Dokumentation PhexProxy

1.)Beschreibung Backend
- 1.1.)Phex
- 1.2.)BitJoe Paris
- 1.3.)BitJoe Distributed Paris

2.)BitJoe Paris.pl

2.1.)Programminterne Abschnitte
- 2.1.1.)Abschnitt IPBlocker
- 2.1.2.)Abschnitt LoadCheck
- 2.1.3.)Abschnitt Entschlüsselung Handynachricht
- 2.1.4.)Abschnitt Licence Check

2.2.)Programminteren Funktionen
- 2.2.1.)CheckStatusFlag()
- 2.2.2.)FindFunction()
- 2.2.3.)ResultFunction()
- 2.2.4.)DownloadStartFunction
- 2.2.5.)DownloadEndFunction
- 2.2.6.)LicenceFunction
- 2.2.7.)ErrorFunction

2.3.)Programmexterne BitJoe Module

2.3.1.)IO.pm
- 2.3.1.1.)readSocket()
- 2.3.1.2.)writeSocket()
- 2.3.1.3.)CreatePhexConnection()
- 2.3.1.4.)CreateProxySocket()
- 2.3.1.5.)CreateHandyDistributedSocket()
- 2.3.1.6.)CreateSocketToParis()
- 2.3.1.7.)readRandomBytes()
- 2.3.1.8.)ReadFileIntoArray()
- 2.3.1.10.)ReadFileIntoScalar()
- 2.3.1.11.)WriteFile()

2.3.2.)Gzip.pm
- 2.3.2.1.)decompress_string_zlib()
- 2.3.2.2.)compress_string_zlib()
- 2.3.2.3.)compress_zlib()
- 2.3.2.4.)decompress_zlib()
- 2.3.2.5.)compress_gzipprog()
- 2.3.2.6.)decompress_gzipprog()
- 2.3.2.7.)compress_string_gzipprog()

2.3.3.)Time.pm

- 2.3.3.1.)GetValid8DayDateForLicence()
- 2.3.3.2.)GetCurrentTimeStamp()
- 2.3.3.3.)MySQLDateTime()

- 2.3.3.4.)SimpleMySQLDateTime()

2.3.4.)Phex.pm
- 2.3.4.1.)readToken()
- 2.3.4.2.)find()
- 2.3.4.3.)result()
- 2.3.4.4.)del()

2.3.5.)Logging.pm
- 2.3.5.1.)LogToFileInitDistr()
- 2.3.5.2.)LogToFileGetResultsDistr()
- 2.3.5.3.)LogToFileStartDownloadDistr()
- 2.3.5.4.)LogToFileFinishDownloadDistr()
- 2.3.5.5.)LogToFileInvalidLicenceDistr()
- 2.3.5.6.)LogToFileInit()
- 2.3.5.7.)LogToFileGetResults()
- 2.3.5.8.)LogToFileStartDownload()
- 2.3.5.9.)LogToFileFinishDownload()
- 2.3.5.10.)LogToFileInvalidLicence()
- 2.3.5.11.)LogToSql()

2.3.6.)PhexSortRank.pm
2.3.6.1.)PhexSortRank()
- 2.3.6.1.1.)Abschnitt For-Content-Splitter
- 2.3.6.1.2.)Abschnitt For-Double SortedHashRefResults
- 2.3.6.1.3.)Abschnitt While-Ergebniszusammenstellung
- 2.3.6.1.4.)Abschnitt Sortierung
- 2.3.6.2.)GetKeyWordMatching()
- 2.3.6.3.)GetSpeedPoints()
- 2.3.6.4.)SortArray()

2.3.7.)CheckSum.pm
- 2.3.7.1.)MD5ToHEX()
- 2.3.7.2.)MD4ToHEX()
- 2.3.7.3.)SHA1ToHEX()

2.3.8.)ResultCache.pm
- 2.3.8.1.)readCache()
- 2.3.8.2.)writeCache()

