Design

Compact, secure, peer-to-peer ZRTP

Zorg implements all the mandatory features of the ZRTP protocol, and the following notable optional features:

- Secrets cache for key continuity
- The SHA-384 hash type
- All Diffie–Hellman key agreement types, including all elliptic curve Diffie–Hellman types
- The base256 SAS type
- The `a=zrtp-hash` SDP attribute

Additionally, Zorg supports the following extensions to the ZRTP specification:

- Disabling select mandatory algorithms (e.g. disabling mandatory AES-128, only allowing AES-256)
- Compatibility with the non-compliant LibZRTP implementation

Support for the following optional features depends on the SRTP implementation used:

- The AES-192 and AES-256 cipher types
- All TwoFish cipher types
- All Skein authentication tag types

In the interest of providing a compact implementation of ZRTP for secure peer-to-peer communications only, Zorg does not implement GoClear/ClearACK, nor any proxy or MitM feature.

Modularity

The SRTP implementation and all cryptographic primitives are implemented as modules with an abstract binary interface. Default implementations are provided, but any compliant implementation can be substituted, for example:

- an alternate open source implementation
- the operating system's implementation
- a hardware-accelerated implementation
- your own implementation

Mobile networks

Zorg lets applications tune the (otherwise hardcoded by the ZRTP specification) retransmission schedule, to meet the requirements of mobile networks.

Mobile platforms

The Java implementation is designed to run with minimal Java API support (it runs on J2ME MIDP 2.0) and to have a very low memory footprint by reducing JVM garbage collector runs.

Keep in mind that running VoIP realtime applications in Java on a JVM require extreme care about the garbage collection to avoid getting frequent hole in secure audio flow or even mobile phone reboot.