

Mobile Devices Payment Protocol Version 1.0

Stand: 07.11.2005

Abkürzungsverzeichnis

API	Application Programming Interface
CP	Content Provider
GPRS	Global Packet Radio Standard
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identificator
IMEI	International Mobile Equipment Identifier
IMSI	International Mobile Subscriber Identifier
IP	Internet Protokoll
ISO	International Stardardization Organisation
MDPP	Mobile Devices Payment Protocol
MSISDN	Mobile Number Integrated Services Digital Number
MU	Mobile User
NO	Network Operator
RADIUS	Remote Authentication Dial In User Service
SIM	Subscriber Identification Module
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
TLS	Transport Layer Security
UMTS	Universal Mobile Telecommunication Standard
URL	Unified Resource Locator
PDA	Personell digital assistant

Inhaltsverzeichnis

Abkürzungsverzeichnis.....	2
Inhaltsverzeichnis	3
Abbildungsverzeichnis	3
Copyright	3
Terminologie.....	4
Technische Beschreibung	6
Service Autorisierung	8
Mobile User Authentisierung.....	10
Content Autorisierungsprozess.....	11
„trusted Token“ Validierung durch den Content Provider.....	14
Abonnement Verfahren mit MDPP	15
RSA Schlüsselaustausch.....	16
Beispiel: Erzeugen einer Signatur	17
Beispiel: Verifizieren einer Signatur	17
Abrechnungsprinzipien	17
Sicherheitsbetrachtungen	19
Man-in-the-Middle	19
Replay Angriffe	19
Reproduktion von Tokens	19
Identifikation des Benutzers	19
Betrug durch Parameter Manipulation.....	19
Betrug durch falsche Token Parameter.....	19
Betrug durch falsche Abrechnungsdaten eines Content Providers	19
Generelle Betrachtung	20
Referenz.....	20
Annex 1 Parameter Definition.....	21
Annex 2 Implementation Examples	22

Abbildungsverzeichnis

Abbildung 1: Übersicht.....	7
Abbildung 2: Kommunikationsbeziehungen.....	7
Abbildung 3: Diagramm MU Authorisierung.....	10
Abbildung 4: Content Autorisierung	12
Abbildung 5: „trusted Token“ Validierung.....	14

Copyright

Dieses Dokument sowie sein Inhalt unterliegen dem Copyright der Cyber-Dynamix GmbH, Nürnberg © 2005. Jegliche Nutzungen, Veröffentlichungen auf elektronischer, schriftlicher oder anderer Weise sowie Vervielfältigungen bzw. Kopien sind ohne ausdrücklicher schriftlicher Genehmigung untersagt. Der Herausgeber und der Autor behalten sich alle Rechte vor.

Terminologie

Content Provider

Anbieter von Internet Inhalten, z.B.: Nachrichten, Klingeltöne, Videos, Animationen, Spielen, etc.

Mobile User

Nutzer eines mobilen Endgerätes [Device] mit Zugang zu und innerhalb von einem Mobilfunknetz eines Mobilfunknetzbetreibers [Network Operator] und Konsument von Mobilfunk- und Internet-basierten Angeboten auf Basis von vorausbezahlten Guthaben (PrePaid) oder Verträgen (PostPaid) mit zyklischer Rechnungsstellung.

Network Operator

Betreiber eines Mobilfunknetzes und der dazugehörigen Infrastruktur zur Authentisierung, Autorisierung und Abrechnung von Kunden und Mobilfunkdienstleistungen sowie zur Bereitstellung von Zusatzleistungen, z.B. Geopositionsdaten, Messaging Dienstleistungen, Short- und MultiMedia Diensten, etc.

Content

Kostenfreie und –pflichtige Informationen oder Dienstleistungen eines Content Providers.

Token

Definiertes und zusammengehöriges Set an Informationen.

Signatur

Digitale nicht reproduzierbare Kennzeichnung eines Tokens unter Einsatz von kryptographischen „public key“ Verfahren.

MSISDN

Eindeutige Mobilfunknummer zur Identifikation eines MU innerhalb eines Mobilfunknetzes.

Billing

Berechnung von konsumierten Dienstleistungen gegenüber eines Mobile Users auf Basis von Ereignis- oder Nutzungsnachweisen.

RSA Kryptverfahren

Asymmetrisches Verschlüsselungsverfahren der Mathematiker Ronald L. Rivest, Adi Shamir und Leonard Adleman.

Autorisierung

Ermittlung der Berechtigung

Authentisierung

Ermittlung der Echtheit

Autorisierungsobjekt

Server basierte Anwendung des NO zur Autorisierung, persistenten Speicherung, Guthabenprüfung, Abrechnung und Beantwortung von MU Autorisierungsanfragen.

Device

Mobiles Endgerät mit SIM Karte (z.B. Mobiltelefon) oder Endgerät (z.B. PDA, Notebook, ...) mit Zugang zu einem Mobiletelefon(z.B. über Infrarot-, Kabel- oder Bluetooth Schnittstelle)

Technische Beschreibung

Das MDP-Protokoll wurde zur Authentisierung von Kunden in mobilen Datennetzen sowie zur anonymen Nutzung und Abrechnung von Internet-Inhalten entwickelt. Es beruht auf dem Prinzip der dezentralen Informationsbereitstellung, zentralen Benutzerauthentisierung und signierten Nachrichten [Token], welche zwischen einem Content Anbieter [Content Provider], einem Konsumenten [Mobile User] und einem Mobilfunkbetreiber [Network Operator] ausgetauscht werden.

Die Kommunikation zwischen einem CP und einem MU erfolgt über HTTP (RFC2616), wobei die Informationsübermittlung direkt als Parameter in HTML erfolgt. Dieser Mechanismus ist im folgenden Text detailliert beschrieben.

Die Kommunikation zwischen einem MU und einem NO erfolgt entweder über HTTP (siehe oben) oder als SOAP basierter Webservice Aufruf, welcher entweder direkt aus einer HTML-Seite, über eine API oder eine Applikation erfolgen kann.

Weiterhin kommen „trusted Token“ zur Autorisierung von Kundenanfrage zur Anwendung, welche durch die authentifizierende Instanz (NO) digital signiert werden und dem CP somit die Möglichkeit bieten, eine sichere Aussage über die Authentizität einer Kundenanfrage zu treffen und als Grundlage für die Abrechnung dienen.

Das MDP-Protokoll erfordert prinzipiell keine weitere Online Kommunikation zwischen Content Providern und Network Operatoren. Jedoch bleibt es den Kommunikationspartner unbelassen, übermittelte „trusted Token“, z.B. ab einem definierbaren Betrag online zu verifizieren. Hierfür stehen geeignete Protokolle, z.B. SOAP zur Verfügung, auf die in diesem Dokument nicht weiter eingegangen wird. Ebenso werden keine zusätzlichen Kommunikationskomponenten, wie z.B. gateways oder access proxies benötigt.

Die nachfolgende Graphik verdeutlicht die Kommunikationsbeziehungen:

Ein MU kommuniziert mit einem Content Provider über das Internet um freie und / oder kostenpflichtige Dienste zu nutzen. Hierbei wird der Zugang zum Internet über das Mobilfunknetz eines Network Operators gewährleistet.