2.3.9.)CryptoLibrary.pm
- 2.3.9.1.)GetPrivateCryptoKeyFromDatabase()
- 2.3.9.2.)Encrypt()
- 2.3.9.3.)Decrypt()
- 2.3.9.4.)CreateCryptoObject()
- 2.3.9.5.)SimpleRandom()
- 2.3.9.6.)URandom()
- 2.3.9.7.)SimpleURandom()
- 2.3.9.8.)GenerateTemporaryKey()
- 2.3.9.9.)ShuffleArrayAdvanced()
- 2.3.9.10.)ShuffleArray()

2.3.10.)LicenceManagement.pm
- 2.3.10.1.)CheckLicence()

2.3.11.)LicenceGenerator.pm
- 2.3.11.1.)GenerateALLLicenceForToday()
- 2.3.11.2.)GenerateKombiJavaMP3LicenceForToday()
- 2.3.11.3.)GenerateKombiKlingelDocuLicenceForToday()
- 2.3.11.4.)GenerateKombiBilderVideoLicenceForToday()
- 2.3.11.5.)GenerateDocuLicenceForToday()
- 2.3.11.6.)GenerateJavaLicenceForToday()
- 2.3.11.7.)GenerateVideoLicenceForToday()
- 2.3.11.8.)GenerateKlingelLicenceForToday()
- 2.3.11.9.)GenerateMP3LicenceForToday()
- 2.3.11.10.)GenerateBildLicenceForToday()
- 2.3.11.11.)AddLicenceToSQL()
- 2.3.11.12.)GenerateLicence()
- 2.3.11.13.)SaveLicenceSQLToFile()
- 2.3.11.14.)SaveCryptoSQLToFile()

2.3.12.)LicenceTransfer.pm
- 2.3.12.1.)EncryptLicenceForTransfer()
- 2.3.12.2.)DecryptLicence()
- 2.3.12.3.)TransferEncryptedContent()
- 2.3.12.4.)CheckIfEncryptedFileExists()
- 2.3.12.5.)SendCheckSumOfFiles()
- 2.3.12.6.)InstallLicenceToDataBase()
- 2.3.12.7.)TestInstalledLicence()

2.3.13.)SQL.pm
- 2.3.13.1.)SQLConnect()

2.3.15.)SortArray.pm
- 2.3.15.1.)Discard_Duplicates()
- 2.3.15.2.)Sort_Table()

2.3.16.)Mail.pm
- 2.3.16.1.)SendNoHandyClientSocketMail()
- 2.3.16.2.)SendProxyDownMail()
- 2.3.16.3.)SendMail()

2.3.17.)Filter.pm
- 2.3.17.1.)SpamFilter()
- 2.3.17.2.)SizeFilter()

2.3.18.)IPFilter.pm
- 2.3.18.1.)IPBlocker()
- 2.3.18.2.)SimpleFilter()
- 2.3.18.3.)AdvancedFilter()
- 2.3.18.4.)ReloadFilter()
- 2.3.18.5.)GenerateIPAdressesFromRange()

1.)Beschreibung Backend

Das BitJoe Paris Backend besteht aus den Modulen Phex, der Backendsoftware Paris.pl und der erweiterten, verteilt arbeitenden Software Distributed Paris. Die beiden Paris Backends sind in der Programmiersprache PERL realisiert worden. Hauptaufgabe des Backend ist es die Suchanfragen des BitJoe Paris Handyclients, mittels AES gesicherter und bandbreitenschonender Gzip Kompression, anzunehmen, auszuwerten und passende Ergebnisse an den Handyclient weiterzuleiten. BitJoe Communications GmbH setzt aktuell 3 Server für Testzwecke der Software BitJoe Paris ein. Bei zwei der 3 Server handelt es sich um einen Dualcore Operon mit 1,8 GHZ und 4 GB Ram, der dritte Server ist ein Celeron mit 2,4 GHZ und 512 MB Ram. Bei moderaten Netzwerkeinstellungen mit maximal 45 Ultrapeer Verbindungen liefert dieser Server auf Celeron Basis stabile Ergebnisse und trägt somit zu einer preisgünstigen Serverstrukur bei.

