CAMELOID
Secure Voice and Video

By Doctor Tek and Raven

Introduction

Cameloid is a secure TCP/UDP based Peer-to-Peer VOIP communications software, which should allow people to securely communicate regardless of sensitivity of information. CAMELOID is designed to be used for all levels of communication but we are not making *ANY* claims, you should determine how much you trust the design and structure and use it accordingly.

Cameloid is a Secret-Key over Secret-Key encryption system (No public key is used).
We cannot guarantee security of this system or any other system nor can we say that any of the included components will be impossible to break. Let us answer some possible question before we get started.

1. Why not use a public-key exchange?
This was one of those decision that needed to be made before development based on our threat model. Contemporary "secure" communication systems use public-key exchange for easy and seamless integration into existing software and new paradigms. Cameloid wasn't designed to be used unless privacy is a top concern of the users. People who fit this criteria are privacy groups, rebels, independent governments, spies, agents and hackers.

Public key systems like RSA and Diffie-Hellman have received a lot of scrutiny by the academic community and are deemed adequate for most situations since your average attacker will have zero success at attacking such a system with success. Unfortunately, the two mentioned algorithms (and similar designs) may have already have been compromised, we don't know if they have but we can't afford to trade off privacy for simplicity.

Public-Key systems are great, but we feel that if we can knock out a few problems with public-key exchange (such as man-in-the-middle attacks) it will only be conducive to the success of this project.

Secret keys intrinsically add a better means of authentication of a user since only a trusted party **should** have your secret key. Unlike WEP, CAMELOID does not use a fixed key so you can have X-number of different keys for any number of people you want to communicate with and the key can be as dynamic as you want. Also this key does not need to be changed since it is only used for authentication and session key negotiation.

2. What about Elliptic Curve Cryptography (ECC)?
This is a question that we have been faced with due to the lack of understanding of elliptic curves. Elliptic curves are *not* new nor are they a magical cure for all the fears we have about public-key systems. Elliptic curves are just another algebraic group and have been around for a century or so but only recently been proposed for use in cryptography. Elliptic curves have lots of interesting qualities one being the key sizes. You may achieve comparable security with much smaller key sizes than other public-key systems.

Remember elliptic curves in cryptography are still new and no damaging attacks on elliptic curves have been discovered to date, but that may change tomorrow.

3. So only secret-key systems are safe?
This is an unsafe assumption, attacks only get better so no algorithm (secret or public) will be safe forever. The best you can do is research and stay on top of the every changing world of cryptanalysis. Don't believe any claims of absolute security, just look at recent attacks on Rijndael , SHA-1 and other and tell us that the attackers are stagnant.

4. Why P2P?
The reason why CAMELOID is a P2P system is because it is the most flexible and in our humble opinion the most secure means for people to communicate. If we decided to create a centralized server we run a lot of risk for ourselves and for the users. Governments are becoming more strict in enforcing laws and regulations (check our website) regarding Voice and Video communications over IP. They can fine, imprison and/or shutdown service providers for not providing a means of "lawful" interception of this type communication, so logically failure to comply will result in the denial of hosting such a service and thus prevent communication between CAMELOID users. This is unacceptable.

Cameloid is designed to be a pure-P2P system in which a person can directly connect to any person with a accessible IPv4 address (IPv6 coming soon) so that means you do not even need to have an Internet connection to talk to some one on your LAN or Laptop-to-Laptop in a Ad-Hoc or Infrastructure network environment. This removes dependency of an Internet connection in order to communicate with people in your area. Which adds a whole layer of anonymity also (Pretty cool eh?)

In a NAT'ed environment a person can use something like nat-traverse (http://linide.sourceforge.net/nat-traverse/) to establish PPP sessions when behind a NAT router that you don't have the ability to setup port forwarding on. You can also establish PPP over a PC-to-PC dialup link. (**No Internet connection required**)

Terminology

NK (Negotiation Key)
This is used for user authentication, challenge-response and used to encrypted session key negotiations.

SK (Session Key)
This is used for encrypting voice data during the session.