Zur Autorisierung von kostenpflichtigen Angeboten kommuniziert der Mobile User mit dem Network Operator.

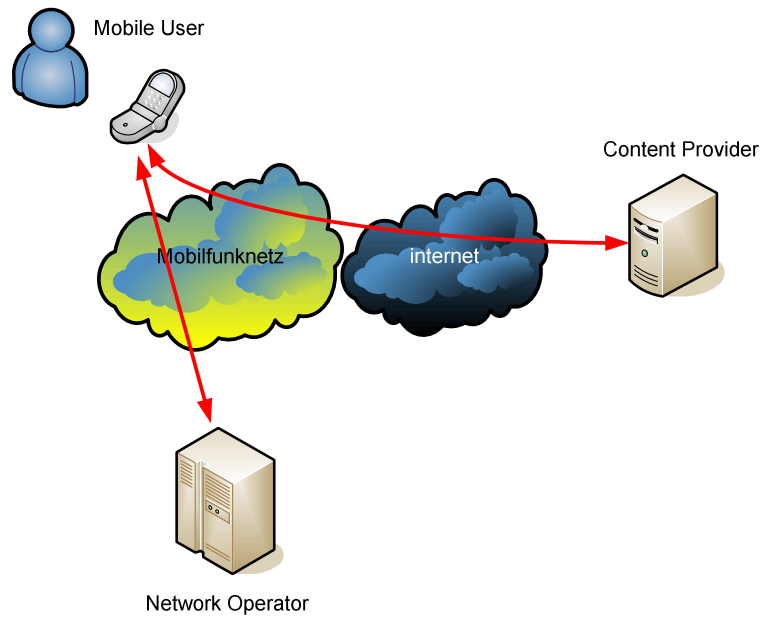


Abbildung 1: Übersicht

Das folgende Flussdiagramm zeigt die Interaktionen zwischen MU, CP und NO. Die Details der Nachrichten sind in den folgenden Kapiteln beschrieben.

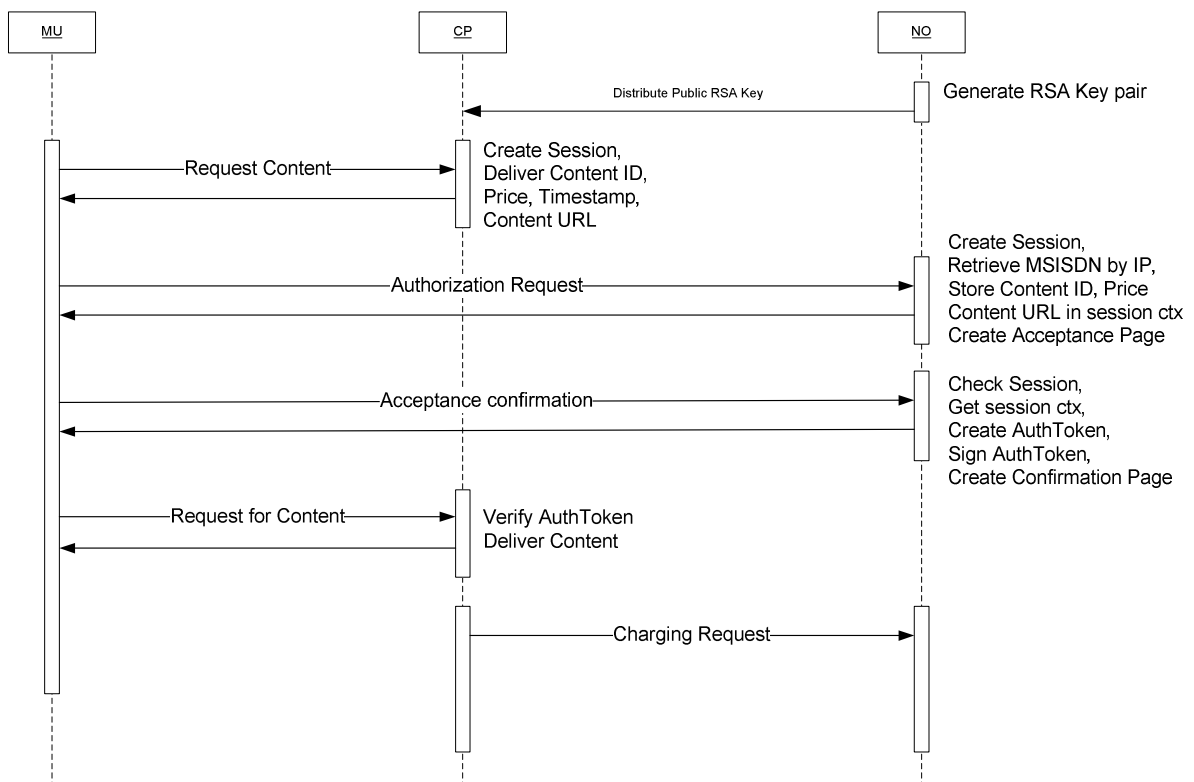


Abbildung 2: Kommunikationsbeziehungen

Service Autorisierung

Sobald der MU eine kostenpflichtige Dienstleistung eines Content Providers nutzen möchte, wird der CP diese dem Kunden mittels einer Autorisierungsanfrage anbieten. Das Angebot erfolgt über die Definition eines Parametersatzes innerhalb einer http Post Action, welche auf das Autorisierungsobjekt (URL) des Network Operators zeigt. Hierzu fügt der Content Provider innerhalb HTML Seite den folgenden Code-Abschnitt ein:

```
<p>
<form method="POST" action="http://NO-host/auth-object" accept-
charset="ISO8859-1">
  <input type="hidden" name="ServiceID" value="1234567890">
  <input type="hidden" name="ServiceName" value="Newsticker">
  <input type="hidden" name="ServiceURL" value="http://CP-
host/service">
  <input type="hidden" name="CPID" value="Anbieter 123">
  <input type="hidden" name="Price" value="1,99">
  <input type="hidden" name="Currency" value="EURO">
  <input type="hidden" name="Timestamp" value="27-10-2005T18:33">
  <input type="hidden" name="Validity" value="26-11-2005T18:33">
  <input type="hidden" name="SessionID"
value="98846AF87897B0AC6876868F376">
  <input type="submit" value="weiter">
</form>
</p>
```

Die Parameter besitzen folgende Bedeutung:

ServiceID	Eindeutiger Identifikationscode des Dienstes. Die ServiceID muss vom Network Operator vergeben werden.
ServiceName	Eindeutiger Service Name des Dienstes
ServiceURL	URL des Dienstes, wird nach erfolgreicher Autorisierung zum Service Aufruf benutzt
CPID	Eindeutiger Identifikationscode des Content Providers. Die CPID muss vom Network Operator vergeben werden.
Price	Kosten für die Dienstenutzung.
Currency	Währungsangabe für das Attribute „Price“.
Timestamp	ISO konforme Datums- und Zeitangabe, markiert den frühestmöglichen Beginn der Dienstenutzung
Validity	ISO konforme Datums- und Zeitangabe, markiert den Ablauf der Dienstenutzung
SessionID	Session Identifikator des Content Providers

Der Parametersatz muss vom CP im Session Context / Datenbank gespeichert werden, da er für die spätere Validierung der Signatur und damit zur Überprüfung der Autorisationsantwort erforderlich ist.