1.1.)Phex

Eines der 3 Backendmodulen stellt der durch BitJoe Communications GmbH modifizierte Phex dar. Die Software ist eine auf Basis der Java Technologie entwickelten Filesharingsoftware, die auf dem Gnutella Protokoll aufsetzt und auf Basis der GNU Public Licence vertrieben und modifiziert werden darf. Aktuell läuft die Software unter Linux und Windows, im Testeinsatz werden zur Zeit Linuxserver eingesetzt. Der Phex wurde dahingehend optimiert, dass er auf einem zusätzlichen, nur über die interne IP 127.0.0.1 zu erreichenden Port, auf TCP Verbindungen wartet, auf denen die BitJoe Paris Software Paris.pl die Suchanfragen des Handyclient an den Phex übergibt. Der Phex startet anschließend eine neue Suchanfrage und filtert dabei automatisch alle Ergebnisse herraus, die nicht dem übergebenen Dateityp gleichen. Dadurch wird im weiteren Verlauf Rechenzeit im Backend Paris.pl gespart und zusätzlich wirkungsvoll ein primitiver Spamfilter aufgebaut. Innerhalb eines fest definierten Zeitkontigentes fragt der Handyclient kontinuierlich das Backend Paris.pl nach bereits vorhandene Ergebnissen des Phex an. Das Backend Paris.pl dient
hierbei als Bindeglied zwischen Handyclient und Phex. Je nach verwendeter Server und Netzwerkhardware sollten die Netzwerkeinstellungen angemessen angepasst werden, als Richtwerte dienen hierbei die Einstellung von maximal 150 gleichzeitigen Ultrapeers bei 2 Core Servern mit 4 GB Ram. Der Phex ist für das

Auffinden von Quellen für einen Suchbegriff zuständig.

1.2.)BitJoe Paris

Die Software BitJoe Paris Paris.pl ist eine ,in PERL geschrieben, Kommunikationssoftware, die als Bindeglied zwischen Handyclient und dem Backendmodul Phex fungiert. Als Besonderheit gilt hierbei die 128 Bit AES verschlüsselte und mittels gzip komprimierter TCP Kommunikationsverbindung mittels derer Handyclient und Backend Paris.pl
miteinander kommunizieren. Die TCP Verbindung zwischen der Software Paris.pl und der Backend Software Distributed Paris erfolgt ebenfalls auf diese Art und Weise. Dadurch wird der Kommunikationoverhead zwischen Handy und Backend minimiert.Je nach Länge des Suchwortes fielen testweise weniger als 1024 Bytes als Traffic zwischen Handy und Backend an. Dies minimiert die Handy Internet Traffic Kosten für den Endverbraucher. Paris.pl folgt dem einfachen Informatikregeln der Eingabe,Datenverarbeitung und anschließende Ausgabe. Ein Handyclient verbindet sich zum Backend Paris.pl - wobei empfohlen wird die distributed Variante des BitJoe Paris Backend zu verwenden - und stellt eine Suchanfrage an das Backend Paris.pl. Dieses wiederum entschlüsselt zuerste die ankommenden Daten, dekomprimiert diese und prüft anschließend, ob der Handyclient eine gültige Lizenz und damit zur Suche authorisiert ist. Anschließend wird eine 32 stellige zufällige ID Nummer generiert und diese zusammen mit dem Suchwort und dem geforderten Dateityp an den Phex weiter. Dieser startet anschließend eine neue Suchanfrage. Nach einem festgelegten Timerintervall versucht der Handyclient nun die Ergebisse zu seiner Suchanfrage abzuholen und stellt dazu einen Result Request an das Backend Paris.pl bzw. das Backend Distributed Paris.pl. Diese nehmen die übergebenen IDs des Handyclient und prüfen den Phex über den erweiterten Phex Server Socket anhand dieser ID auf Ergebnisse. Liefert der Phex Ergebisse aus, werden diese enschließend mittes des Backendes Paris bzw Distributed Paris die Ergebnisse nach einem speziellen Rankingverfahren sortiert und mittel Gzip komprimiert an den Handyclient weitergeleitet. Das Distributed Paris Backend leitet bei einer Suchen/Ergebniss Anfrage des Handys diese automatisch an alle angeschlossenen Server mit
Phexsoftware weiter und integriert alle von diesen Servern ausgelieferten Ergebnisse und präsentiert dem Handyclient dann diese Ergebnisse. Das Aufsetzen des BitJoe Backends auf
Basis eines Linuxservers mit vorinstalliertem Debian dauert etwa 1h Arbietstunden für einen versierten Administrator. Dazu muss dem Linuxsystem eine grafische Oberfläche hinzugefügt
werden, Java Version 1.6 installiert werden, das grafisches Administrationstool VNC installiert werden und schließlich BitJoe Paris und BitJoe Distributed Paris aufgesetzt werden,
was 10 min in Anspruch nimmt. Danach muss die grafische Oberfläche gestartet werden, das Backendmodul Phex unter dieser Oberfläche gestartet werden. Hierzu sollte das Programm phexrestart.pl genutzt werden, was den Phex im Falle eines Absturzes sofort wieder neustartet.
Die Aufgabe des BitJoe Paris Backendes ist es die Schnittstelle zwischen Handyclient und Phex darzustellen, wobei auf eine bandbreitenschonende und verschlüsselte Kommunikation zwischen
Paris Backend und Handyclient geachtet wird.

