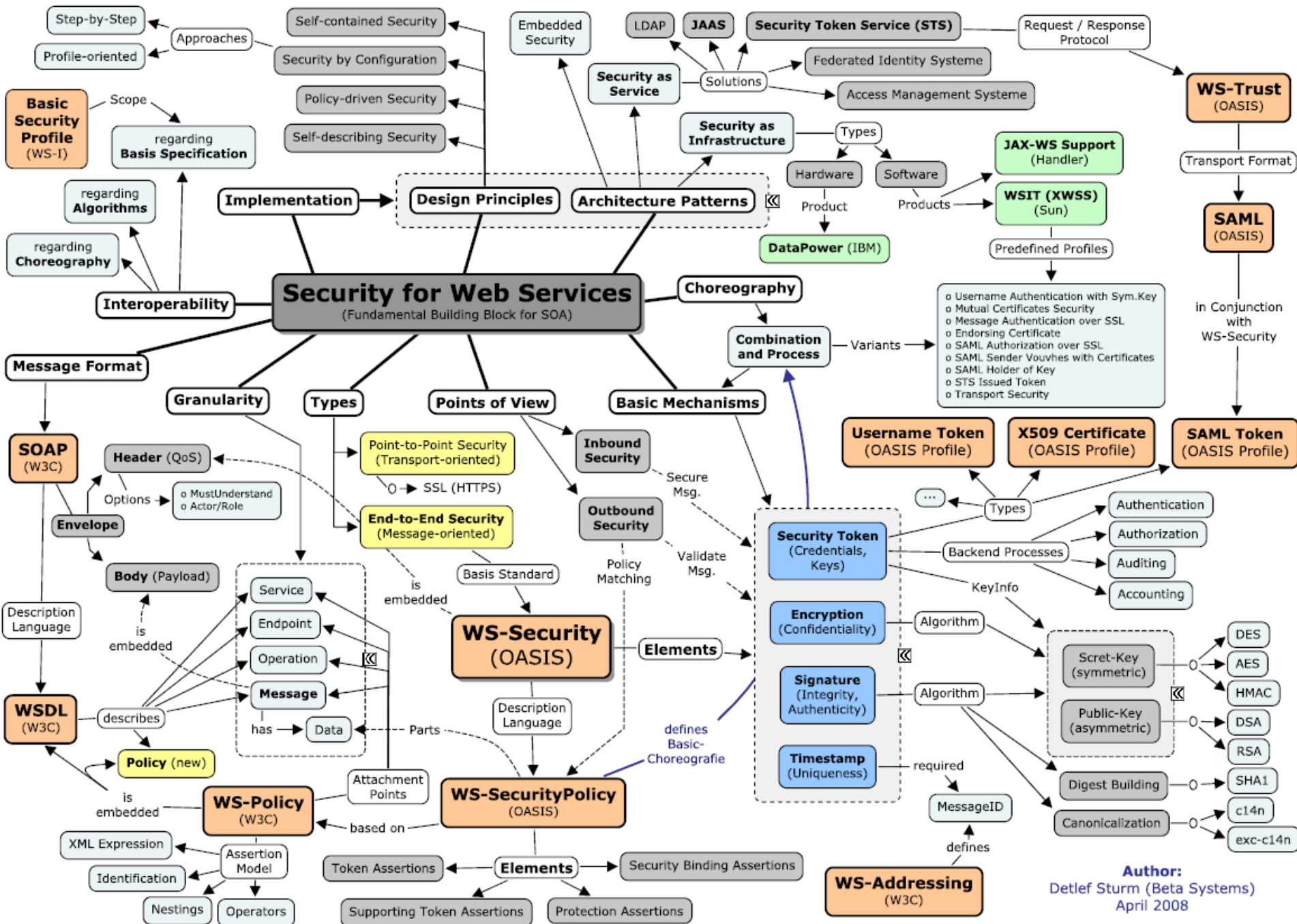
**Vielen Dank für die
Aufmerksamkeit**



Interoperabilität bzgl. der Security-Choreografie am Beispiel von WSIT und WCF

Choreografie	<div style="display: flex; justify-content: space-between; align-items: center;"> 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 & Signature		Endorsing Certificate	Mutual Certificate (SecurityVersion = WSSecurity11)
Symmetric Binding & Username Token		Username Authentication with Symmetric Key	Username for Certificate

Concept Map : Security for Web Services



Author:
Detlef Sturm (Beta Systems)
April 2008