Sobald der Mobile User die Service Nutzung (durch Betätigen des „Submit-Buttons“) des Dienstes selektiert hat, wird ein http request gegen das Autorisierungsobjekt des Network Operators mit den oben definierten Parametern gestellt.

```
POST http://NO-host/auth-object HTTP/1.1
```

```
Host: NO-host
```

```
User-Agent: xyz
```

```
Accept: */*
```

```
Content-Length: 123
```

```
ServiceID%3D1234567890%26ServiceName%3DXYZ+NewsTicker%26ServiceURL%3Dhttp%3A%2F%2F%2FCP%2Dhost%2Fservice%26CPID%3DAnbieter+123%26Price%3D1%2C99%26Currency%3DEURO%26Timestamp%3D27%2D10%2D2005T18%3A33%26Validity%3D26%2D11%2D2005T18%3A33%26SessionID%3D98846AF87897B0AC6876868F376
```

Die Übertragung der Parameter erfolgt URL-encoded.

Mobile User Authentisierung

Der NO muss vor jeder Dienste Autorisierung die Authentizität des Mobile Users überprüfen. Dies geschieht anhand der Client IP Adresse mittels einer Abfrage der RADIUS Datenbank, in welcher die MSISDN des MU und seine derzeit zugewiesene IP Adresse hinterlegt sind.

Das Autorisierungsobjekt prüft – nach erfolgreicher Authentisierung des MU – die übermittelten Parameter auf Plausibilität und Gültigkeit und hält diese für eine spätere Verarbeitung im Session Context vor. Die übermittelten Informationen werden gegen ein Content Repository abgeglichen. Weichen die Informationen von den bei NO hinterlegten ab, so wird der Autorisierungsvorgang mit einer Fehlermeldung an den MU abgebrochen. Weiterhin muss der NO überprüfen, ob der MU über ein ausreichendes Guthaben verfügt, um den angeforderten Dienst nutzen zu können.

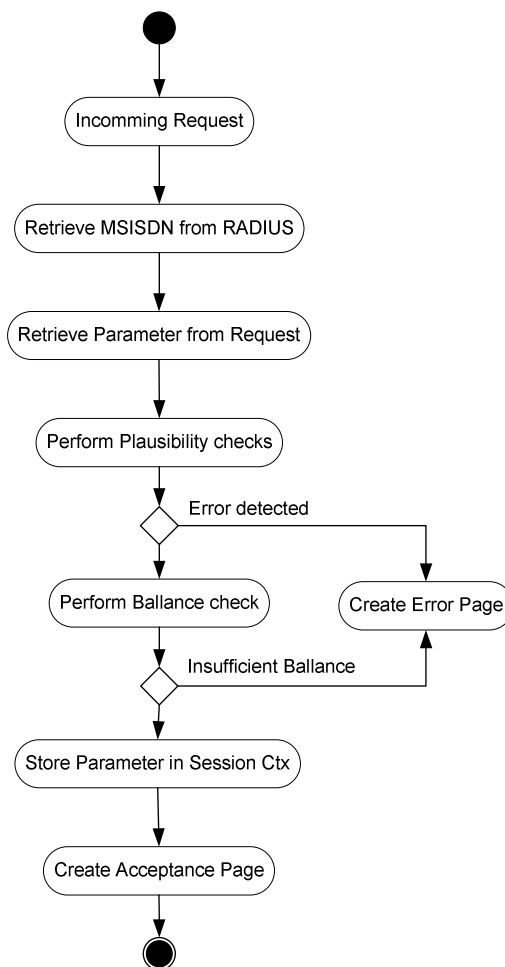


Abbildung 3: MU Authorisierung

Fällt das Ergebnis der Überprüfung positiv aus, so wird anhand der übermittelten Parameter eine HTML Seite generiert, welche den MU über die Service Details informiert und sein Einverständnis einholt.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>Bestaetigung</title>
</head>
<body>
Sie abonnieren folgenden Service:
<p>
Content-Anbieter: Anbieter 123<BR>
Content-Name: XYZ NewsTicker<BR>
Preis: 1,99 Euro<BR>
Nutzbar bis: 26.11.2005 / 18:33<BR>
</p>
<form method="POST" action=http://NO-host/confirmation accept-
charset="ISO8859-1">
    <input type="submit" value="kaufen">
    <...>
</form>
</body>
</html>
```

Zusätzlich kann der NO eine PIN Abfrage integrieren. Hierbei wird ein zusätzliches Input-Tag `<input type="text" name="PIN">` in die `<form>` Sektion eingefügt. Das Vorhalten der MU PIN sowie der Abgleich der übermittelten PIN werden in diesem Dokument nicht beschrieben.

Bestätigt der MU das Service Angebot durch Betätigung des "Submit-Buttons", so wird der Content Autorisierungsprozess beim NO angestoßen.

Content Autorisierungsprozess

Nachdem der MU das Service Angebot akzeptiert hat, wird durch das Autorisierungsobjekt eine digitale Signatur erzeugt und zusammen mit den im Session Context vorgehaltenen Parametern sowie der MSISDN des MU als Datensatz persistent beim NO gespeichert.

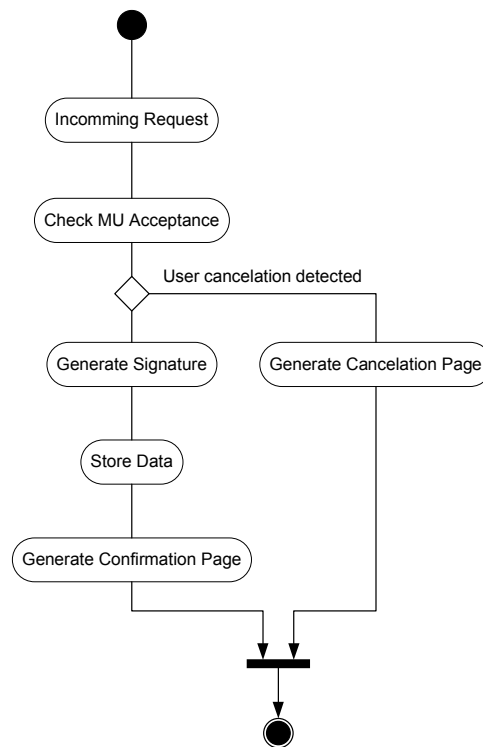


Abbildung 4: Content Autorisierung

Die CP Session Information und die Signatur der CP Parameter werden innerhalb der Bestätigungsseite (Confirmation Page) mit folgendem Code an den MU zurückgegeben.

