AuthCodeSEC

extending EPP with
public key authentication to
make domain transfer faster and safer

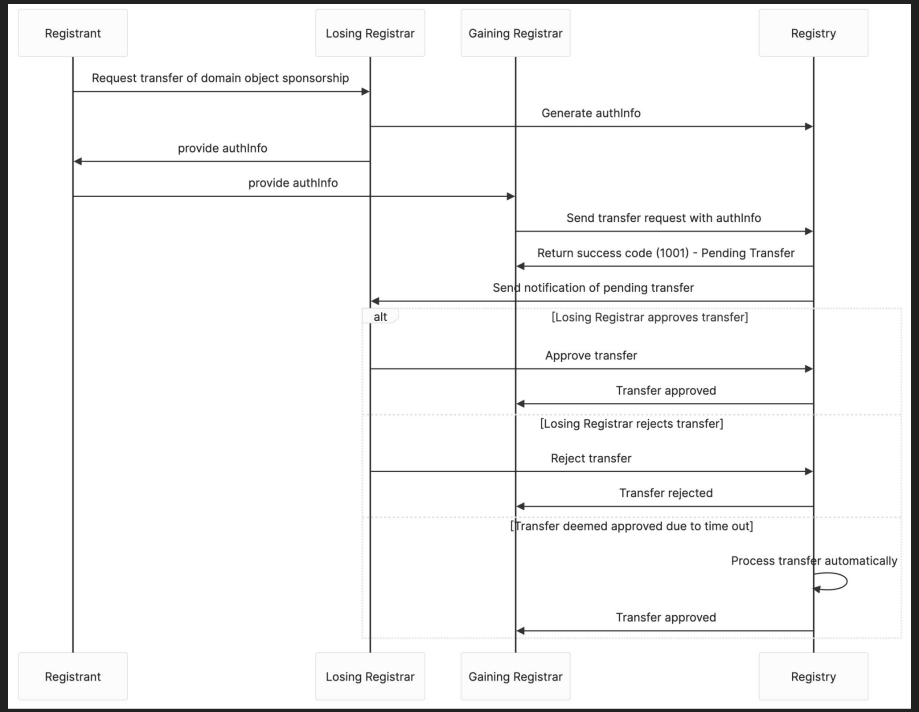
Victor Zhou, Namefi.io

Table of Contents

- Current mechanism of AuthCode
- Why do we want to upgrade it
- Previous efforts
- The proposed solution: AuthCodeSEC
- The backward compatibility and roadmap to an incremental upgrade
- Design discussions for the WG for the first iteration of draft

Current mechanism of AuthCode: a “Shared Password”

- Issued by Losing Registrar
- Stored by Registry
- Given to Registrant
- Submitted by Gaining Registrar
- Slow auto approval or when LR explicitly approves it



Note to the diagram

- authInfos are often pre-generated at the time of creation or update too
- Losing registrar can

Why do we want to upgrade it: “shared password”

- Too wide of a attack surface
  - Rely on authenticity, secrecy and availability in easy segment of the communications across 5 parties (sometimes 7 parties with resellers)
  - Rely on in-time update of secrecy when expire
  - Rely on secrecy and availability in storage in many parties
  - Also vulnerable to attack from insider, or conflict of interest
- Doesn’t specify the transfer-ee
  - double-transfer attempt
- Verifying requires Registry or Registry+Losing Registar
  - Vulnerable to brute-force attack or replay attack
  - Making other parties’s authenticity and reputation vulnerable

Previous efforts

- RFC-8807: help improve the authenticity between EPP server and client
- RFC-9154: use (crypto-safe) Hash for AuthCode, and a TTL
  - Help reduce attack surface
  - Help reduce vulnerability of Registry leaking

