Willkommen zur Vorlesung

Modellbasierte Softwaretechniken für sichere Systeme

im Sommersemester 2012

Prof. Dr. Jan Jürjens

TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

4. Modell-basierte Sicherheit mit UML

Anforderungen an die UML Extension für Security I

Stellt grundlegende Sicherheitsanforderungen wie secrecy, integrity, authenticity bereit.

Ermöglicht die Erwägung verschiedener Bedrohungsszenarios abhängig von den Fähigkeiten des Gegenspielers.

Ermöglicht das Einfügen wichtiger Sicherheitskonzepte (wie z.B. *tamper-resistant hardware*).

Ermöglicht das Aufnehmen von Sicherheitsmechanismen (wie z.B. access control).

Anforderungen an die UML Extension für Security II

Stellt grundlegende Sicherheit bereit (z.B. mit (a)symmetrischer Verschlüsselung).

Ermöglicht die Begutachtung der physikalischen Sicherheit.

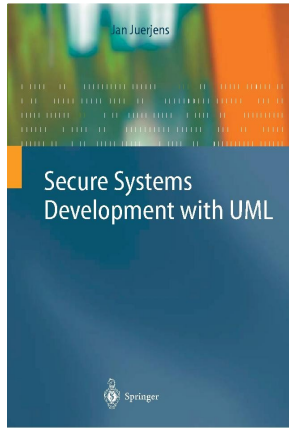
Ermöglicht das Einbeziehen des Sicherheitsmanagement (z.B. secure workflow).

Also: Fügt domain-spezifisches Sicherheitswissen hinzu (Java, smart cards, CORBA, ...).

Modell-basierte Sicherheit mit UMLsec

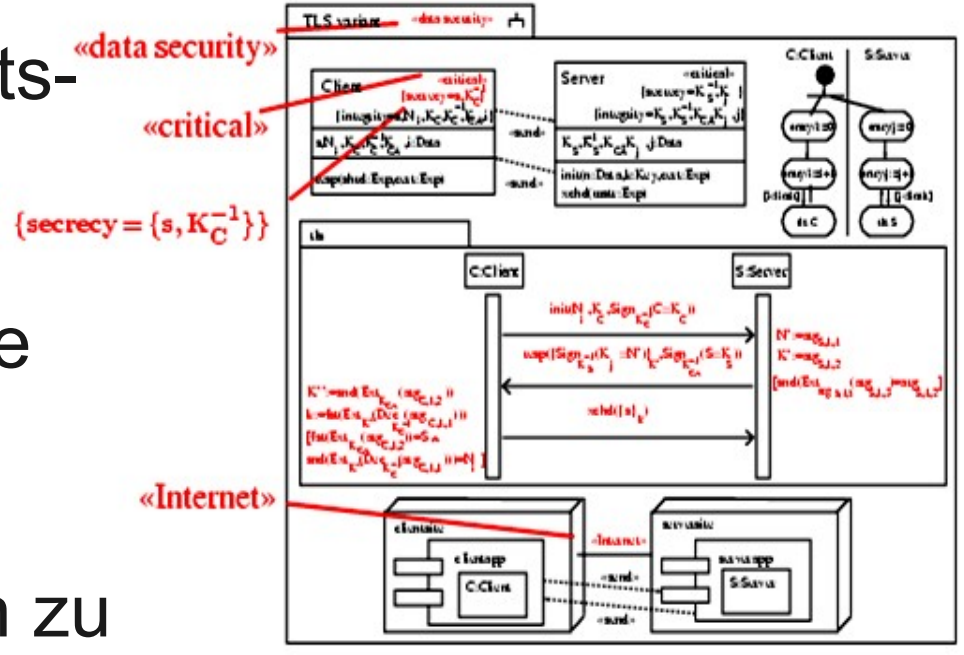
Erweiterung der Unified Modeling Language (UML) für sichere Systementwicklung.

- Evaluiert UML Modelle auf ihre Sicherheit.
- Fasst bestehende Regeln des gewissenhaften sicheren Entwickelns zusammen.
- Macht diese Entwicklern zugänglich die nicht auf sichere Systeme spezialisiert sind.
- Diskutiert Sicherheitsanforderungen aus frühen Designphasen im Systemkontext.
- Kann auch in der Zertifizierung benutzt werden.



UMLsec

Fügt wiederkehrende Sicherheitsanforderungen, feindliche Szenarios, und Sicherheitmechanismen als vordefinierte Marker hinzu.



Nutzt verknüpfte logische constraints um die Spezifikation zu verifizieren. Dabei kommen Model-Checker und ATPs, die auf formalen Semantiken basieren, zum Einsatz.

Stellt sicher das die die Modellspezifikation in UML die Sicherheitsanforderungen im Kontext des Dolev-Yao Angreifermodells durchsetzt.

Was deckt UMLsec ab ?

Sicherheitsanforderungen: <<secrecy>>,…

Bedrohungsszenarien: Vewende $\text{Threats}_{\text{adv}}(\text{ster})$.

Sicherheitskonzepte: Zum Beispiel <<smart card>>.

Sicherheitsmechanismen: Z.B. <<guarded access>>.

Grundlegende Sicherheit: eingebaute Verschlüsselung

Physikalische Sicherheit: Im Verteilungsdiagramm.

Sicherheitsmanagement: Im Aktivitätsdiagramm.

Technologie spezifisch: Java, CORBA Sicherheit.

UMLsec: Generelle Ideen

Aktivitätsdiagramm: Sicherer Kontrollfluss, Koordination

Klassendiagramm: Austausch von Daten
hält Sicherheitslevel bereit

Sequenzdiagramm: Sicherheitskritische Interaktion

Zustandsdiagramm: Sicherheit innerhalb eines Objekts

Verteilungsdiagramm: Physikalische Sicherheitsanforderungen

Package: Ganzheitliche Betrachtung der Sicherheit

UMLsec Profile (Auszug)

Stereotyp	Basisklasse	Tags	Bedingungen (Constraints)	Erläuterung
Internet	link			Internetverbindung
secure links	subsystem		dependency security matched by links	erzwingt sichere Kommunikationsverbind.
secrecy	dependency			gewährl. Geheimhaltung
secure dependency	subsystem		call, send respect data security	strukturelle Interaktion Datengeheimhaltung
no down-flow	subsystem	high	prevents down-flow	Informationsfluss
data security	subsystem		provides secrecy, integrity	grundlegende Daten-sicherheitsanforderung
fair exchange	package	start, stop	after start eventually reach stop	Erzwingt einen fairen Handel
guarded access	Subsystem		guarded objects acc. through guards.	Zugangskontrolle durch „guard objects“

UMLsec als integriertes formales Framework

Basiert auf Formalisierung der wichtigsten Sicherheitsanforderungen in einer integrierten Notation.