1.3.)BitJoe Distributed Paris

Das Distributed Paris Programm ist für die perfekte Skalierbarkeit des BitJoe Paris Systems zuständig. Hierbei werden verschiedene normale Paris Server im distributed Programm integriert und zu sogenannten Clustern zusammengefasst. Ein Cluster sollte aus etwa 3 bis 5 Server bestehen. Theoretisch können unbegrenzt viele normale Backend Server als Cluster zusammengefasst werden, allerdings summieren sich ab einer gewissen Anzahl beteiligter Rechner im Distributed Verband die Antwortzeiten eben beteiligter Rechner. So können beispielsweise bei einer Antwortzeit von 1 Sekunde pro beteiligtem Rechner sich die Gesamtzahl der Antwortzeiten auf bis zu 50 Sekunden summieren, wenn 50 Rechner in einem Cluster zusammengefasst werden. Im Handyclient muss dann nur diese IP des Clusterservers angegeben werden, er kümmert sich dann um das Verteilen der Suchanfragen und das Abholen der Ergebnisse der angeschlossenen Rechner. Der Distributed Server kann auch gleichzeitig als normales Backend funktionieren, ohne dass es zu nennenswerten Beeinträchtigungen des laufenden Betriebes kommt. Zusammenfassend lässt sich die Aufgabe des Distributed Programmes als Bindeblied zwischen den verschiedenen Paris Servern beschreiben.

2.)BitJoe Paris.pl

Paris.pl ist eine TCP Client und Serverstruktur, deren Aufgabe es ist, Anfragen vom Handyclient an den Phex weiterzureichen und die Ergebnisse des Phex zu sortieren und anschließend diese dem Client mittels gzip komprimiert zugänglich zu machen.

2.1.)Programminterne Abschnitte

- Verbindungen des HandyClients werden in while Schleife verarbeitet
- Parallele Verarbeitung der Verbindungen mittels Prozessabspaltung

2.1.1.)Abschnitt IPBlocker

- Kommunikation zu vorher definierten IPs kann geblockt/unterbunden werden

2.1.2.)Abschnitt LoadCheck

- Loadcheck sorgt dafür,dass ab einer bestimmten Serverlast keine Anfragen mehr entgegengenommen werden

2.1.3.)Abschnitt Entschlüsselung Handynachricht

- Mittels Public Key des Handys wird die Kommunikation von Handy zum Paris Backend entschlüsselt

2.1.4.)Abschnitt Licence Check

- Nur valide Handyclients, die eine gültige Lizenz vorweisen können, dürfen mit dem Backend kommunizieren

2.2.)Programminteren Funktionen

- Verwendung von speziellen Bibliotheken des BitJoe Paris Backendes

2.2.1.)CheckStatusFlag()

- Bestimmung, welches Protokoll durchgeführt wird: Suchen, Ergebnisse abholen, Logs senden

2.2.2.)FindFunction()

- Verarbeitung einer eingehenen Suchanfrage, Weiterleitung an den Phex, Generierung einer Ergebniss ID für den Handyclient, mittels deren man Ergebnisse einer Suchanfrage zuordnen/abholen kann

2.2.3.)ResultFunction()