Algorithms

Encryption Algorithms

Py(roo)
Developed by Eli Biham and Jennifer Seberry.

Phelix
Developed by Doug Whiting, Bruce Schneier, Stefan Lucks and Frederic Muller

Serpent
Developed by Eli Biham, Ross Anderson and Lars Knudsen

CAST6

Developed by Carlisle Adams and Stafford Tavares

Twofish

Developed by Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall and Neils Ferguson

Rijndael

Developed by Vincent Rijmen and Joan Daemen

Hashing Functions

SHA-256

National Security Agency

MD5

Developed by Ronald L. Rivest

TIGER

Developed by Ross Anderson and Eli Biham

Speech Codec/Compression

Speex

Jean-Marc Valin

The Session Process

Step 1: Place a call

User A initiates a call to User B.
If this is a voice session User A also tells User B the duplexity of his soundcard.

Step 2: Acknowledgment/Challenge

User B choose to accept or deny the call.
If he chooses to accept User B generates a random 32-byte challenge hashes it and enters in the pre-agreed NK for User A and uses it to encrypt a random 32-byte challenge and sends it to User A.
User B agrees on the duplexity (if this is a voice call) of the session.

Step 3: Acknowledgment/Response

User A receives the challenge decrypts and hashes it and encrypts it with the NK and sends it to User B.

Step 4: Verification

User B decrypts the received hash and compares it to the saved hash of the random challenge. If they match then the SK exchange begins, if not the connection is severed.

Step 5: SK generation and exchange

User B generates a random 256-bit SK segment encrypts it and sends it to User A.

Step 6: Final SK generation

User A receives decrypts and stores the 256-bit SK segment and then proceeds to generate his own 256-bit SK segment, encrypts and sends it to user B.

The SK will be User B's SK segment and User A's SK segment concatenated together and hashed to make a single 256-bit SK.

Step 7: Encrypted voice/video session begin

5. Technical Details

4a. Encryption

Each Key (SK or NK) is fixed to a size of 256-bits using the SHA-256 hashing function and that is used as the actual key. Each nonce is fixed at 128-bits. The nonce used is never made public only a part of it is, the actual nonce used is:

Nonce = TigerHash128(TigerHash192(inputkey (SK or NK)) ‖ PublicNonce)
* "‖" *denotes concatenation*

This allows us to have a unique and secret nonce for each packet, this was done for security purposes to add some level of protection to the nonce's used.

*Py(roo) and Phelix are used in their native states, not in any special mode. Serpent, Rijndael, CAST6, Twofish are used in Counter-Mode (CTR).

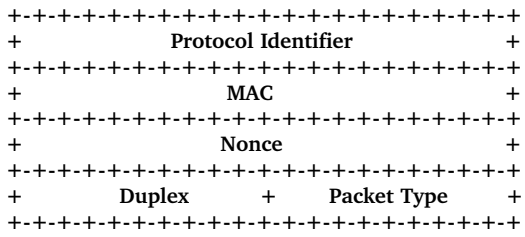
4b. Random Number Generation

The random numbers are generated using a random number generating device (default devices /dev/random and /dev/urandom on UNIX systems). You can change the devices used by setting -DURANDOM_DEVICE for the PRNG and -DRANDOM_DEVICE for the RNG in the provided Makefile .

4c. Cameloid Protocol

CAMELOID uses TCP from session initiation and UDP for voice packets. Each protected packet uses HMAC-MD5 with an 80-bit MAC key for data authentication. CAMELOID uses fixed size packets and protocol structures, the following is a breakdown of the protocol structures used:

I. ***The Authentication Header (Fig. 1)***



1. Protocol Identifier (4 bytes)
Identifies the packet as a CAMELOID packet with the following 4-byte string "CVEP".

2. MAC (16 bytes)
Message authentication code.

3. Nonce (32 bytes)
Random nonce used as specified above.

4. **Duplex (1 byte)**

Indicates the supported soundcard duplex of the sender/receiver. This field is not included in the authentication header for the cameloid video sessions.

5. **Packet Type (1 byte)**

Indicates the type of packet.