Erlaubt es, Anforderungen zu ordnen / kombinieren; ermöglicht Modularität / Komponierbarkeit, hierarchische Zerlegung, Verfeinerung, … :

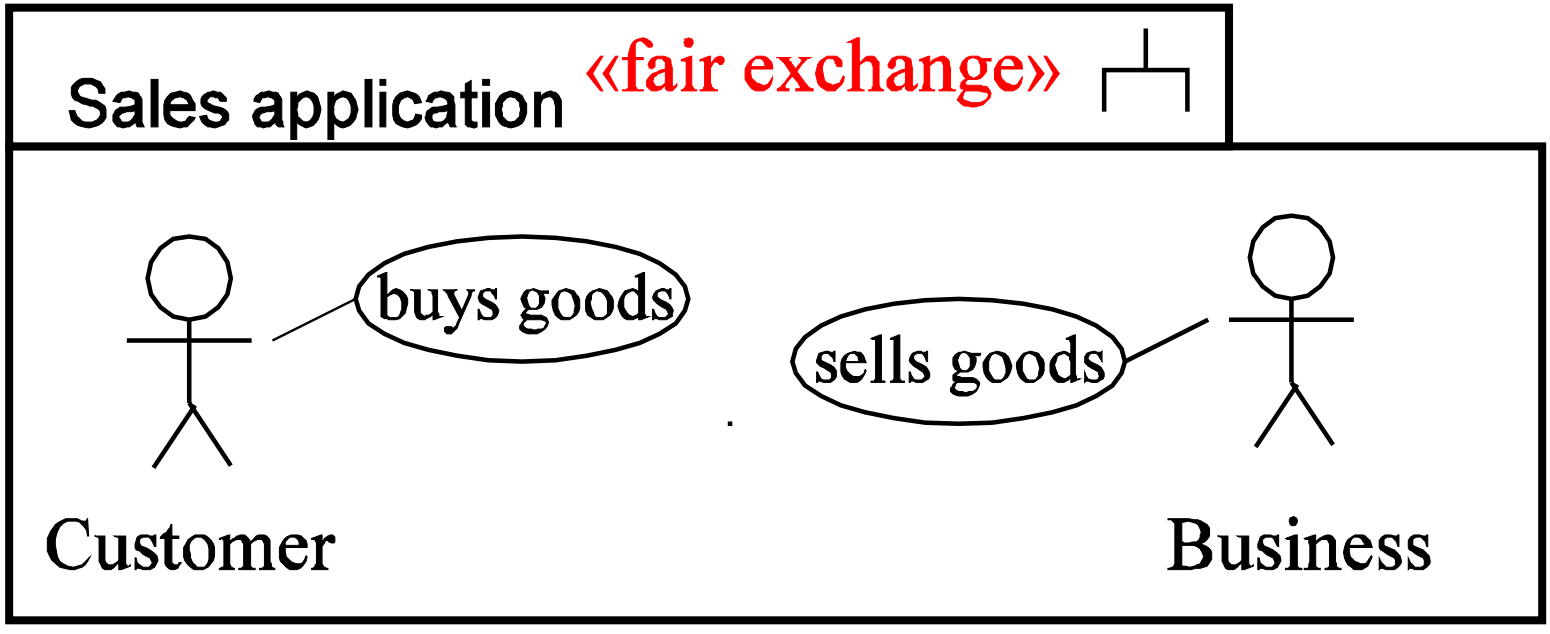
Zum Beispiel:

Wenn das System <<secure links>> und

das Subsystem <<data security>> genügt,

dann erfüllt das System auch <<data security>>.

Anforderungen mit Anwendungsfall-Diagrammen



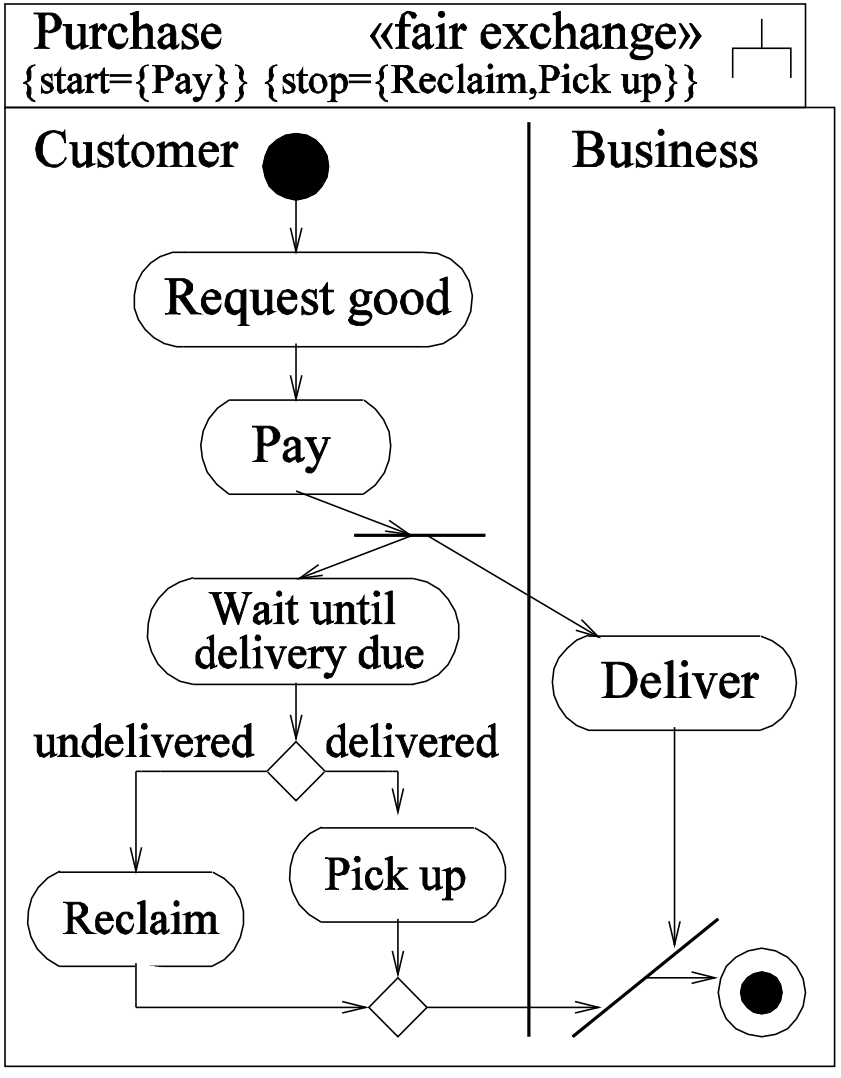
Erfasst Sicherheitsanforderungen in Anwendungsfalldiagrammen.

Das relevante Stereotyp muss ebenfalls im dazugehörigen Aktivitätsdiagramm auftauchen.

Fairer Handel

Ein Kunde kauft Ware bei einem Händler.

Wie kann man einen fairen Handel erzwingen ?
Nach der Bezahlung muss der Kunde bis zur Lieferfrist warten und kann dann die Zahlung zurückziehen.



<<fair exchange>>

Sichert generisch die Bedingungen für fairen Handel.