- Ausliefern von Suchergebnissen zu einer Anfrage mittels Ergebniss ID, Ergebnissortierung wird mittels PhexSortRank.pm ausgeführt

2.2.4.)DownloadStartFunction

- Protokoll zum Logging bei gestartetem Download

2.2.5.)DownloadEndFunction

- Protokoll zum Logging bei beendetem Download

2.2.6.)LicenceFunction

- Prüfen auf gültige Lizenz (Funktion wird nicht verwendet)

2.2.7.)ErrorFunction

- Allgemeine Fehlerfunktion

2.3.)Programmexterne BitJoe Module

2.3.1.)IO.pm

- Ein- und Ausgabe spezifische Funktionen

2.3.1.1.)readSocket()

- Daten von einem Socket Lesen

2.3.1.2.)writeSocket()

- Daten an einen Socket schreiben

2.3.1.3.)CreatePhexConnection()

- Aufbau eines Sockets zur Kommunikation mit dem Phex Server Thread

2.3.1.4.)CreateProxySocket()

- Socket zur Verarbeitung von Handyanfragen

2.3.1.5.)CreateHandyDistributedSocket()

- Socket für Kommunikation vom Handy zum Distributed Paris

2.3.1.6.)CreateSocketToParis()

- Socket für Kommunikation zwischen Paris und Distributed Paris

2.3.1.7.)readRandomBytes()

- Lese eine zufällige Anzahl von zufälligen Informationen

2.3.1.8.)ReadFileIntoArray()

- Öffne eine Datei und schreibe diesen Inhalt in ein Array, gib eine Referenz auf dieses Array zurück

2.3.1.10.)ReadFileIntoScalar()

- Öffne eine Datei und schreibe diesen Inhalt in ein Scalar, gib eine Referenz auf diesen Scalar zurück

2.3.1.11.)WriteFile()

- Schreibe den Inhalt, auf den eine Referenz zeigt, in eine Datei

2.3.2.)Gzip.pm

- Kompressions Modul

2.3.2.1.)decompress_string_zlib()

- Dekomprimieren einen String mittels zlib()-Perl Bliothek

2.3.2.2.)compress_string_zlib()

- Komprimiere einen String mittels zlib()-Perl Bliothek

2.3.2.3.)compress_zlib()

- Komprimieren mittels zlib()-Perl Bliothek

2.3.2.4.)decompress_zlib()

- Dekomprimieren mittels zlib()-Perl Bliothek

2.3.2.5.)compress_gzipprog()

- Komprimieren mittels exterem Gzip Programm

2.3.2.6.)decompress_gzipprog()

- Dekomprimieren mittels exterem Gzip Programm

2.3.2.7.)compress_string_gzipprog()

- Komprimieren eines Strings mittels exterem Gzip Programm

2.3.3.)Time.pm

- Zeit- und Datumsrelevanten Funktionen

2.3.3.1.)GetValid8DayDateForLicence()

- Generiere ein valides 8-Tage Datum für Lizenzfunktion

2.3.3.2.)GetCurrentTimeStamp()

- Rückgabe des aktuellen Unix Zeitstempels

2.3.3.3.)MySQLDateTime()

- Rückgabe des aktuellen Zeitstempels im Mysql DateTime() Formates

2.3.3.4.)SimpleMySQLDateTime()

- Rückgabe des aktuellen Zeitstempels im Mysql Date() Formates

2.3.4.)Phex.pm

- Funktionen, die bei der Kommunikation mit dem Phex gebraucht werden

2.3.4.1.)readToken()

- Funktion zum Einlesen des aktuellen Kommunikationstokens - ohne dem ist keine Kommunikation mit dem Phex möglich

2.3.4.2.)find()

- Funktionsabschnitt, der einen Suchbegriff an den Phex übergibt

2.3.4.3.)result()

- Ergebnisse vom Phex abolen

2.3.4.4.)del()

- Lösen von Suchanfragen im Phex mittels übergebener Ergebniss ID

2.3.5.)Logging.pm

- Protokollierungfunktionen

2.3.5.1.)LogToFileInitDistr()