```

<p>
<form method="POST" action="http://CP-host/service" accept-
charset="ISO8859-1">
  <input type="hidden" name="SessionID"
value="98846AF87897B0AC6876868F376">
  <input type="hidden" name="Token" value="
MhdWZM3E+l1EHgrUxhCXSjdXXcCpUN7zS+hOLx+87BN2FVT71Vzqb3GsEdTEHlGm
Sm3xU1/XAdsa9HkXA/qpBFsRApTM06JXI3vrjRM1QGBPqwV4aPJoetAojR5G5o1S
Zi5dNvzQHtcSZfLOGRG2wcvHWYEuao8aCHhQqTrHfNw="
  <input type="hidden" name="OperatorID" value="XYZ Mobilfunk GmbH">
  <input type="submit" value="weiter zum Content">
</form>
</p>
  
```

Hierbei bedeutet:

SessionID Die ursprüngliche SessionID des Content Providers
 Token „trusted Token“, digitale Signatur
 OperatorID Der Name des autorisierenden Mobilfunk Betreibers.

Die in der initialen Autorisierungsanfrage mitgegebenen Parameter werden digital signiert:

Diese sind beispielsweise (vergleiche oben):

```
ServiceID%3D1234567890%26ServiceName%3DXYZ+NewsTicker%26ServiceURL%3Dhttp%3A%2F%2FCP%2Dhost%2Fservice%26CPID%3DAnbieter+123%26Price%3D1%2C99%26Currency%3DEURO%26Timestamp%3D27%2D10%2D2005T18%3A33%26Validity%3D26%2D11%2D2005T18%3A33%26SessionID%3D98846AF87897B0AC6876868F376
```

Die Signatur wird durch den nicht-öffentlichen RSA Key des NO erzeugt [siehe Kapitel RSA Schlüsselaustausch]. Das Ergebnis ist in base64 zu codieren:

```
MhdWZM3E+l1EHgrUxhCXSjdXXcCpUN7zS+hOLx+87BN2FVT71Vzqb3GsEdTEHlGmSm3xU1/XAdsa9HkXA/qpBFsRApTM06JXI3vrjRM1QGBPqwV4aPJoetAojR5G5o1Si5dNvzQHtcSZfLOGRG2wcvHWYEuao8aCHhQqTrHfNw=
```

Die Signatur inkludiert den Hash-Wert der Übergabeparameter. Diese können zusammen mit dem öffentlichen Schlüssel des NO durch den CP validiert werden.

Innerhalb der Bestätigungsseite werden die CP Session Information und die Signatur – jetzt als „trusted Token bezeichnet“ – und die OperatorID als Parameter einer `<form>` Methode dargestellt. Die `action` dieser Methode hat nun die in der ursprünglichen Autorisierungsanfrage enthaltenen Content URL zum Ziel.

Betätigt der MU den „Submit Button“ der Confirmation Page, so werden die SessionID, der „trusted Token“ und die OperatorID zum Content Provider übermittelt. Hierbei wird die Ziel URL des angeforderten Content aufgerufen.

```
POST http://CP-host/service HTTP/1.1
Host: CP-host
User-Agent: xyz
Accept: */*
Content-Length: 399
```

```
SessionID%3D98846AF87897B0AC6876868F376%26Token%3D+MhdWZM3E%2B11EHgrUxhCXSjdXXcCpUN7zS%2BhOLx%2B87BN2FVT71Vzqb3GsEdTEHlGm%0D%0ASm3xU1%2FXAdsa9HkXA%2FqpBFsRApTM06JXI3vrjRM1QGBPqwV4aPJoetAojR5G5o1S%0D%0AZi5dNvzQHtcSZfLOGRG2wcvHWYEuao8aCHhQqTrHfNw%3D%26OperatorID%3D+XYZ+Mobilfunk+GmbH%0D%0A
```

Die Übertragung der Parameter erfolgt URL-encoded.

„trusted Token“ Validierung durch den Content Provider

Der Content Provider erhält nun eine http Anfrage welche eine SessionID und einen „trusted Token“ als POST Body beinhaltet.

Anhand der SessionID kann der CP die ursprünglichen Parameter der initialen MU Anfrage aus dem Context der mit der SessionID assoziierten Session ermittelt.

Diese sind in diesem Beispiel:

```
ServiceID%3D1234567890%26ServiceName%3DXYZ+NewsTicker%26ServiceURL%3Dhttp%3A%2F%2F%2FCP%2Dhost%2Fservice%26CPID%3DAnbieter+123%26Price%3D1%2C99%26Currency%3DEURO%26Timestamp%3D27%2D10%2D2005T18%3A33%26Valitidy%3D26%2D11%2D2005T18%3A33%26SessionID%3D98846AF87897B0AC6876868F376
```

Mithilfe des im Post Body enthaltenen „trusted Tokens“ kann der CP nun die Authentizität und somit die Gültigkeit der Anfrage ermitteln.

Hierbei wird der „trusted Token“ – die digitale Signatur – zusammen mit den ursprünglichen Parametern gegen der öffentlichen Schlüssel des NO validiert.

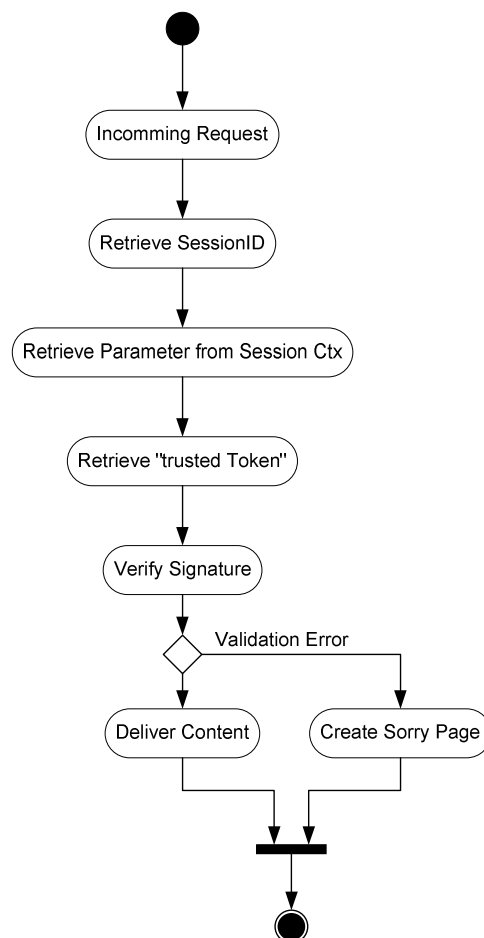


Abbildung 5: „trusted Token“ Validierung

Nach erfolgreicher Auslieferung des Contents an den MU wird der Session Context durch den CP zerstört und ggf. eine neue Session instanziiert.

Abonnement Verfahren mit MDPP

Es ist ebenfalls möglich, Abonnements auf Basis des MDP-Protokolls zu realisieren.

Hat ein MU bereits ein Service Angebot angenommen, also eine erfolgreiche Autorisierung eines Dienstes für einen definierten Zeitraum erhalten, so antwortet das Autorisierungsobjekt des NO direkt mit der Bestätigungsseite:

```
<p>
<form method="POST" action="http://CP-host/service" accept-
charset="ISO8859-1">
  <input type="hidden" name="SessionID"
value="98846AF87897B0AC6876868F376">
  <input type="hidden" name="Token" value="
MhdWZM3E+l1EHgrUxhCXSjdXXcCpUN7zS+hOLx+87BN2FVT71Vzqb3GsEdTEHlGm
Sm3xU1/XAdsa9HkXA/qpBFsRApTM06JXI3vrjRMlQGBPqwV4aPJoetAojR5G5o1S
Zi5dNvzQHtcSZfLOGRG2wcvHWYEuao8aCHhQqTrHfNw="
  <input type="hidden" name="OperatorID" value="XYZ Mobilfunk GmbH">
  <input type="submit" value="weiter zum Content">
</form>
</p>
```

Der Wert des Tokens wird hierbei aus dem Repository des NO ermittelt und erneut signiert. Die SessionID entspricht der aktuellen Session des Content Providers, welcher dem Kunden ein Service Angebot unterbreitet hat.

Das Autorisierungsobjekt überprüft, ob das Ende-Datum [Validity] bereits erreicht oder überschritten ist. Erfolgt eine Anfrage außerhalb des Gültigkeitszeitraums, so wird die Anfrage als neu – und wie oben beschrieben – behandelt.

RSA Schlüsselaustausch

Bevor das MDPP zum Einsatz kommen kann, muss ein NO ein RSA Schlüsselpaar generieren. Die Länge des zu erzeugenden Schlüssels bestimmt die Sicherheit des Verfahrens und sollte nicht unter 2048 Bit liegen.

Nach erfolgreicher Generierung eines Schlüsselpaares, ist der öffentliche und der private Schlüssel zu trennen und separate zu verwahren. Der öffentliche Schlüssel ist den Content Providern auszuhändigen.

Das folgende Beispiel zeigt die Schlüsselgenerierung mittels der freien Software openssl:

```
$ openssl
OpenSSL> version
OpenSSL 0.9.7b 10 Apr 2003
OpenSSL> genrsa -out PrivateKey 2048
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
OpenSSL> rsa -in PrivateKey -out PublicKey -pubout
writing RSA key
```

In diesem Beispiel wird ein neues Schlüsselpaar mit einer Länge von 2048 Bit generiert und gemeinsam in der Datei `PrivateKey` abgelegt. Im Anschluß wird der öffentliche Schlüssel extrahiert und in die Datei `PublicKey` geschrieben.

Beispiel: Nicht-öffentlicher Schlüssel aus `PrivateKey`

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAAzmhp6m4dfZqHhyI196Poss7HGgxswJx4KuoopZktrZv14eaY
NSmMe4GBuY9bOfuRjnJK3y+Uno jFcSKS9RGvbaTormXR3tMhKb9DeDDYsK+grcQS
N4J4jInS8IFl35IXHQdjn+VmIN3eWy3Y/NO/pSwDxYNWp4B7Qvi8zckvTjnH8MSz
5j0zJn9DE70pvAXwfYvb3tgwzeEeaNwpxpqvDVdaEUyBmmNyuQipn8aV6AcWYWWP
VxL6IkM3l7iIUrr+KNK43djWbvbNj/p1EhMeT1sTOyUDZ8JzNbuY5M9BdkjpXGWF
GX7sebEcVgXOQVuEPRSxQdANzena/aOn3CezewIDAQABAoIBAQAfz60X+By
wE8qpBW83xjfy/dp5Wx7y77sxXYGczpA3fd3EHJs+k99zQDSUs6OAEq47LrbNylb
DuosXIya8wZl144QJJyIzLkmUMBcp/XsE7IJ9aBRsIUTH6oua0afxYRzhIESi+ii
J6r23VOEp/s/JU6SnuYiWLUqMtOaAQeKlrV4ZjPULvSIyWlpXN8B1hVg4lPBYWQC
/SIRxCVCpYOf8L+Ztur7DD2aVJz2zPJobWDckDRnJBOH76hPelrM3lAZ+Ohmzsoe
ALHcq8wBAoGBAPqCEQtPwvaI/7TkD2plpF9Tom9NCHFdDWzCLLSAR15g051HtBIW
eth5hf6GKEXKYU5Gzo84NWREecLI+iW62vAyc3rTlOB55Tn+4rCNatpewUuF3xs7
aint4jki0vZGhReoFQepZ923+zvvXKo00SI40TPyIkmkMJLq/hpptgy5AoGBANLu
16OUOA/1zpVlDJs66xSCpuLD/IQ4FG1xiMt87vmn6YCpNwYS8eTXByRzY6bXlpxq
xtMDw/0vlAVWMI5ZJ+nRPlf6zc9edTYa4dWmkF8Y3yFhZu/5VfoDhEl8LGYZL2/
KqeSvvBEkiRx5LgYmjG/5hIOhft5WiE18c+Sz2/TAoGBALhgewMEVc72zp3pLz91
6CFxgSDCsnv9rR/bWuQPdbuIwNfmKpcVjJ0/9Gt9eq7DYhMm8mlfSYzfCW9gVRzo
BrS7rVs911nQ3fJts5OwoqvKz3W7nsw09bwi4zaIMO6673Q7omRGi2KeJOfx99
IFg70V6WXL4u5sF7zELMg32hAoGBALS0W1bPNwwtSFLiY99kpVpH59LjliRrqsxr
9IZnvI9zE27fCL2SY1rqADtxA1E+5zuBeF30nuX76bJ8ubWvF45TDZsaWndTmlkH
I2+peLNbbhuSg/j/d+S64To9p6tt4/hOmqs+44cRJ6ZDUG+K3CZ8wQx9FVUMbHOB
NGzg2AdzAoGAGuCCoUp314ux5SK8Dqiv5aLqJOLXt6THCPcXk9YMHhprgVbIM+aY
```

```
8CaI3x/Yyrvb7TeY9HTsAgKn+/fmBB8Pwt++K+XzpUOGR8qjqfx6Y5xsS7PZZ3Mx
SKfufb8bPEZKKbB/uGPMDawGsQ73GBD7HmSFCn3DPBVi5KQdopQDwBQ=
-----END RSA PRIVATE KEY-----
```

Beispiel: Öffentlicher Schlüssel aus `PublicKey`

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAAzmhp6m4dfZgHhyI196Po
ss7HGxswJx4KuoopZktrZv14eaYNSmMe4GBuY9bOfuRjnJk3y+UnojFcSKS9RGv
baTormXR3tMhKb9DeDDYsK+grcQSN4J4jInS8IF135IXHQdjn+VmIN3eWy3Y/NO/
pSwDxYNWp4B7Qvi8zckvTjnH8MSz5j0zJn9DE70pvAXwfYvb3tgwzeEeaNwpxpqv
DVdaEUYBmmNyuQipn8aV6AcWYWwPVxL6Ikm3l7iIUrr+KNK43djWbvbnj/plEhMe
T1sTOyUDZ8JzNbuY5M9BdkjpXGWF7sebEcVgXOQVuEpRSxQdANzena/aOn3Cez
ewIDAQAB
-----END PUBLIC KEY-----
```

Beispiel: Erzeugen einer Signatur

Die zu signierenden Parameter sind in der Datei `Parameterblock` gespeichert.

```
$ openssl dgst -sign PrivateKey -out Token < Parameterblock
```

Die Signatur ist nach diesem Aufruf in der Datei `Token` abgelegt.

Beispiel: Verifizieren einer Signatur

Die zu verifizierende Signatur ist in der Datei `Token` und die Parameter in der Datei `Parameterblock` abgelegt.