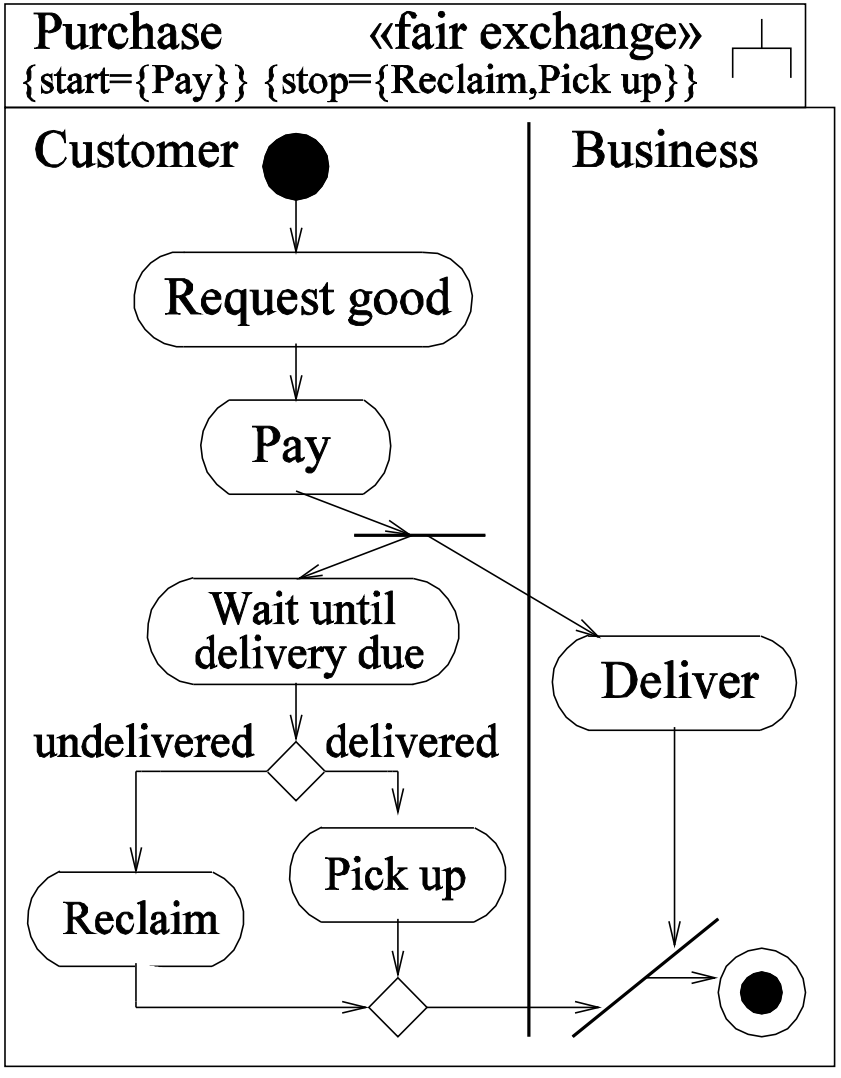
Bedingung: Wenn in einem Aktivitätsdiagramm ein {start} Punkt erreicht wurde, wird letzten Endes immer ein {stop} Punkt erreicht.

(Das kann nicht für ein System sichergestellt werden, das durch einen Angreifer komplett lahm gelegt werden kann.)

Beispiel

<<fair exchange>>

Erfüllt, wenn kein Angreifer das System stoppen kann: Nach der Zahlung kann der Kunde entweder die angelieferte Ware in Empfang nehmen, oder hat die Möglichkeit, die Zahlung zu stornieren.



<<Internet>>, <<encrypted>>, …

Arten von Kommunikationsverbindungen (Links) oder Systemknoten (Nodes).

Für Angreiferart A, Stereotyp s, gibt es eine Menge

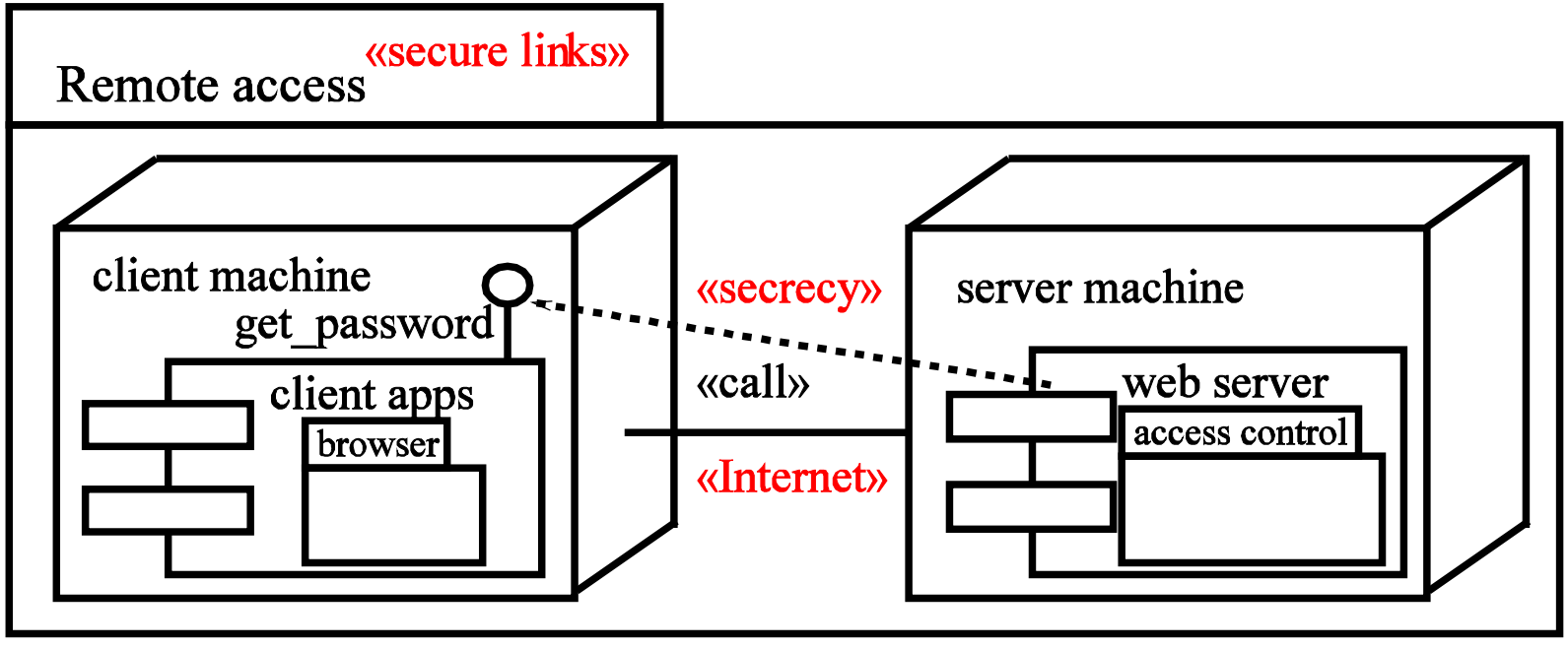
$Threats_A(s) \in \{delete, read, insert, access\}$

von Aktionen, die der Angreifer ausführen kann.

Standardangreifer:

Stereotype	$Threats_{default}()$
Internet	{delete, read, insert}
encrypted	{delete}
LAN	∅
smart card	∅

Frage



Architektur sicher gegen Standardangreifer ?