- Logging bei eingehenden Suchanfragen für das Distributed Backend

2.3.5.2.)LogToFileGetResultsDistr()

- Logging beim Abholen von Ergebnissen für das Distributed Backend

2.3.5.3.)LogToFileStartDownloadDistr()

- Logging beim Starten eines Downloades für das Distributed Backend

2.3.5.4.)LogToFileFinishDownloadDistr()

- Logging beim Beenden eines Downloades für das Distributed Backend

2.3.5.5.)LogToFileInvalidLicenceDistr()

- Logging beim Kommunizieren mittels falscher Lizenz für das Distributed Backend

2.3.5.6.)LogToFileInit()

- Logging bei eingehenden Suchanfragen für das normale Backend

2.3.5.7.)LogToFileGetResults()

- Logging beim Abholen von Ergebnissen für das normale Backend

2.3.5.8.)LogToFileStartDownload()

- Logging beim Starten eines Downloades für das Distributed Backend

2.3.5.9.)LogToFileFinishDownload()

- Logging beim Beenden eines Downloades für das normale Backend

2.3.5.10.)LogToFileInvalidLicence()

- Logging beim Kommunizieren mittels falscher Lizenz für das Distributed Backend

2.3.6.)PhexSortRank.pm

- Sortierung von Ergebnissen des Phex für spätere Darstellung auf dem Handyclient

2.3.6.1.)PhexSortRank()

- Hauptfunktion von PhexSortRank.pm

2.3.6.1.1.)Abschnitt For-Content-Splitter

- Aufteilen der ankommenden Phex-Ergebnisse, Kontrolle der Dateigröße und Installation der Ergebnisse in eine Hash Struktur

2.3.6.1.2.)Abschnitt For-Double SortedHashRefResults

- Zusammenführung von gleichen Ergebnissen anhand des SHA1 Wertes und entsprechende Aktualisierung der verwendeten Hash Struktur

2.3.6.1.3.)Abschnitt While-Ergebniszusammenstellung

- Vorbereitung der Ergebnisse für finale Sortierung

2.3.6.1.4.)Abschnitt Sortierung

- Eigentliche Sortierung der Ergebnisse anhand der Interen Sortierungspunkte einer Quelle

2.3.6.2.)GetKeyWordMatching()

- Funktionsteil zur Bestimmung der Sortierungspunkte

2.3.6.3.)GetSpeedPoints()

- Zuordnung der Sortierungspunkte einer Datei anhand der Übertragungsgeschwindigkeit des ausliefernden Rechners

2.3.6.4.)SortArray()

- Sortierung des Inhaltes eines Arrays

2.3.7.)CheckSum.pm

- Funktionsbibliothek zum Erstellen verschiedener Prüfsummen

2.3.7.1.)MD5ToHEX()

- Erstellung einer MD5 Prüfsumme

2.3.7.2.)MD4ToHEX()

- Erstellung einer MD4 Prüfsumme

2.3.7.3.)SHA1ToHEX()

- Erstellung einer SHA1 Prüfsumme

2.3.8.)ResultCache.pm

- Vorhalten eines Ergebniscaches für eine Suchanfrage

2.3.8.1.)readCache()

- Cache lesen

2.3.8.2.)writeCache()

- Cache schreiben

2.3.9.)CryptoLibrary.pm

- Funktionen zur Abwicklung verschlüsselter Kommunikation

2.3.9.1.)GetPrivateCryptoKeyFromDatabase()

- Bestimmung des Privaten Cryptokeys anhand eines übergebenen Public Keys basierend auf einem MySQL Services

2.3.9.2.)Encrypt()

- Funktion zum Verschlüsseln von Daten

2.3.9.3.)Decrypt()

- Funktion zum Entschlüsseln von Daten

2.3.9.4.)CreateCryptoObject()

- Erstellung eines benötigten Crypto Objektes in Perl

2.3.9.5.)SimpleRandom()

- Lieferung einer zufälligen Zahl

2.3.9.6.)URandom()

- Auslesen von zufälligen Daten aus dem urandom()-Device

2.3.9.7.)SimpleURandom()

- Erstellen von Zufallsdaten

2.3.9.8.)GenerateTemporaryKey()