```
$ openssl dgst -verify PublicKey -signature Token < Parameterblock
Verification OK
```

Abrechnungsprinzipien

Der Content Provider ist in der Lage, auf Basis der zurück gelieferten Autorisierungsantworten des NO's, Abrechnungslisten zu erstellen und einem Network Operator zuzuordnen. Diese müssen

- einen Zeitstempel der Autorisierung
- die benutzte SessionID
- den autorisierten Content (ContentID) und
- den Nutzungspreis enthalten.

Der Network Operator kann auf Basis der oben angeführten Informationen eine Zuordnung zu den Einträgen in seiner Datenbank / Repository herstellen und die Grundlage der Abrechnung anhand eigener Einträge ermitteln.

Weiterhin ist der NO in der Lage, die autorisierten Dienstleistungen mittels der MSISDN einem Kunden zuzuordnen.

Prinzipiell sollte die Grundlage für eine Abrechnung mit einem Content Provider die Datensätze des NO sein.

Die Übermittlung der CP Datensätze an den NO ist in diesem Dokument nicht beschrieben, da ausreichend technische Lösungen zur Verfügung stehen.

Sicherheitsbetrachtungen

Man-in-the-Middle

Durch die Nutzung von End2End SSL/TLS Verbindungen kann eine Man-in-the-Middle Attacke wirkungsvoll unterbunden werden.

Replay Angriffe

Durch die temporäre Erzeugung des CP SessionID und die Kopplung der Zugangsinformationen an die SessionID sind Replay Attacken praktisch ausgeschlossen, da nach der erfolgreichen Auslieferung des Contents die Session zerstört wird.

Reproduktion von Tokens

Durch die Verwendung des asynchronen RSA Verfahrens ist dies nur möglich, wenn ein potentieller Angreifer in Besitz des nicht-öffentlichen Schlüssels gelangt. Dieser ist jedoch, wie prinzipiell alle Geheimnisse – für Dritte unzugänglich aufzubewahren.

Identifikation des Benutzers

Eine Identifikation des Benutzers seitens des Content Providers ist nicht möglich, da keinerlei personenbezogenen Informationen, wie z.B. MSISDN oder wiederkehrende Informationen, z.B. XID übertragen werden.

Betrug durch Parameter Manipulation

Durch die erforderliche Registrierung des Content Providers beim Network Operator und die dadurch mögliche Plausibilitätsprüfung der übermittelten Daten, können Manipulationen erkannt und der Autorisationsprozess abgebrochen werden. Ein CP erkennt maximal die Zugehörigkeit eines MU zu einem NO.

Betrug durch falsche Token Parameter

Bedingt durch die Tatsache, dass die Content relevanten Informationen seitens des CP vorgehalten werden und nicht an den zu übermittelten „trusted Token“ geknüpft sind, kann ein CP Manipulationen erkennen und die Content Auslieferung verhindern.

Betrug durch falsche Abrechnungsdaten eines Content Providers

Abrechnungsdaten eines CP werden mit den Autorisierungsdatensätzen des NO verglichen. Falsche Daten können erkannt und aussortiert werden.

Generelle Betrachtung

Durch die dezentrale Datenhaltung und die Möglichkeit übertragene Parameter abzugleichen, fallen Manipulationen auf. Weiterhin gewährleistet das RSA Signatur Verfahren eine zuverlässige Verifikation von Autorisierungstoken.

Weiterhin sind Angriffe gegen das Autorisierungssystem des NO als unwahrscheinlich einzustufen, da es nur innerhalb des GPRS/UMTS –Netzwerkes eines NO erreichbar ist und keinen Zugang aus dem Internet besitzt.

Betrügerische Manipulationen gegenüber dem Autorisierungssystem sind immer an eine SIM Karte gebunden. Somit können böswillige Kunden erkannt und blockiert werden. Bei der missbräuchlichen Nutzung von z.B. entwendeten SIM Karten besteht die Möglichkeit der netzseitigen Sperrung.

Referenz

RFC2616	Hypertext Transfer Protocol HTTP/1.1
RFC2138	Remote Authentication Dial In User Service
RFC2313	PKCS #1 RSA Encryption Version 1.5
RFC2246	Transport Layer Security
RFC1738	URL encoding
HTML	W3C-Recommendation 24. December 1999
openssl	open source software, siehe www.openssl.org
ISO 8601	International Date and Time notation

Annex 1 Parameter Definition

Name	Typ	Länge	Inhalt	Kommentar
ServiceID	String	16	Zwischen NO & CP vereinbarter Service Identifikator	
ServiceName	String	64	Name des Service Angebots	
ServiceURL	String	128	URL des Services	
CPID	String	16	Zwischen NO & CP vereinbarter Service Provider Identifikator	
Price	String	6	Preis eines Service mit 2 Nachkommastellen	z.B. 1,99
Currency	String	5	Angabe der Währungseinheit	EURO USD CFR
Timestamp	String	16	Datum & Zeit Angabe im Format DD-MM-YYYYThh:mm	Aktueller Zeitstempel der Aktion
Validity	String	16	Datum & Zeit Angabe im Format DD-MM-YYYYThh:mm	Angabe der letztmöglichen Nutzung
SessionID	String	64	Session Identifikator eines Content Providers	
OperatorID	String	64	Kennung eines Network Operators	
Token	String	256	Digitale Signatur	

Annex 2 Implementation Examples

Die nachstehenden Code Beispiele veranschaulichen lediglich die Funktionsweise. Sie stellen keine Referenzimplementation dar !

ServiceObject eines Content Providers

```
#!c:/Tcl/bin/tclsh.exe
# Content Provider Service Object

# load network cgi library
package require ncgi
# load session handler
source session.tcl
# load the RSA handler
source signature.tcl
# load the http library
package require http

# initialize variables
set SessionID ""
set ServiceURL "http://localhost"
set ServiceName "Cyber-Dynamix Newsticker"
set ServiceID "1234567890"
set Price "1,99"
set CPID "Cyber-Dynamix GmbH Germany"
set Currency "Euro"
set Timestamp [clock format [clock seconds] -format %d-%m-%YT%T -gmt true]

# set the service validity date 30 days
set until [expr [clock seconds] + (86400 \* 30)]
set Validity [clock format $until -format %d-%m-%YT%T -gmt true]

set a [array get env REQUEST_METHOD ]
set Method [lindex $a 1]

if {$Method == "POST" } {
    # check if authToken is posted
    ::ncgi::parse
    # check signature
    set Token [::verify [::ncgi::value Token]]

    set context [::getctx [::ncgi::value SessionID]]

    # get timestamp from context
    set atime [lindex $context 7]
    regsub T $atime " " atime
    set atime [lindex $atime 0]
    regsub -all \\- $atime " " atime
    set atime [lindex $atime 2][lindex $atime 1][lindex $atime 0]

    # get validity from context
    set btime [lindex $context 8]
    regsub T $btime " " btime
    set btime [lindex $btime 0]
    regsub -all \\- $btime " " btime
    set btime [lindex $btime 2][lindex $btime 1][lindex $btime 0]

    # check delivered values against stored session data
    if {[lindex $context 0] == [::ncgi::value SessionID] &&
        [lindex $context 1] == $ServiceID &&
        [lindex $context 2] == $ServiceName &&
        [lindex $context 3] == $ServiceURL &&
        [lindex $context 4] == $CPID &&
        [lindex $context 5] == $Price &&
        [lindex $context 6] == $Currency &&
        [clock scan $atime] < [clock seconds] &&
        [clock scan $btime] > [clock seconds] } {
```

```

        set content [::http::geturl http://www.suse.de ]
        puts "Content-Type: text/html"
        puts ""
        puts -nonewline [::http::data $content]