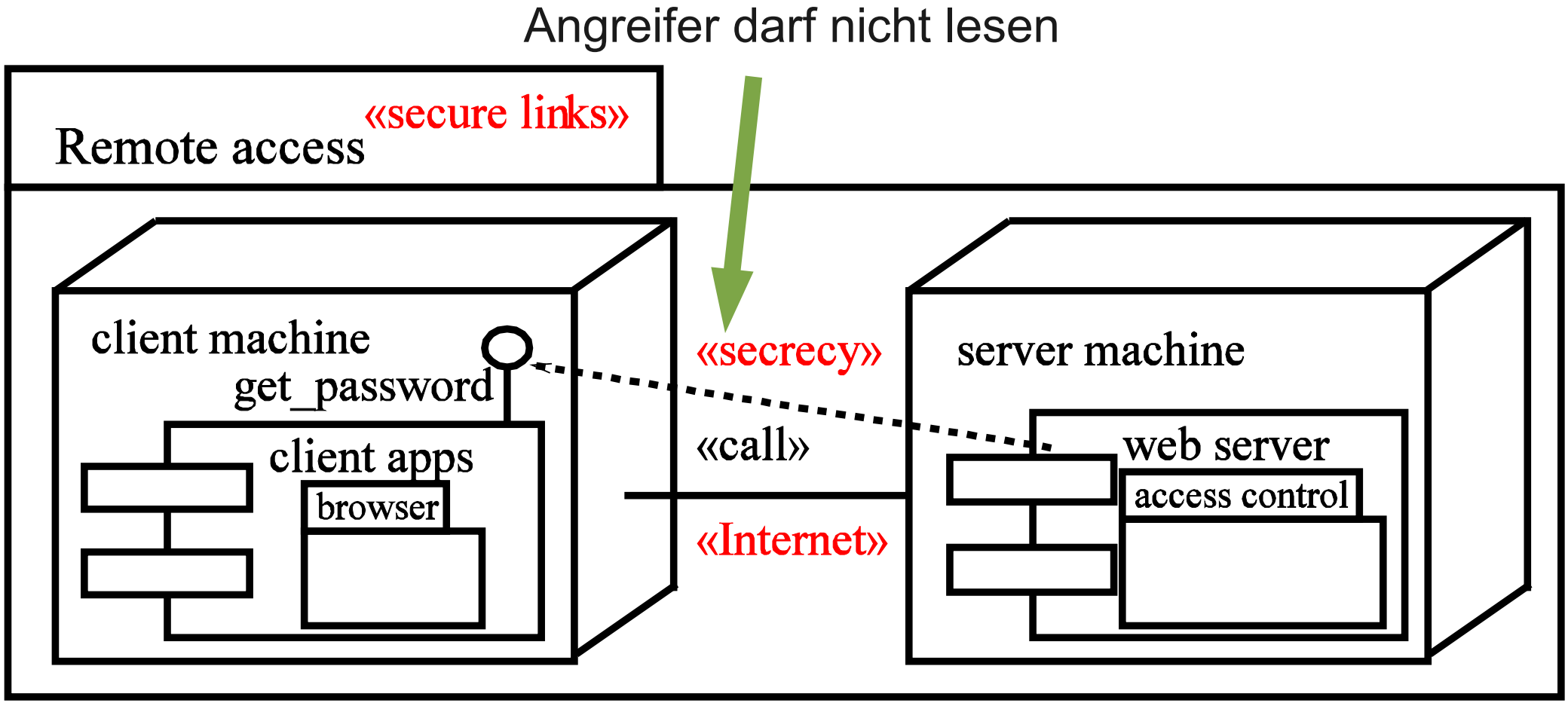
<<secure links>>

Stellt sicher, dass die physikalische Ebene die Sicherheitsbestimmungen bei der Kommunikation gegen einen gegebenen Angreifer des Typs *A* einhält.

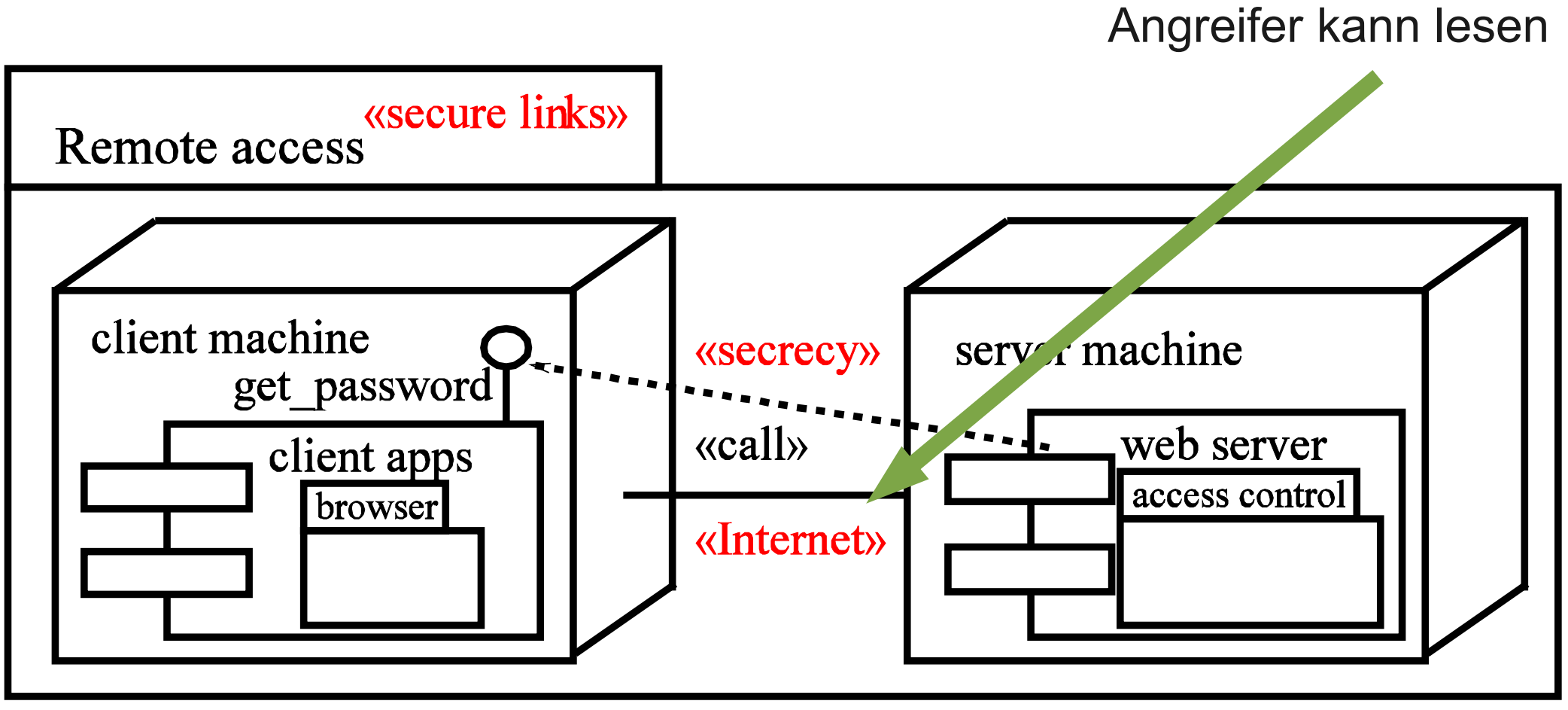
Bedingung: Für jede Abhängigkeit *d* mit Stereotyp $s \in$ {<<secrecy>>, <<integrity>>} zwischen Komponenten auf Knoten $n \neq m$ verläuft die Kommunikation über eine Kommunikationsverbindung *l* zwischen *n* und *m* mit Stereotyp *t* sodass:

- Wenn *s* = <<secrecy>>: habe read $\notin$ $Threats_A$ *(t)*.
- Wenn *s* = <<integrity>>: habe insert $\notin$ $Threats_A$ *(t)*.