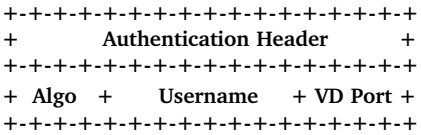
Valid Audio Packet Types

0xEF: *Indicates packet is a challenge packet*
0xF3: *Indicates packet is a response packet*
0xF9: *Indicates packet is a session initiation packet*
0xF8: *Indicates packet is a voice data packet.*
0xFF: *Indicates packet is a switch packet (for half-duplex communication packet) Tells the listener he may now speak*
0xD0: *Indicates packet is a SK negotiation packet*

Valid Video Packet Types

0x02: *Indicates packet is a challenge packet*
0x03: *Indicates packet is a response packet*
0x01: *Indicates packet is a session initiation packet*
0x05: *Indicates packet is a video data packet*
0x04: *Indicates packet is a SK negotiation packet*

II. ***Session Initiation Packet (Fig.2)***



1. **Algorithm (1 byte)**

Indicates the preferred encryption algorithm.

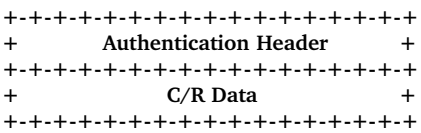
2. **Username (10 bytes)**

Identity of caller (This field is obfuscated by XORing username with the nonce).

3. **Voice/Video Data Port (2 bytes)**

Indicates the port the client is using to receive voice/video data.

III. ***Challenge/Response Packet (Fig. 3)***

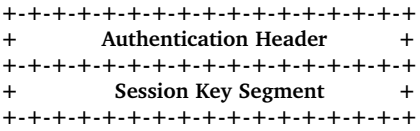


Fields:

1. C/R Data (32 bytes):

Encrypted Challenge or Encrypted Response.

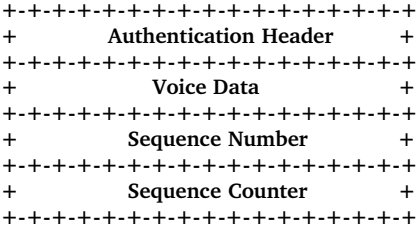
IV. **Session Key Segment Exchange Packet (Fig. 4)**



Fields:

1. Session Key Segment (32 bytes)
Encrypted SK segment data.

V. ***Voice Data Packet (Fig. 5)***

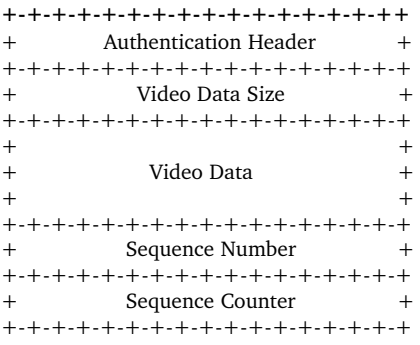


1. Voice data (200 bytes)
Compressed and Encrypted Voice Data.

2. Sequence Number (4 bytes)
32-bit counter to prevent replay attacks.

3. Sequence Counter (4 bytes)
32-bit sequence counter which is incremented every time the sequence number returns to zero. This is used to prevent replay attacks.

VI. ***Video Data Packet (Fig. 6)***



1. Video Data Size (2 bytes)
Size of the video data.

2. Video Data(? bytes)
Compressed and Encrypted Video Data.
The size of the video data is dynamic.

3. **Sequence Number (4 bytes)**
   32-bit counter to prevent replay attacks.

4. **Sequence Counter (4 bytes)**
   32-bit sequence counter which is incremented every time the sequence number returns to zero. This is used to prevent replay attacks.

Conclusion

That is essentially all for the CAMELOID protocols and design schematics if you are still uncertain about anything please don't hesitate to check out the source code. If you have any questions, comments or bug reports you can e-mail the main developer via e-mail at doctor_tek@ hushmail.com.

Thank you for you interest! Please use this as a program to fight back and regain freedom, ensure your privacy and to make the world a better place. Checkout our website at: http://cameloid.sourceforge.net for all the latest updates and news regarding cameloid.