        # destroy the session
        ::destroy [::ncgi::value SessionID]

        exit

    }

    puts "Content-Type: text/html\n"
    puts ""
    puts " <html>
        <body>
            <h2>Fehler</h2>
            <p>
            <h3>Ihre Autorisierung konnte nicht bestätigt werde
!</h3>
            </p>
            <a href='http://localhost/cgi-
bin/ServiceObject.cgi'>weiter</a>
        </body>
    </html>"
    exit
}

if { $Method == "GET" } {
    set SessionID [::create]
    set status [::setctx $SessionID $ServiceID $ServiceName $ServiceURL $CPID $Price
$Currency $Timestamp $Validity ]

    puts "Content-Type: text/html\n"
    puts ""
    puts " <html>
        <head>
            <title>ServiceObject</title>
        </head>
        <body>
            <h2>NewTicker Online</h2>
            <p>
            <h3>The quick brown fox jumps<br>over the leazy dog</h3>
            </p>
            <form method='POST' action=/cgi-bin/AuthObject.cgi accept-
charset=ISO8859-1>
                <input type=hidden name=ServiceID value='$$ServiceID'>
                <input type=hidden name=ServiceName
                value='$$ServiceName'>
                <input type=hidden name=ServiceURL value='$$ServiceURL'>
                <input type=hidden name=CPID value='$$CPID'>
                <input type=hidden name=Price value='$$Price'>
                <input type=hidden name=Currency value='$$Currency'>
                <input type=hidden name=Timestamp value='$$Timestamp'>
                <input type=hidden name=Validity value='$$Validity'>
                <input type=hidden name=SessionID value='$$SessionID'>
                <input type=hidden name=Status
                value='Authenticate'>
                <input type=submit value='Full story'>
            </form>
        </body>
    </html>"
}

```

AutorisierungsObjekt

```

#!c:/Tcl/bin/tclsh.exe
# Network Operator Autorization Object

# load network cgi library
package require ncgi
# load signature handler
source signature.tcl
# load session handler
source session.tcl

# parse the request body
::ncgi::parse

# store the request for token creation
set tokenIn [::ncgi::query]
set tokenIn [::ncgi::decode $tokenIn]
# perform checks against service repository
# ...
# ...
# ...

set Token [::sign $tokenIn]

if {[::ncgi::value Status] == "Confirmation" } {
    # check PIN
    # ...
    # confirm payment
    # ...

    puts "Content-Type: text/html\n"
    puts ""
    puts " <!DOCTYPE HTML PUBLIC \\"-//IETF//DTD HTML 2.0//EN\">
        <html>
            <head>
                <title>Bestätigung</title>
            </head>
            <body>
                <p>
                    <h2>XYZ Mobilfunk GmbH</h2>
                </p>
                <h2>Sie haben den folgenden<br>Service erfolgreich gebucht:</h2>
                <p>
                    <table border=1>
                        <tr><td>Anbieter</td><td>[::ncgi::value CPID]</td>
                        <tr><td>Service</td><td>[::ncgi::value ServiceName]</td>
                        <tr><td>Preis</td><td>[::ncgi::value Price]</td>
                        <tr><td>Nutzbar bis </td><td>[regsub T [::ncgi::value
Validity] \ / \ ]<td>
                    </table>
                </p>
                <form method='POST' action=/cgi-bin/ServiceObject.cgi>
                    <input type=hidden name=SessionID value=[::ncgi::value
SessionID]>
                    <input type=hidden name=Token value='$Token'>
                    <input type=hidden name=OperatorID value='XYZ Mobilfunk
GmbH'>
                    <input type=submit value='Weiter zum Content'>
                </form>
            </body>
        </html>"
    }

if {[::ncgi::value Status] == "Authenticate" } {
    # prepare payment
    # ...

    puts "Content-Type: text/html\n"
    puts ""
    puts " <!DOCTYPE HTML PUBLIC \\"-//IETF//DTD HTML 2.0//EN\">
        <html>
            <head>
                <title>Bestätigung</title>
            </head>
            <body>
                <p>

```

```

        <h2>XYZ Mobilfunk GmbH</h2>
    </p>
    <h2>Sie Abonnieren folgenden Service:</h2>
    <p>
    <table border=1>
        <tr><td>Anbieter</td><td>[::ncgi::value CPID]</td>
        <tr><td>Service</td><td>[::ncgi::value ServiceName]</td>
        <tr><td>Preis</td><td>[::ncgi::value Price]</td>
        <tr><td>Nutzbar bis </td><td>[regsub T [::ncgi::value
Validity] \ /\ ]<td>
    </table>
    </p>
    <form method='POST' action=/cgi-bin/AuthObject.cgi>
        <input type=hidden name=SessionID value='[::ncgi::value
SessionID]'\>
        <input type=hidden name=CPID value='[::ncgi::value
CPID]'\>
        <input type=hidden name=ServiceName value='[::ncgi::value
ServiceName]'\>
        <input type=hidden name=Price value='[::ncgi::value
Price]'\>
        <input type=hidden name=Validity value='[::ncgi::value
Validity]'\>
        <input type=hidden name=Token value='$Token'\>
        <input type=hidden name=OperatorID value='XYZ Mobilfunk
GmbH'\>
        <input type=hidden name=Status
value='Confirmation'\>
        Geben Sie zur Bestätigung bitte ihre PIN ein:
        <p>
        <input type=password name=PIN><br>
        </p>
        <input type=submit value='jetzt kaufen'\>
    </form>
    </body>
</html>"
}

```

SessionObject

```

# Session Handler
# (c) 2005 Cyber-Dynamix GmbH
# Syntax
# ::create
# ::setctx $SessionID {parameter1} {parameter2} {parameter3} ... {parameter10}
# ::destroy $SessionID

# Config section
set odbc "true";           # application behaves as ODBC client
set user "abc";           # DB user
set password "xyz";       # DB password
set table "test";         # DB table

#----- do not modify -----
proc init {} {
    package require mysqltcl
    global odbc user password table
    set ODBC_handle [::mysql::connect -odbc $odbc -user $user -password $password]
    ::mysql::use $ODBC_handle $table
    # puts $ODBC_handle
    return $ODBC_handle
}

proc create {} {
    # create sessionID
    set randomToken [expr { int(999999999 * rand()) }]
    set timeToken [clock clicks]
    set SessionID "$randomToken$timeToken"
    # get DB handle
    set ODBC_handle [::init]
    # insert sessionID into database
    set statement "insert into test.session values
('$SessionID',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ')"
    ::mysql::exec $ODBC_handle $statement
    ::mysql::close $ODBC_handle
    return $SessionID
}

proc destroy {SessionID} {
    # get DB handle
    set ODBC_handle [::init]
    # insert sessionID into database
    set statement "delete from test.session where SessionID = '$SessionID'"
    ::mysql::exec $ODBC_handle $statement
    ::mysql::close $ODBC_handle
    return
}

proc setctx {args} {
    # check if SessionID has been delivered

    # check if not more than 11 parameters are handed over
    if {[llength $args] > 11} {
        return "ERROR: Array size too big"
    }

    # initialize DB variables
    for {set x 0} { $x<11} {incr x} {
        set param($x) null
    }

    set x 0
    foreach count $args {
        set param($x) $count
        incr x
        # puts $param($x)
    }

    set SessionID $param(0)