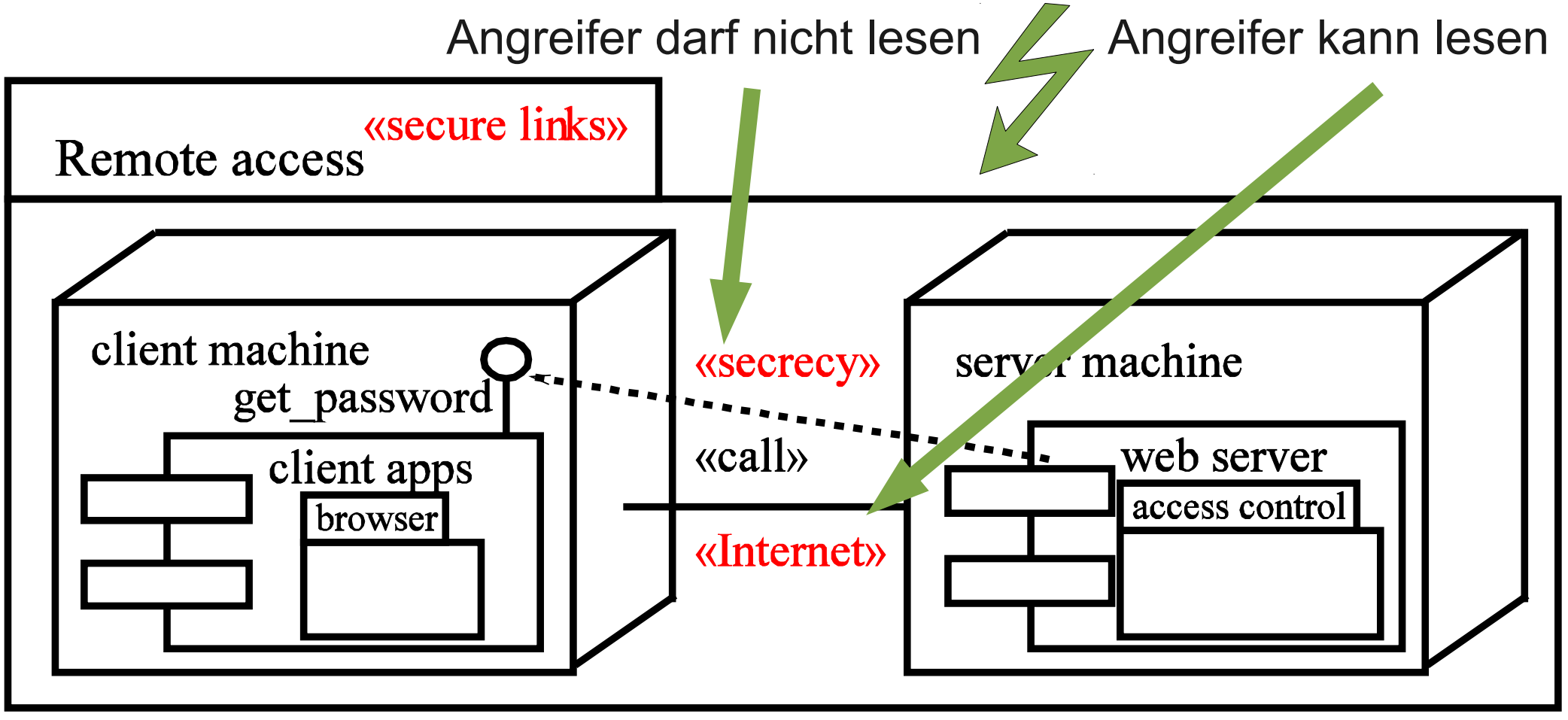
Beispiel <<secure links>>



Beispiel <<secure links>>



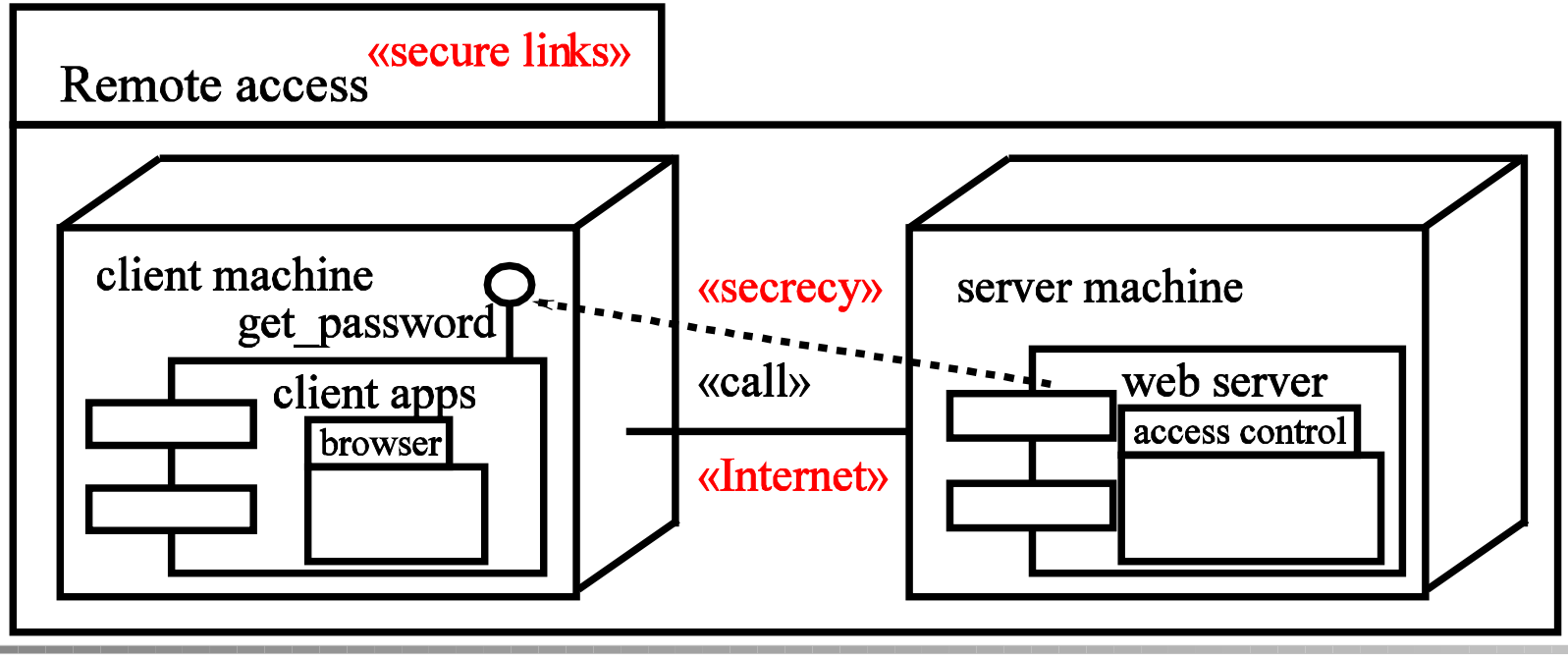
Beispiel <<secure links>>



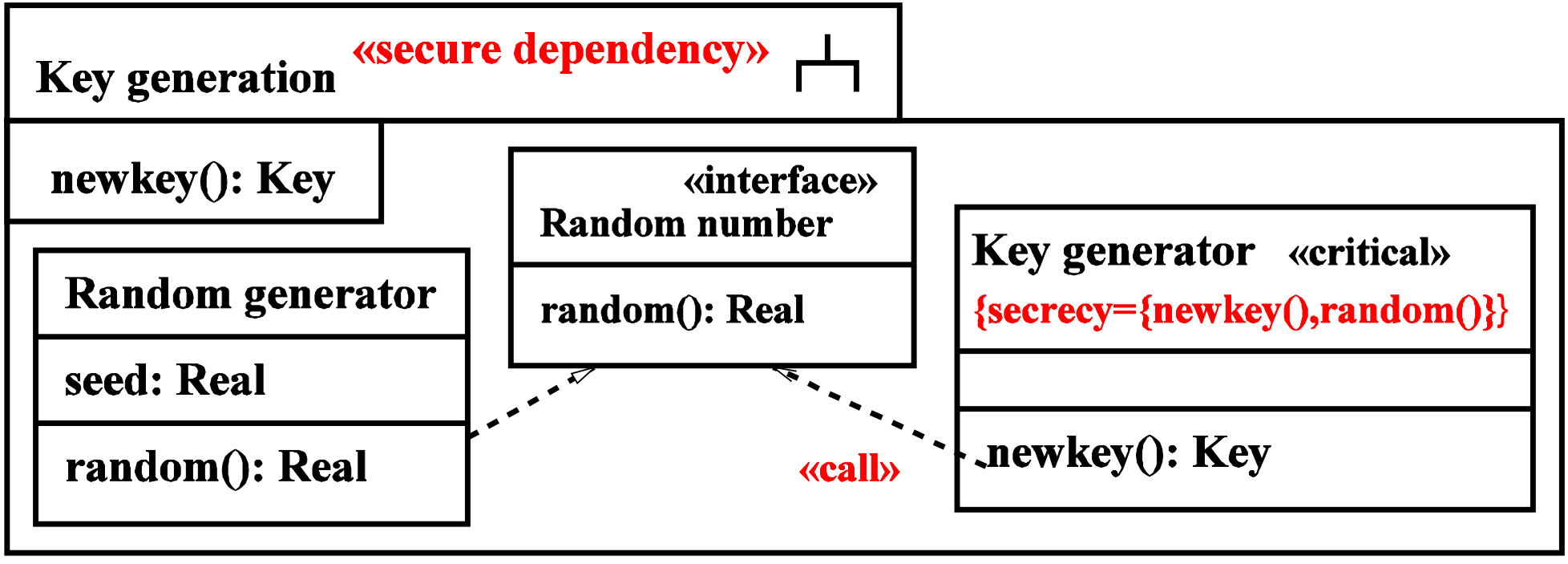
Beispiel <<secure links>>

Standard-Angreifertyp gegeben, Bedingung für Stereotyp <<secure links>> verletzt:

In Bezug auf das $Threats_{default}$ (Internet) Szenario, bietet <<Internet>> keine Sicherheit gegen den Standard Angreifer.



Frage



Datenverteilung sicher ?

<<Secure dependency>>

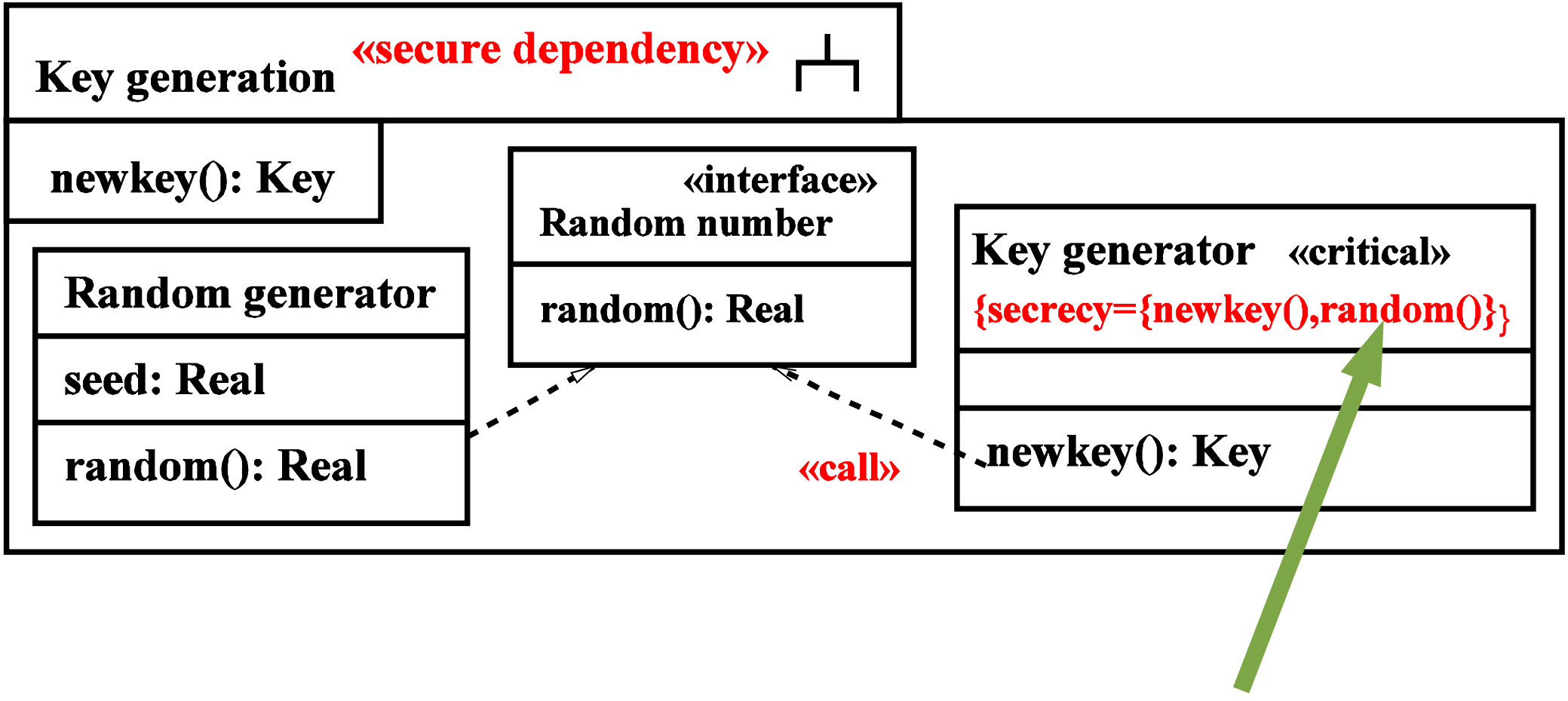
Stellt sicher, dass <<call>>- und <<send>>-Abhängigkeiten zwischen Komponenten die Sicherheitsanforderungen für versendete Daten respektieren. Das geschieht mit den Tags {secrecy}, {integrity}.

