Enhanced Closed Swarm protocol

Dušan Gabrijelčič

dusan@e5.ijs.si

Jožef Stefan Institut

November 2, 2012

Presentation outline

- Motivation
- What ECS provides
- How it works
- Protocol outline
- Implementation
- Meeting PPSP requirements

Motivation

- P2P systems are mature and often superior solution for content distribution but . . .
- Content providers would like to limit in some cases access to the content
  - legal, commercial and other issues,
  - access control based on geolocation, time, service differentiation, contribution to the community (private tracker), etc.
  - private swarms
- Support both connection (TCP) and datagram (UDP) oriented protocols
- Support file transfer, VoD and live streaming
- Similar work exists, but not complete or standardized
- Secure protocols are available in IETF but designed for different application domains (IPsec, TLS, DTLS)

What the ECS protocol draft provides

- Distributed access control mechanism
  - Allows swarm initiators to decide who can access an ECS protected swarm
  - Allows specifying basic conditions to access the swarm (rules)
  - Allows protecting invested resources in the content distribution
- Secures the communication among peers
- Needed elements and mechanisms for motivation scenarios implementation
- Specification how to use the protocol with PPSPP (UDP encapsulation)
- Slim but extensible

The ECS protocol is not a DRM solution

- Controls access to distribution channel and peer resources, and not content
- Content is distributed as is, without modifications
- Every swarm initiator can choose at their will how to control access to the swarm
- Could be used with proprietary DRM solutions but the draft doesn't specify how this can be done

How it works

- A swarm initiator needs to create a swarm certificate
- Peer willing to join the swarm needs to provide his/her public key to the initiator
    - no central authority needed (peer key pair could be generated per swarm)
- The swarm initiator provides an authorization credential (PoA) to interested peers
- Peers by themselves control access to the swarm
- Relation with the peer in association ends when the content parts needed are transfered or the protocol safe operation conditions are met
- New connection can be made, if needed

Protocol in brief

- Two parts, the first runs once when two peers connect, the other for the rest of communication

(1) Handshake
  - 4 messages
  - mutual authenticatication of the connected peers via digital signature mechanism
  - mutual authorization based on peers authorization credentials (PoAs)

(2) Application data communication protection
  - based on cryptographic material and data exchanged in the handshake
  - reuses a subset of TLS mechanisms - AEAD interface and key establishment
  - provides data authentication, integrity and confidentiality service

Swarm certificate and PoA

- Digitally signed statements:
  - Authorization credential or PoA:
    - Rules specify conditions under which the peer can join the swarm

$$SW_{id}, K_s, K_h, ET, [Rules,]\{SW_{id}, K_s, K_h, ET[, Rules]\}_{K_{s}-1}$$

  - Swarm certificate:
    - Security parameters (SP) specify what key, signature and protection mechanisms types are used in swarm

$$SW_{id}, SP, \{SW_{id}, SP\}_{K_{s}-1}$$

SW_{id} - Swarm ID	K_s - swarm key	[] - optional
ET - expiry time	K_h - peer key	$\{\}_{K_{s}-1}$ - signed
$Rules$ - access control rules	SP - swarm security parameters	

The protocol

$$A \rightarrow B: \quad SW_{id}, V_a, N_a \tag{1}$$

$$A \leftarrow B: \quad SW_{id}, V_b, N_b \tag{2}$$

$$A \rightarrow B: \quad PoA_a, [RS_a,]\{N_a, N_b, PoA_a[, RS_a]\}_{K_{a-1}} \tag{3}$$

$$A \leftarrow B: \quad PoA_b, [RS_b,][\{S_{ab}\}_{K_a},]\{N_a, N_b, PoA_b[, RS_b][, \{S_{ab}\}_{K_a}]\}_{K_{b-1}} \tag{4}$$

[on success]

$$A \leftrightarrow B: \quad \textit{application data communication protection} \tag{5}$$

[on failure]

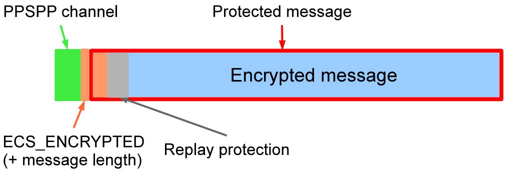
$$A \leftarrow B: \quad PoA_b, I_b, \{N_A, N_b, PoA_b, I_b\}_{K_{b-1}} \tag{6}$$

$$A \rightarrow B: \quad PoA_a, I_a, \{N_a, N_b, PoA_a, I_a\}_{K_{a-1}} \tag{7}$$

SW_{id} - Swarm ID	$V_{a\|b}$ - version	$[]$ - optional
$PoA_{a\|b}$ - authorization credential	$N_{a\|b}$ - nonces	$\{\}_{K_{a\|b-1}}$ - signed
$RS_{a\|b}$ - requested service	S_{ab} - secret	$\{\}_{K_a}$ - encrypted
$K_{a\|b}$ - peer A, B private key	$I_{a\|b}$ - error information	

Using ECS protocol with PPSPP

- The draft specifies ECS UDP encapsulation
- Two new PPSPP message types are defined:
  - ECS_PROTOCOL: enables exchange of ECS handshake messages
  - ECS_ENCRYPTED: enables security services protected application data communication
- ECS_ENCRYPTED message example is visualized below
  - Protection overhead from 24 to 68 bytes, depending on protection algorithm used



Implementation experience

- ECS protocol has been designed and implemented in context of P2P-Next project
- Implementation done in Python and is currently integrated with BitTorrent based NextShare platform (supports file download, VoD and live streaming)
- Expirementaly tested in PlanetLab:
  - protocol tested in batches on swarms up to 400 peers
  - test cases on file download and live streaming (normal RTV Slovenia DVB-T streams)
  - evaluation metrics: pre-buffering time, continuity index, signal delay, sharing ratio and index
- No real performance or scalabilty issues found, metric results are almost on pair for closed and non closed cases

Meeting PPSP requirements, part I

- PPSP.SEC.REQ-1: PPSP MUST support closed swarms, where the peers are authenticated or in a private
  - The ECS protocol provides needed mechanisms and solutions.
- PPSP.SEC.REQ-2: Confidentiality of the streaming content in PPSP SHOULD be supported.
  - confidentiality of streaming content not an issue, confidentiality of content distribution provided as well
- PPSP.SEC.REQ-3: PPSP SHOULD support identifying badly behaving peers, and exclude or reject them from the P2P streaming system.
  - The ECS protocol enables tighter control who can join the swarm but provides no means to revoke authorizations. Expiry time could be used for this purpose. The protocol draft documents expected threats regarding the requirement and suggests possible mitigation strategies.

Meeting PPSP requirements, part II

- PPSP.SEC.REQ-4: Integrity of the streaming content in PPSP MUST be supported to provide a peer with the possibility to identify unauthentic content (undesirable modified by other entities rather than its genuine source).
  - The ECS protocol prevents modifying the content in transit but does not protect against malicious insiders
- PPSP.SEC.REQ-5: The security mechanisms in PPSP, such as key management and checksum distribution SHOULD scale well in P2P streaming systems.
  - There is no central component in content distribution, testing shows no real scaling or performance issues
- PPSP.SEC.REQ-6: Existing P2P security mechanisms SHOULD be re-used as much as possible in PPSP, to avoid developing new security mechanisms.
  - No real world open specification similar to the ECS protocol is available. The protocol reuses as much as possible existing security protocols and solutions (TLS)