The AuthCodeSEC

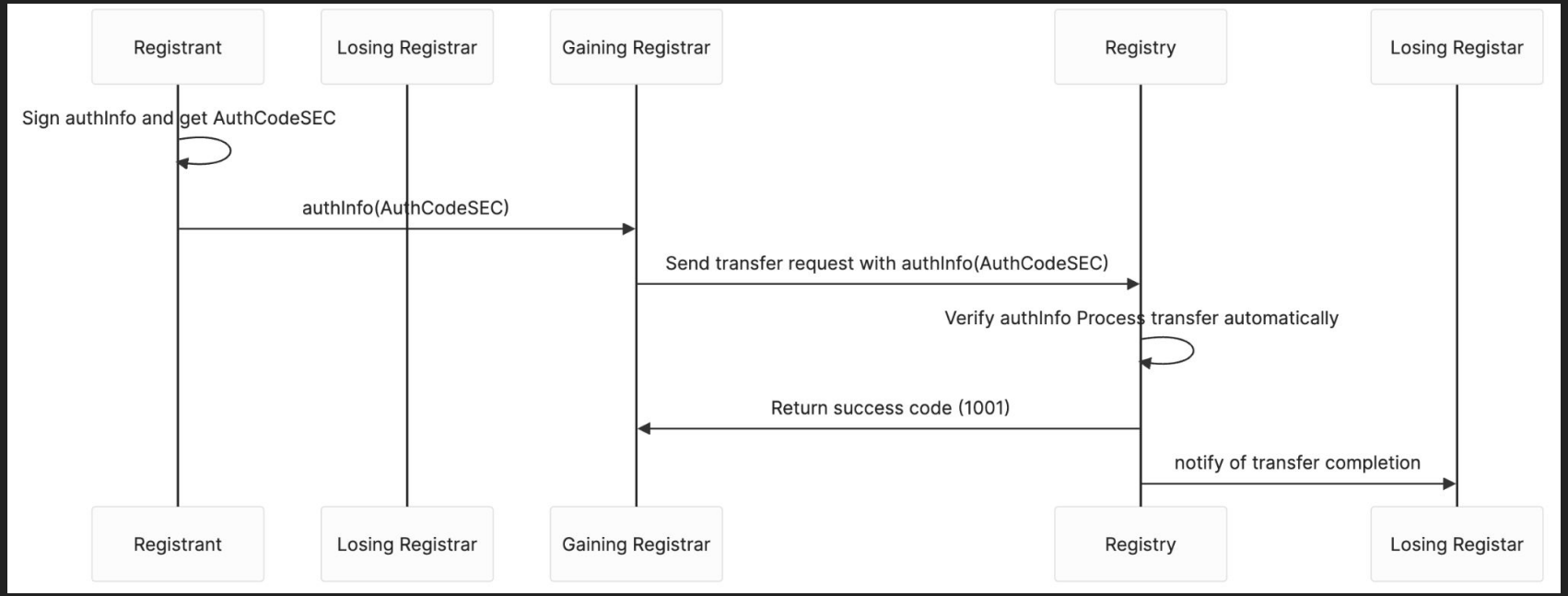
- further improves the security upon existing effort by
  - Reduces reliance on secrecy and authenticity of transmission and storage in many channels
  - Allow specifying receiver to avoid double transferring
  - Adds reply attack prevention

The proposed solution: AuthCodeSEC

A public-key authentication mechanism that

- EPP <extension> field
  - VersionInfo
  - Nonce (or reply-prevention info)
  - ReceiverInfo (who should be the future approver, is it registrar or registrant?)
  - ApproverInfo (current approver with its Public Key Identifier)
  - DigestInfo the digest algo version and the digest value by hashing the structural data
- And sign with public key as “sig” to fit in <domain:pw>
- (Design Question: should we use domain:pw or domain:ext?)

The proposed solution: AuthCodeSEC diagram



Model A: registrant approval model

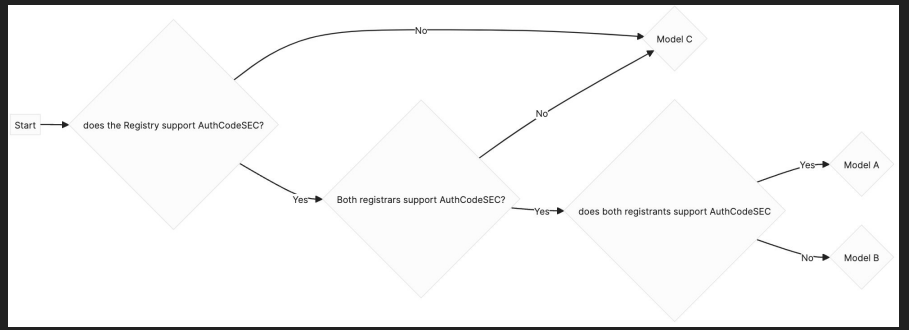
Roadmap to an incremental upgrade

Model A: registrant sign

Model B: registrar sign on behalf of registrants

Model C: current model

- Losing Registra still submit the AuthCodeSEC to Registry to be stored.
- Registry then verify AuthCode by comparing the submission by GR
- Same as current