Bedingung: Für <<call>>- oder <<send>>-Abhängigkeiten von *C* nach *D* (und genauso für {integrity}):

Nachricht in *D* ist markiert {secrecy} in *C* genau dann wenn auch in *D*.

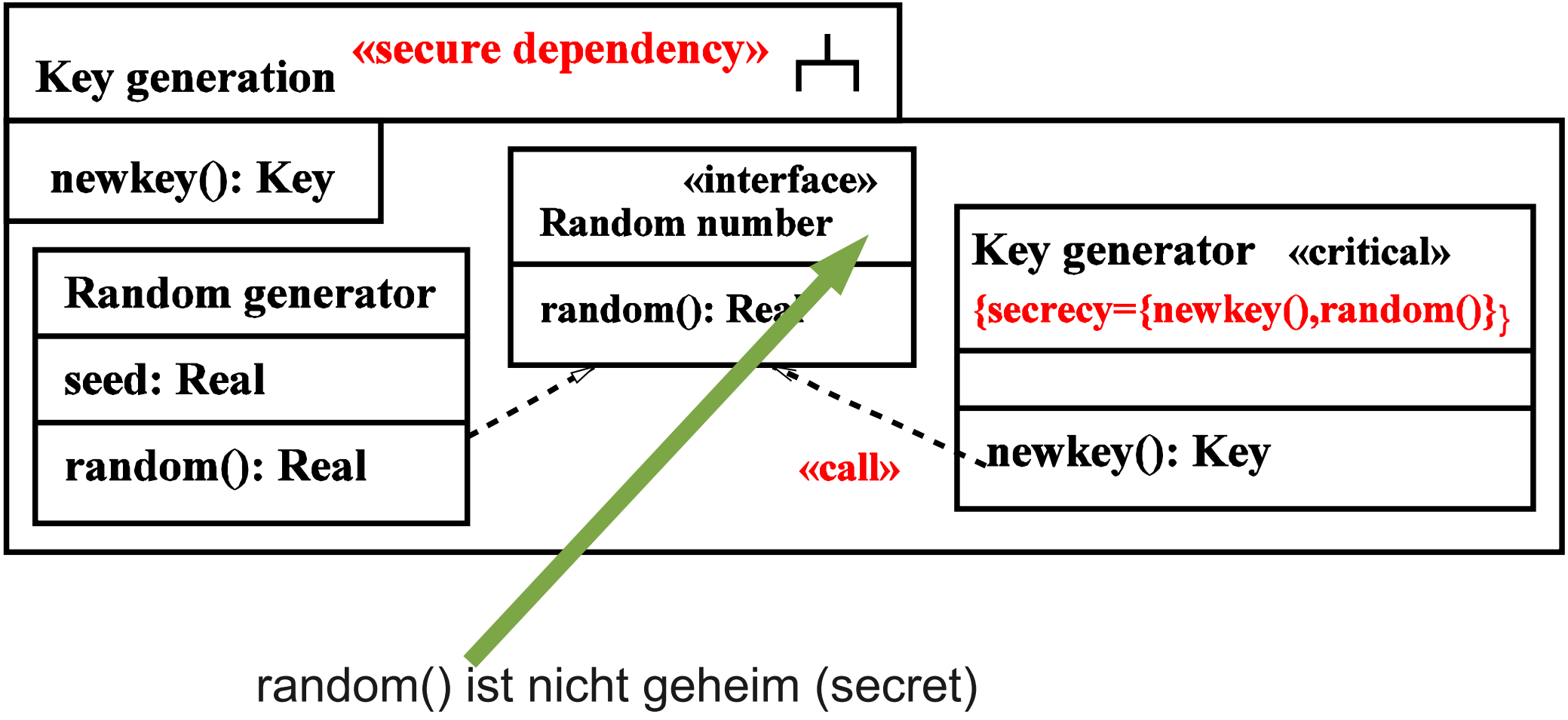
Wenn Nachricht in *D* als {secrecy} in *C* markiert ist, dann ist Abhängigkeit mit <<secrecy>> markiert.

