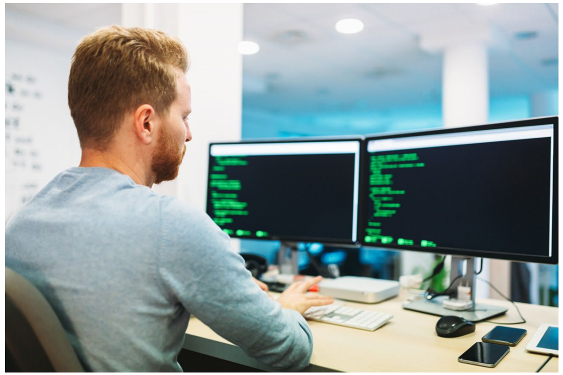
The Usage Models and Risks of STIR/SHAKEN, seen from the Pragmatism of an Implementation

Bogdan-Andrei Iancu

- December 5th, 2019 -

The challenge : STIR/SHAKEN implementation



Standards

Implementation

The Challenger: OpenSIPS

OpenSIPS is a well known, versatile SIP Server

- Highly customizable / programmable
- Feature rich (155 modules)
- High throughput

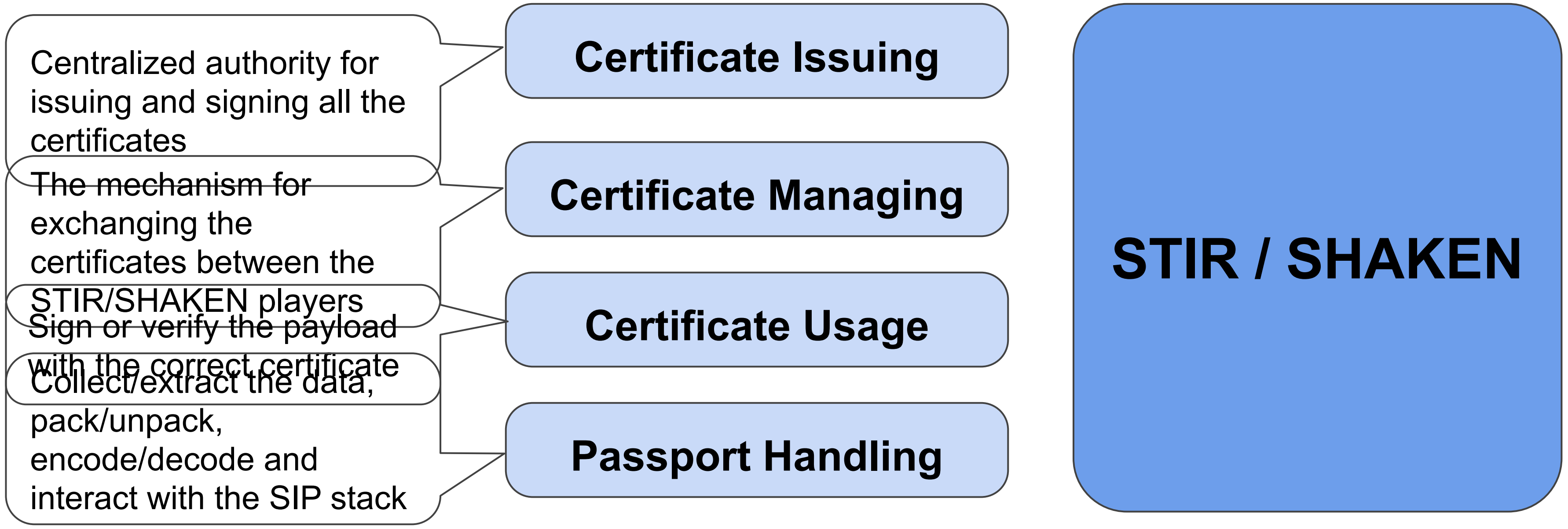
The Reason: OpenSIPS versatility

OpenSIPS implements various SIP components, where STIR/SHAKEN may be needed

	Authenticate	Verify	Inspect
Class 4&5 Switches	✓	✓	✓
SBC		✓	✓
Carrier LB/FrontEnd	✓	✓	✓
Trunking		✓	✓