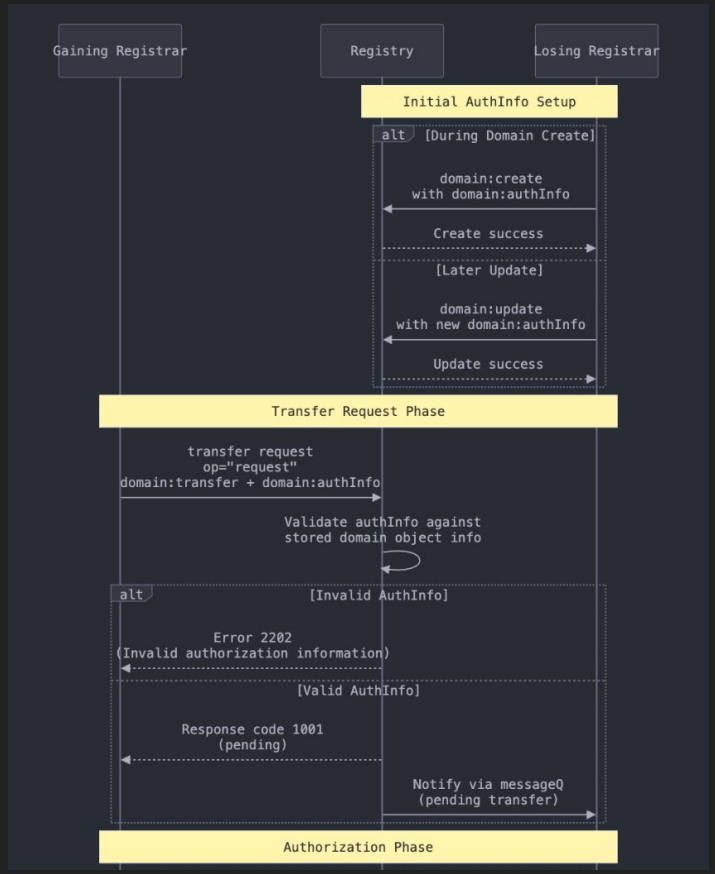


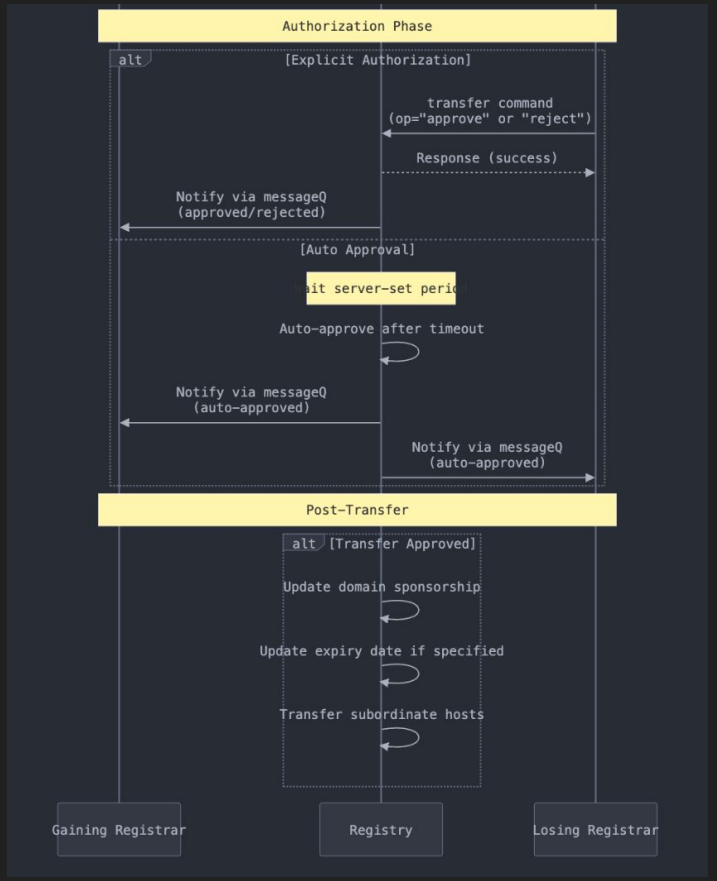
Design discussions for the WG for the first iteration of draft

- Any concern to the proposal of using domain:pw instead of domain:ext for this purpose?
- What are other things we need to include in the digtest for hashing?
- How to make the AuthCodeSEC extendable for future?
- Not technical, but a use-case question based on the practice of ICANN policy
  - Who should enforce rejection? Can it be moved to Gaining Registrar / Registry instead of Losing Registrar? E.g. ICANN Transfer Policy Section 3.8 when registrar MUST deny transfer

Thank you!
Questions and comments please
Send to zzn@namefi.io or

https://github.com/xinbenlv/rfc-draft-authcodesec


Gaining Registrar
Registry
Losing Registrar
Initial AuthInfo Setup
alt
[During Domain Create]
domain:create
with domain:authInfo
Create success
[Later Update]
domain:update
with new domain:authInfo
Update success
Transfer Request Phase
transfer request
op="request"
domain:transfer + domain:authInfo
Validate authInfo against
stored domain object info
alt
[Invalid AuthInfo]
Error 2202
(Invalid authorization information)
[Valid AuthInfo]
Response code 1001
(pending)
Notify via messageQ
(pending transfer)
Authorization Phase


Authorization Phase
alt
[Explicit Authorization]
transfer command
(op="approve" or "reject")
Response (success)
Notify via messageQ
(approved/rejected)
[Auto Approval]
it server-set perio
Auto-approve after timeout
Notify via messageQ
(auto-approved)
Notify via messageQ
(auto-approved)
Post-Transfer
alt
[Transfer Approved]
Update domain sponsorship
Update expiry date if specified
Transfer subordinate hosts
Gaining Registrar
Registry
Losing Registrar