Beispiel <<secure dependency>>

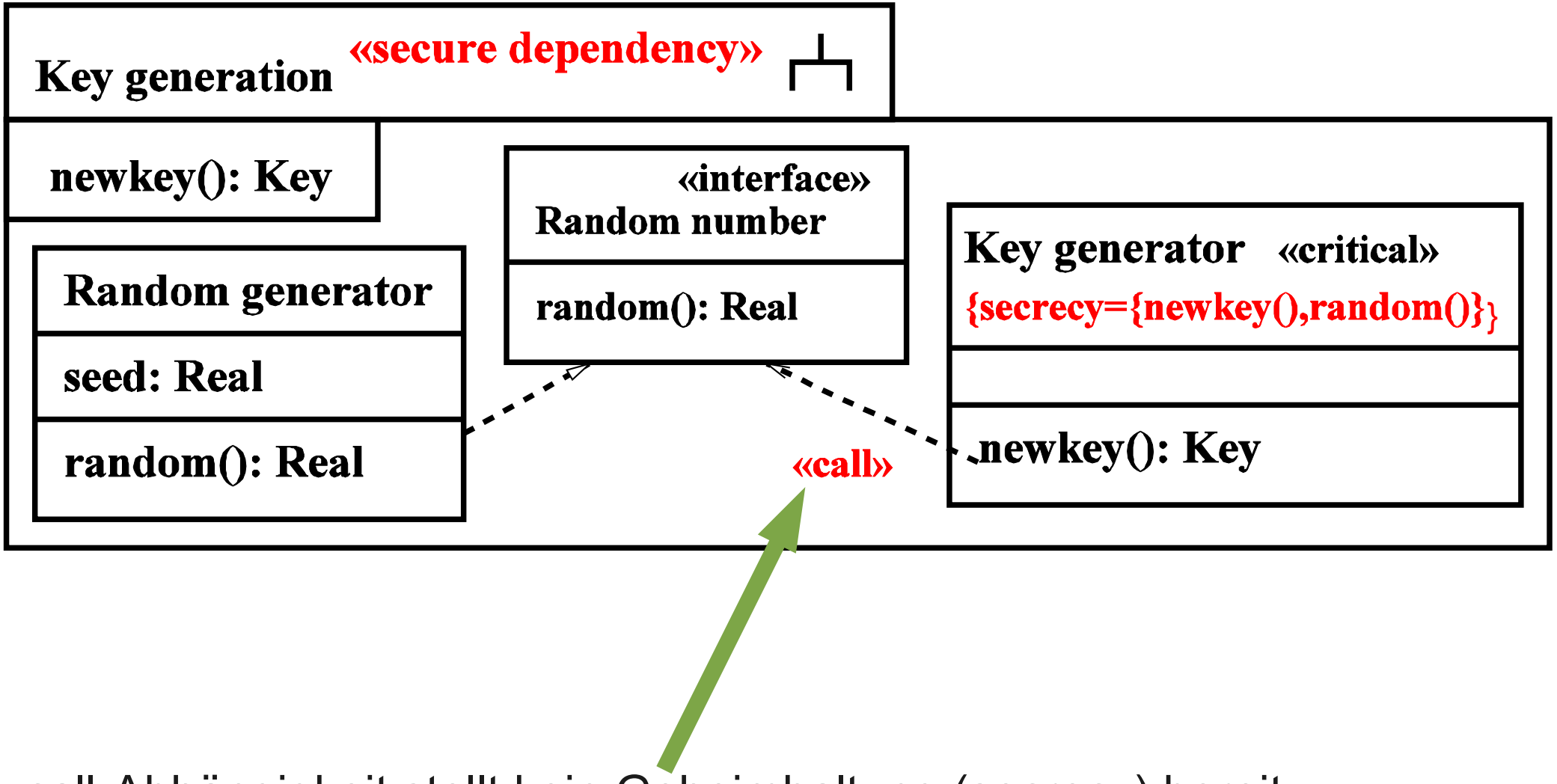


random() muss geheim (secret) sein

Beispiel <<secure dependency>>

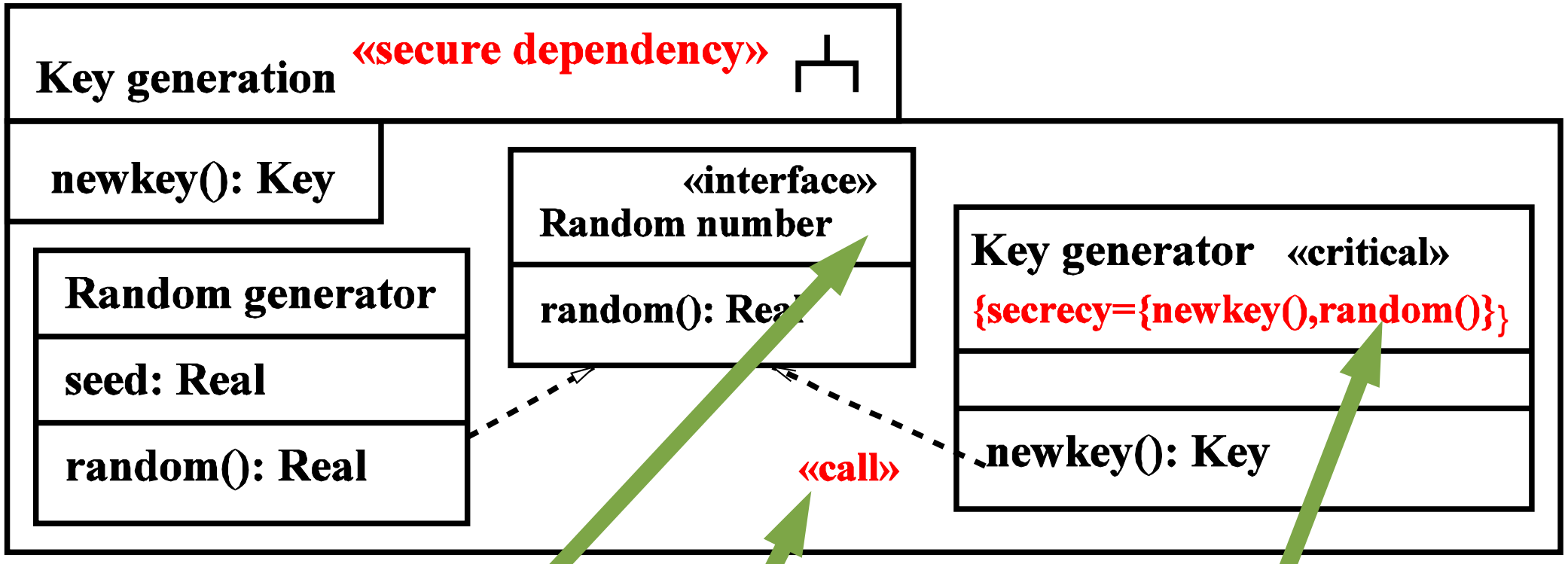


Beispiel <<secure dependency>>



call-Abhängigkeit stellt kein Geheimhaltung (secrecy) bereit

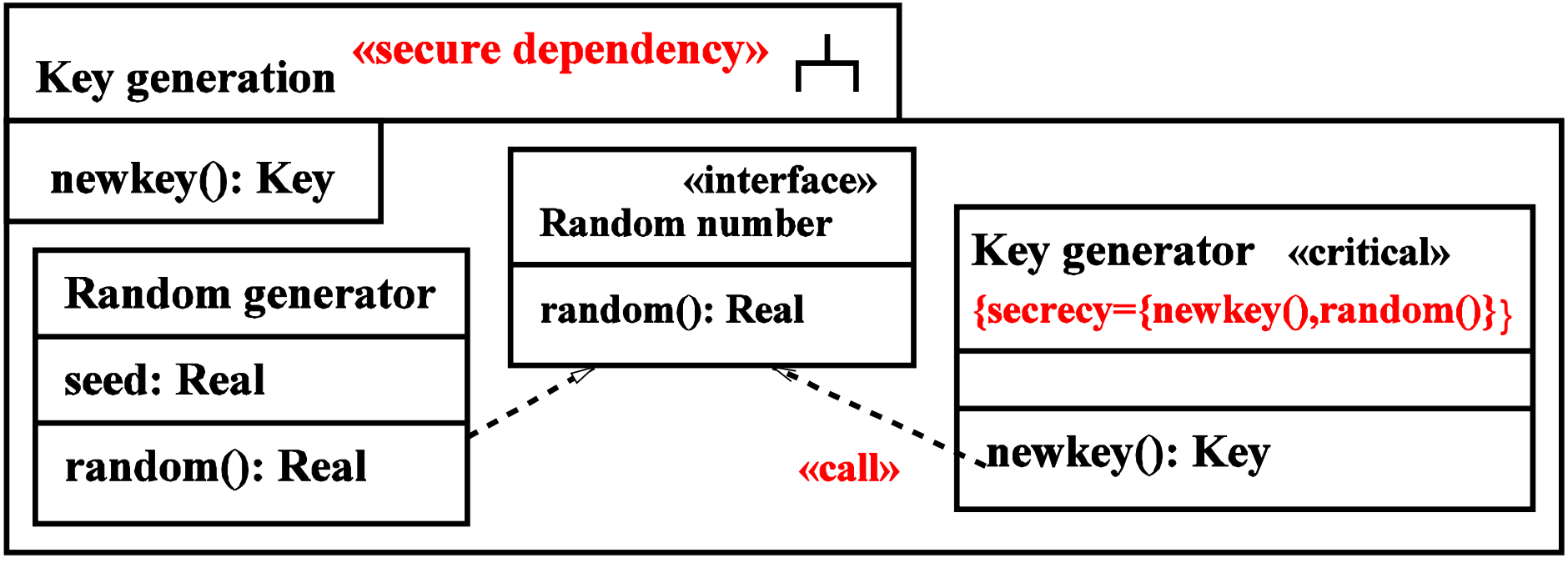
Beispiel <<secure dependency>>



random() nicht geheim (secret)
+
call-Abhängigkeit stellt
keine Geheimhaltung (secrecy) bereit

random() muss geheim (secret) sein

Beispiel <<secure dependency>>



Verletzt <<secure dependency>>: Random und die <<call>>-Abhängigkeit bieten keine Sicherheit von random() zum Key generator.

Zusammenfassung

- Idee hinter UMLsec
- Übersicht UMLsec Stereotypen
- Fair exchange
- Secure links
- Secure dependency