The Approach: Divide et Impera

`Or let's do some breakdown of the "big STIR/SHAKEN picture"`



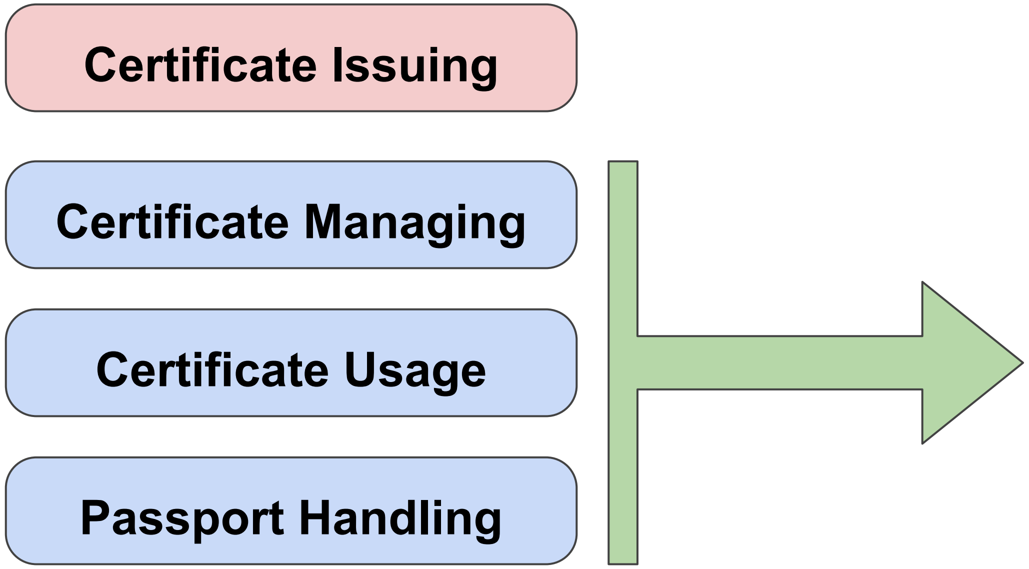
Centralized authority for issuing and signing all the certificates

The mechanism for exchanging the certificates between the STIR/SHAKEN players

Sign or verify the payload with the correct certificate

Collect/extract the data, pack/unpack, encode/decode and interact with the SIP stack

The Result



OpenSIPS

Implementation

Usage Models

Isolate the Uncertainties...

The Certificate Managing is the unclear part:

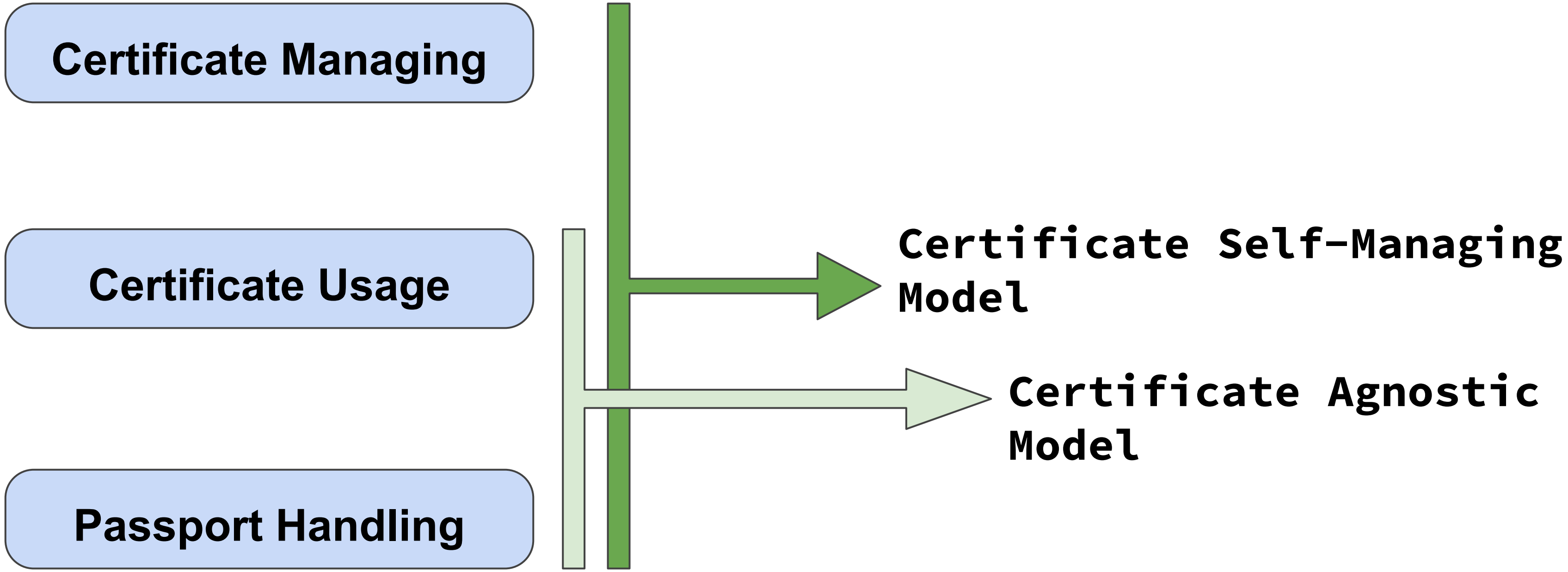
- Will the certificates by identified by HTTP URL?
- Will it be expected to download certificates via HTTP?
- Will the certificate exchange be done in realtime / ondemand?
- Will each operator be responsible for building the exchange infrastructure ?