- Zufallsfunktionen zum Erstellen von Lizenzen

2.3.9.9.)ShuffleArrayAdvanced()

- Zufallsfunktionen zum Erstellen von Lizenzen

2.3.9.10.)ShuffleArray()

- Zufallsfunktionen zum Erstellen von Lizenzen

2.3.10.)LicenceManagement.pm

- Funktionen für das Lizenzmanagement

2.3.10.1.)CheckLicence()

- Prüfe die Gültigkeit einer übergebenen Lizenz

2.3.11.)LicenceGenerator.pm

- Funktion zum Erstellen von Lizenzen

2.3.11.1.)GenerateALLLicenceForToday()

- Funktion zum Erstellen einer globalen Lizenz für den aktuellen Tag

2.3.11.2.)GenerateKombiJavaMP3LicenceForToday()

- Funktion zum Erstellen einer Lizenz für spezielle Kombinationen für den aktuellen Tag

2.3.11.3.)GenerateKombiKlingelDocuLicenceForToday()

- Funktion zum Erstellen einer Lizenz für spezielle Kombinationen für den aktuellen Tag

2.3.11.4.)GenerateKombiBilderVideoLicenceForToday()

- Funktion zum Erstellen einer Lizenz für spezielle Kombinationen für den aktuellen Tag

2.3.11.5.)GenerateDocuLicenceForToday()

- Funktion zum Erstellen einer Lizenz für Dokumentation Downloads für den aktuellen Tag

2.3.11.6.)GenerateJavaLicenceForToday()

- Funktion zum Erstellen einer Lizenz für Java Downloads für den aktuellen Tag

2.3.11.7.)GenerateVideoLicenceForToday()

- Funktion zum Erstellen einer Lizenz für Video Downloads für den aktuellen Tag

2.3.11.8.)GenerateKlingelLicenceForToday()

- Funktion zum Erstellen einer Lizenz für Klingeltöne Downloads für den aktuellen Tag

2.3.11.9.)GenerateMP3LicenceForToday()

- Funktion zum Erstellen einer Lizenz für MP3 Downloads für den aktuellen Tag

2.3.11.10.)GenerateBildLicenceForToday()

- Funktion zum Erstellen einer Lizenz für Bilder Downloads für den aktuellen Tag

2.3.11.11.)AddLicenceToSQL()

- Funktion, die eine aktuellen Lizenz in den SQL Server integriert

2.3.11.12.)GenerateLicence()

- Funktion, die eine Lizenz erstellt

2.3.11.13.)SaveLicenceSQLToFile()

- Speichern einer Lizenz in eine Datei

2.3.11.14.)SaveCryptoSQLToFile()

- Speichern eine CryptoKey Datei in eine externe Datei

2.3.12.)LicenceTransfer.pm

- Funktionen zum Übertragen von Lizenzen

2.3.12.1.)EncryptLicenceForTransfer()

- Verschlüsselung von Lizenzen für den anstehenden Transfer

2.3.12.2.)DecryptLicence()

- Entschlüsseln einer Lizenz (nach dem Transfer)

2.3.12.3.)TransferEncryptedContent()

- Übertragung von verschlüsselten Lizenz und Crypto Daten

2.3.12.4.)CheckIfEncryptedFileExists()

- Testing, ob eine verschlüsselte Datei existiert (aktuell noch nicht verwendet)

2.3.12.5.)SendCheckSumOfFiles()

- Sende die MD5 Werte der aktuellen Lizenzdaten via eMail an den Administrator

2.3.12.6.)InstallLicenceToDataBase()

- Installieren eine Lizenz in die Datenbank

2.3.12.7.)TestInstalledLicence()

- Prüfe eine SQL Datenbank auf die korrekte Installation der Lizenzdaten

2.3.13.)SQL.pm

- SQL spezifische Funktionen

2.3.13.1.)SQLConnect()

- Verbindung zum SQL Server aufbauen

2.3.15.)SortArray.pm

- Funktion zum Sortieren von Daten (externe Datei: PERL- Lizenz)
- Funktionalität ist nun in PhexSortRank.pm integriert
- wird nicht mehr aktiv verwendet

2.3.15.1.)Discard_Duplicates()