    # get DB handle
    set ODBC_handle [::init]
    # update sessionID

```

```
set statement "update session SET \  
    param1='$param(1)', \  
    param2='$param(2)', \  
    param3='$param(3)', \  
    param4='$param(4)', \  
    param5='$param(5)', \  
    param6='$param(6)', \  
    param7='$param(7)', \  
    param8='$param(8)', \  
    param9='$param(9)', \  
    param10='$param(10)' \  
    where sessionID = '$SessionID'"\  
  
# puts $statement\  
  
::mysql::exec $ODBC_handle $statement\  
::mysql::close $ODBC_handle\  
return\  
}  
  
proc getctx {SessionID} {  
    # get DB handle  
    set ODBC_handle [::init]  
    # insert sessionID into database  
    set statement "select * from test.session where SessionID = '$SessionID'"  
    set result [::mysql::sel $ODBC_handle $statement]  
    set result [::mysqlnext $ODBC_handle ]  
    ::mysql::close $ODBC_handle  
    return $result  
}
```

RSA Object

```

# RSA sign & verify handler
# (c) 2005 Cyber-Dynamix GmbH

# For testing only - please store keys in dedicated files
# with dedicated access privileges !!!!

# Syntax
# ::sign token
# ::verify token

package require des
package require base64

set privateKey "-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAzmp6m4dfZqHhyI196Poss7HGgxswJx4KuoopZktrZv14eaY
NSmMe4GBuY9bOfuRjnjk3y+UnoJFcSKS9RGvbaTormXR3tMhKb9DeDDYsK+grcQS
N4J4jInS8IFl35IXHQdjn+VmIN3eWy3Y/NO/pSwDxYNWp4B7Qvi8zckvTjnH8MSz
5j0zJn9DE70pvAXwfYvb3tgwzeEeaNwpxpqvDVdaEUYBmmNyuQipn8aV6AcWYWWP
VxL6Ikm3l7iIurr+KNK43djWbvbNj/p1EhMeT1sTOyUDZ8JzNbuY5M9BdkjpXGWF
GX7sebEcVgXOQVUePrSxQdANzena/aOn3CezewIDAQABAoIBAQAfzfn9IYK236nv
Jhjsm8pNcNPqIFCGgM8/4unmGaOQm7QZEpkcePwq2CUaiVL2fsfNGNcafz60X+By
wE8qpBW83xjfy/dp5Wx7y77sxXYGczpA3fd3EHJs+k99zQDSUs60AEq47LrbNylb
DuosXIya8wZl144QJyIzLkmUMBcp/XsE7IJ9aBRsIUTh6oua0afxYRzhIESi+ii
J6r23VOEp/s/JU6SNUyIWLUqMtOaAQeKlrV4ZjPULvSIyWlpXN8B1hVg41PBYWQC
/SIRxCVCpYOf8L+Ztur7DD2aVJz2zPJobWDckDRnJBOH76hPelrM3lAZ+Ohmzsoe
ALHcq8wBAoGBAPqCEQtPwwaI/7TkD2plpF9Tom9NChFdDwzCLLSAR15g051HtBIW
eth5hf6GKEXKYU5Gzo84NWRDecLI+iW62vAyc3rTlOB55Tn+4rCNatpewUuF3xs7
aint4jki0vZGhREoFQepZ923+zvvXKo00SI40TPyIkkmJLq/hppty5AoGBANLu
16OUOA/1zpVLDJs66xSCpuLD/IQ4FG1xiMt87vmn6YCPNwYS8eTXByRzY6bXlpxq
xtMDw/0vlAVWMI5ZJ+nRPlf6zcedTYa4dWmkF8Y3yFhZu/5VfODhEl8LGYZL2/
KqeSvvBEkiRx5LgYmjG/5hIOhft5WiE18c+Sz2/TAoGBALhgewMEVc72zp3pLZ91
6CFxgSDCsnv9rR/bWuQPdbuIwNfmKpcVjJ0/9Gt9eq7DYhMm8mlfSYzfcW9gVRzo
BrS7rVs91lnQ3fJts5OWwoqvKz3W7nsw09bwi4zaIMO6673Q7omRGi2KeJOIFX99
IFg70V6WXL4u5sF7zELMg32haOGBALS0W1bPNwvtSFLiY99kpVpH59LjliRrqsxr
9IZnvI9ze27fCL2SYlRqADtXAlE+5zuBeF30nuX76bJ8ubWvF45TDZsaWndTmlkH
I2+peLNBbhuSg/j/d+S64To9p6tt4/hOmqs+44cRJ6ZDUG+K3CZ8wQx9FVUMBHOB
NGzg2AdzAoGAGuCCoUp314ux5SK8Dqiv5aLqJOLXt6THCPcXk9YMHhprgVBIM+aY
8CaI3x/YyrvB7TeY9HTsAgKn+/fmBB8Pwt++K+XzpuOGR8qjqfx6Y5xsS7PZZ3Mx
SKfufb8bPEZKKbB/uGPMdAwGsQ73GBD7HmSFCn3DPBV15KQdopQDwBQ=
-----END RSA PRIVATE KEY-----"

set publicKey "-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzmp6m4dfZqHhyI196Po
ss7HGgxswJx4KuoopZktrZv14eaYNSmMe4GBuY9bOfuRjnjk3y+UnoJFcSKS9RGv
baTormXR3tMhKb9DeDDYsK+grcQSN4J4jInS8IFl35IXHQdjn+VmIN3eWy3Y/NO/
pSwDxYNWp4B7Qvi8zckvTjnH8MSz5j0zJn9DE70pvAXwfYvb3tgwzeEeaNwpxpqv
DVdaEUYBmmNyuQipn8aV6AcWYWWPVxL6Ikm3l7iIurr+KNK43djWbvbNj/p1EhMe
T1sTOyUDZ8JzNbuY5M9BdkjpXGWFGX7sebEcVgXOQVUePrSxQdANzena/aOn3Cez
ewIDAQAB
-----END PUBLIC KEY-----"

proc sign {token} {
    global privateKey
    # encrypt the token with the private key and encode the result in base64
    set authToken [::base64::encode [DES::des -mode encode -key $privateKey $token]]
    # puts $authToken
    return $authToken
}

proc verify {token} {
    global publicKey
    # decode the token from base64 and decrypt the result with the public key
    set token [DES::des -mode decode -key $publicKey [::base64::decode $token]]
    # puts $token
    return $token
}

```

Database Schema

```
<!ELEMENT ROOT (row)+>
  <!ELEMENT row
(sessionid,param1,param2,param3,param4,param5,param6,param7,param8,param9,p
aram10)>
    <!ELEMENT sessionid (#PCDATA)>
    <!ELEMENT param1 (#PCDATA)>
    <!ELEMENT param2 (#PCDATA)>
    <!ELEMENT param3 (#PCDATA)>
    <!ELEMENT param4 (#PCDATA)>
    <!ELEMENT param5 (#PCDATA)>
    <!ELEMENT param6 (#PCDATA)>
    <!ELEMENT param7 (#PCDATA)>
    <!ELEMENT param8 (#PCDATA)>
    <!ELEMENT param9 (#PCDATA)>
    <!ELEMENT param10 (#PCDATA)>
```