Rutgers University Security Services Library (RUSSL)

IETF Stockholm Summer 95 WTS Working Group
Simon Cooper <scooper@noc.rutgers.edu>

RUSSL Documentation:
<http://www-ns.rutgers.edu/RUSSL/>

Motivation

A need for secure authenticated network services:

- HTTP
- NNTP
- Local Public Lab Needs
  - Login/Application authentication from
    - "userless" operating systems
    - insecure networks
  - Logging

Design Criteria

- Strong Encryption
- Strong Authentication
- Per context generation session key
    - Neither party must have complete control
- A context initiator must not be required to divuldge its identity
- A context acceptor must authenticate itself
- Compromise of the service should not compromise all prior contexts
- Detection of
    - lost messages
    - replayed messages
    - altered messages
- Context must be independent of transport protocol
- The API must be well designed
    - ease of use
        - in client applications that are GUI/Event driven
        - in server applications that support multiple simultaneous connections
    - extensibility
    - modularity
    - consistancy of interface
    - portable

Overview

RUSSL provides the following services

- An encrypted channel
- Service authentication
- Client Authentication (if service requests it)
- Message authenticity
- Protection against message replay
- Status callback that allow GUI applications to provide transaction progress reports
- Context identification
- Can be used over any reliable transport mechanism
- Preservation of message bounary
- Configurable padding for privacy protected messages

Written in ANSI C. POSIX library calls used

Example

Library routines

Context management

`RusslInit`	Client side context initiation
`RusslGetParam`	Get context specific parameters
`RusslAccept`	Service side context initialization and existing context recognition
`RusslClose`	Terminate a context
`RusslSetParam`	Set context specific parameters

Message transfer functions

`RusslSend`	Send a privacy protected authentication message
`RusslRecv`	Receive a message
`RusslClearSend`	Send an authenticated message

Miscellaneous

`RusslGetErrorString`	obtain a textual description of an error code

Callbacks - Connect/Read/Write/Status/Error

`russlConnect_cb`	used to connect to a service
`russlIO_cb`	"read" and "write" data, a separate callback for each
`russlClientAuth_cb`	authentication callback on client side
`russlServiceAuth_cb`	authentication callback on service side
`russlStatus_cb`	status and transaction progress reporting
`russlError_cb`	determine action to be taken for an error

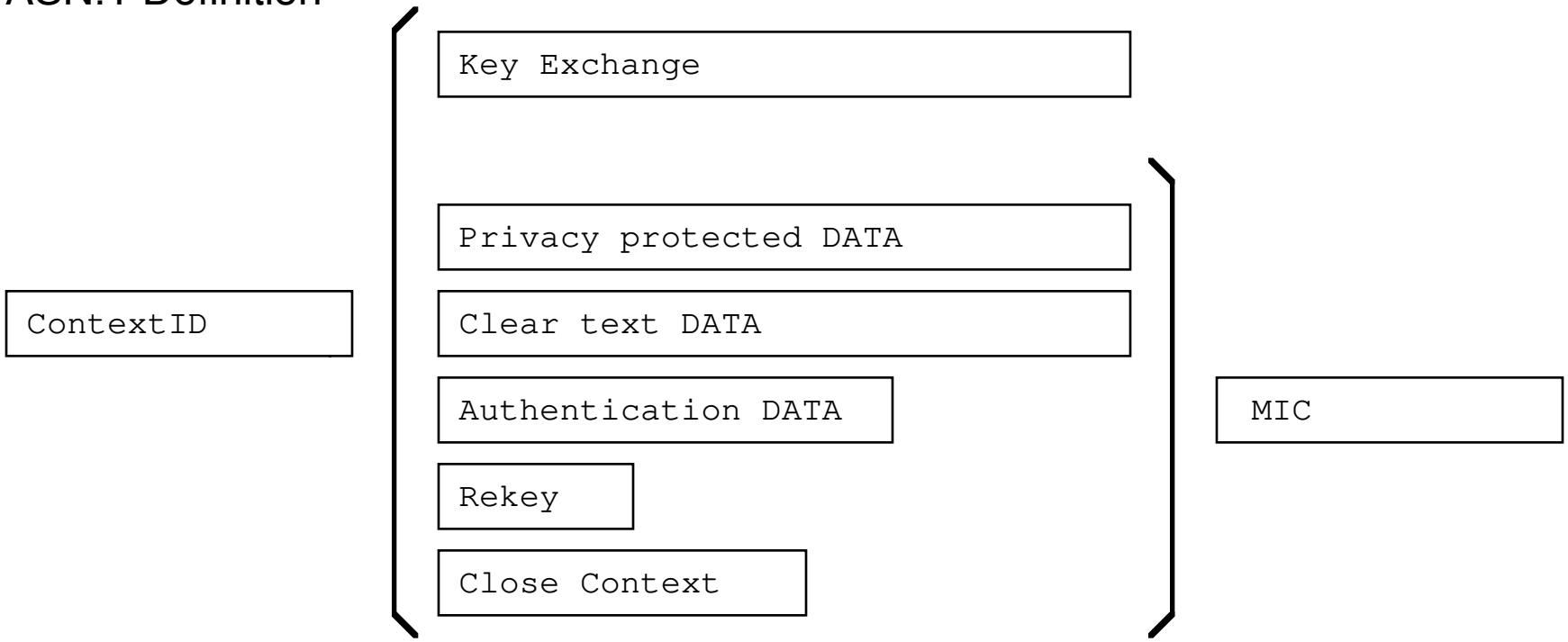
Callbacks - Client Authentication

Flexible challenge response mechanism

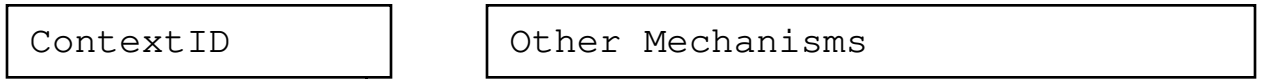
- Arbitary number of "exchanges"
- Arbitary number of prompts (a hint can be provided)
- Arbitary prompt data and response sensitivity
    - Echo response
    - Hide response
- Support for "cachable credentials"
    - Can be used instead of a response
    - Sucessful authentication can yield "cachable credentials"

Protocol

ASN.1 Definition



Future



Cryptographic Algorithms

Message digest function is MD5

Master key generation using Diffie-Hellman

- Predefined <base,mod>
- client provides <base,mod>
- master key is 128-bits in size

Session key k_n is MD5[k | n]

Service authentication using RSA signature

- Signed data is MD5[x | k_0 | y]
  (x,y from Diffie-Hellman)

Encryption algorithm is IDEA in PGP's CFB mode using k_n

MIC Generation

- "least" significant 8 bytes of MD5[m | k_n]

Rekey

- n++, recalculate k_n

Current Work

ASN.1 PER Encoding

Extensibility
- Algorithm modularization
- Mechanism negotiation

Deployment into Applications