- Entferne Doppelte Einträge bei einer Sortierung

2.3.15.2.)Sort_Table()

- Eigentliche Sortierung der Ergebnisse

2.3.16.)Mail.pm

- Verteilung von Informationen mittels eMail (zur Zeit noch nicht verwendet)

2.3.16.1.)SendNoHandyClientSocketMail()

- Sende eMail im Fehlerfall

2.3.16.2.)SendProxyDownMail()

- Sende eMail im Fehlerfall

2.3.16.3.)SendMail()

- Funktion für Versendung von eMails

2.3.17.)Filter.pm

- Abbildung rudimentäre Filterfunktionen

2.3.17.1.)SpamFilter()

- Einfacher Spam Filter

2.3.17.2.)SizeFilter()

- Dateigrößefilter

2.3.18.)IPFilter.pm

- IP Filter

2.3.18.1.)IPBlocker()

- Abblocken von Kommunikationsverbindung bei positiven IP Matching

2.3.18.2.)SimpleFilter()

- Einfache IP Filterung mittels Textflatfile

2.3.18.3.)AdvancedFilter()

- Erweiterte Filtermöglichkeiten des IP Blockers (sehr Speicher und CPU intensiv, aktuell keine Verwendung)

2.3.18.4.)ReloadFilter()

- Filterdateien neu einlesen

2.3.18.5.)GenerateIPAdressesFromRange()

- Erstellung von IP Adressen anhand eines IP Ranges

2.3.19.)FileTypes.pm

- Zuordnung von Dateitypen zu einer Lizenzkategorie

2.3.20.)Daemon.pm

- Funktionen für gesicherte und stabile Parallelverarbeitung des Paris und Distributed Paris Backendes

3.)BitJoe DistributedParis.pl

- Distributed Paris Backend für bandbreitenschonende und verschlüsselte Kommunikation vom Distributed Server mit den angeschlossenen, normalen Paris Servern

3.1.)Programminterne Abschnitte

- Parallele Verarbeitung der Verbindungen mittels Prozessabspaltung
- Gesicherte und komprimierte Kommunikation zu den angeschlossenen Paris Servern
- Darstellung einer öffentlichen Kommunikationsschnittstelle für Handyclients
- Vorhalten eines Ergebnisscaches / Auslieferung von Cache-Ergebnissen
- Initialisierung der Verbindungs- und Ergebnisdatenbank

3.1.1.)Abschnitt Verbindungsaufbau

- Verbindungsaufbau zu allen beteiligten, normalen Paris Rechnern
- Verbindungen werden in einer Hash Struktur vorgehalten
- Versendung von eMail an Administrator im Fehlerfall

3.1.2.)Abschnitt Entschlüsselung Handynachricht

- Mittels Public Key des Handys wird die Kommunikation von Handy zum Paris Distributed Backend entschlüsselt

3.1.3.)Abschnitt Licence Check

- Prüfen auf gültige Lizenz, Abbrechen der Kommunikationsverbindung im Fehlerfall

3.1.4.)Abschnitt Suchanfrage stellen

- Absenden von Suchanfragen an alle beteiligten Paris Rechner auf Basis der Verbindungs Hash Struktur

3.1.5.)Abschnitt Ergebiss abholen

- Abholen der Ergebnisse von allen Paris Rechnern

3.1.6.)Abschnitt Ergebniszusammenführung

- Integration der Ergebnisse aller Paris Rechner in eine Hash Struktur

3.1.7.)Abschnitt Ergebniszusammenführung-Ergebnisparsing

- Ergebnisparsing anhand des BitJoe Paris Ergebnis Protokolles

3.1.8.)Abschnitt Ergebniszusammenführung-Integration

- Integration aller geparsten Ergebnisse aus 3.1.7 in eine Hash Struktur

3.1.9.)Abschnitt Ergebnis Sortierung

- Sortierung der Hash Struktur anhand der Sortierungspunkte einer Quelle

3.1.10.)Abschnitt Abschlussausführungen

- Ausliefern von Ergebnissen an den Handyclient
- Schreiben von Cache Ergebnissen
- Debugging Ausgaben
- Beenden und Schließen von offenen Kommunikationshandles