… and Secure the Certainties

The current level of standardization gives solid grounds for:

- Building the passport
- Signing / Verifying the passport
- SIP handling

Usage Models



Usage Models

- **Certificate agnostic** (or external handling) - other sub-systems in the platform/service are responsible for providing the required certificate for each call;
- **Certificate self managing** - OpenSIPS is performing the certificate managing also, via its own mechanisms of fetching and storing the required certificates.

The Agnostic Model

The implementation is not aware of how the Certificates are managed

- The certificates are in local storage (like DB or files)
- There is a predefined mapping between operators and their certificates
- Static, pre-operational exchanged, nothing realtime
- Off-band exchange

The Self-Managing Model

OpenSIPS takes care of the Certificate Managing:

- Upon Authentication: based on calling number, identify the proper certificate to use (through its own certificate repository)
- Upon Verification: OpenSIPS fetches the certificate by itself and implements its own caching mechanism

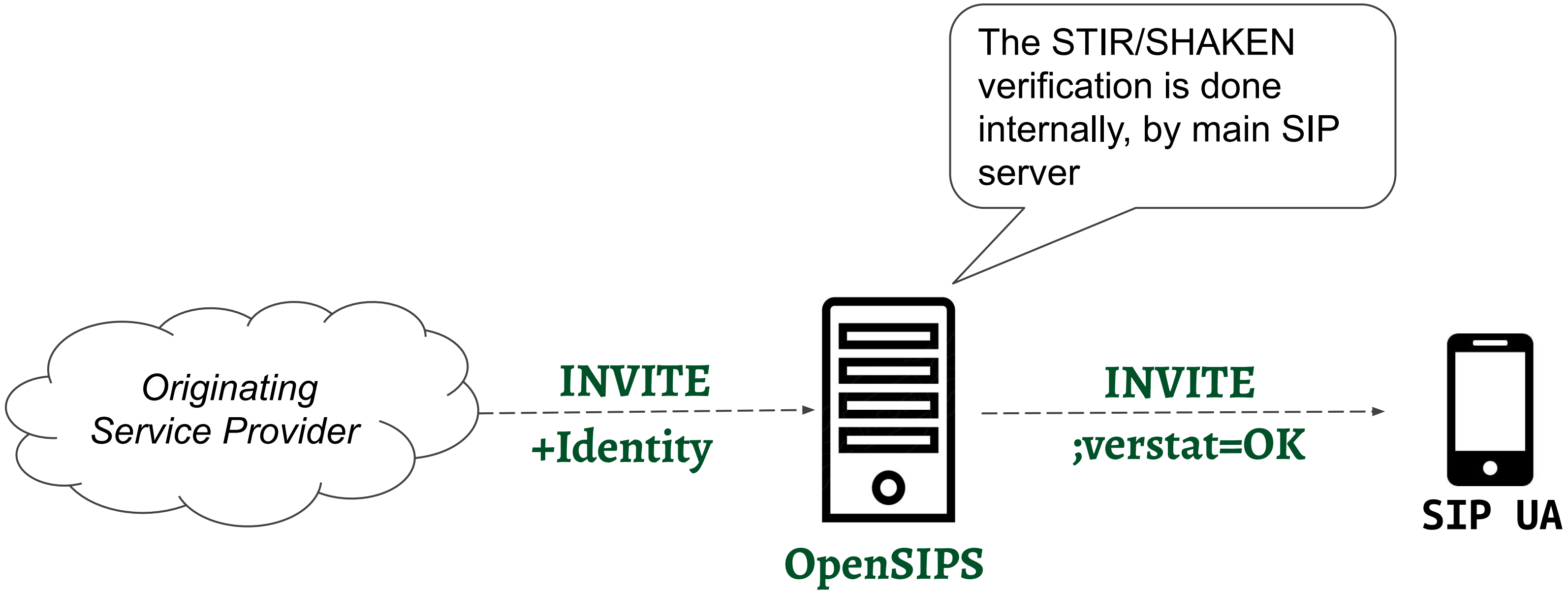
Implementation details

- The agnostic model is provided by a new “stir_shaken” module in OpenSIPS
- The self-managing part is just OpenSIPS scripting, to fetch certificates via HTTP(s) (using the “rest_client” module) and perform local caching (using the “cachedb_local” module)

STIR_SHAKEN module

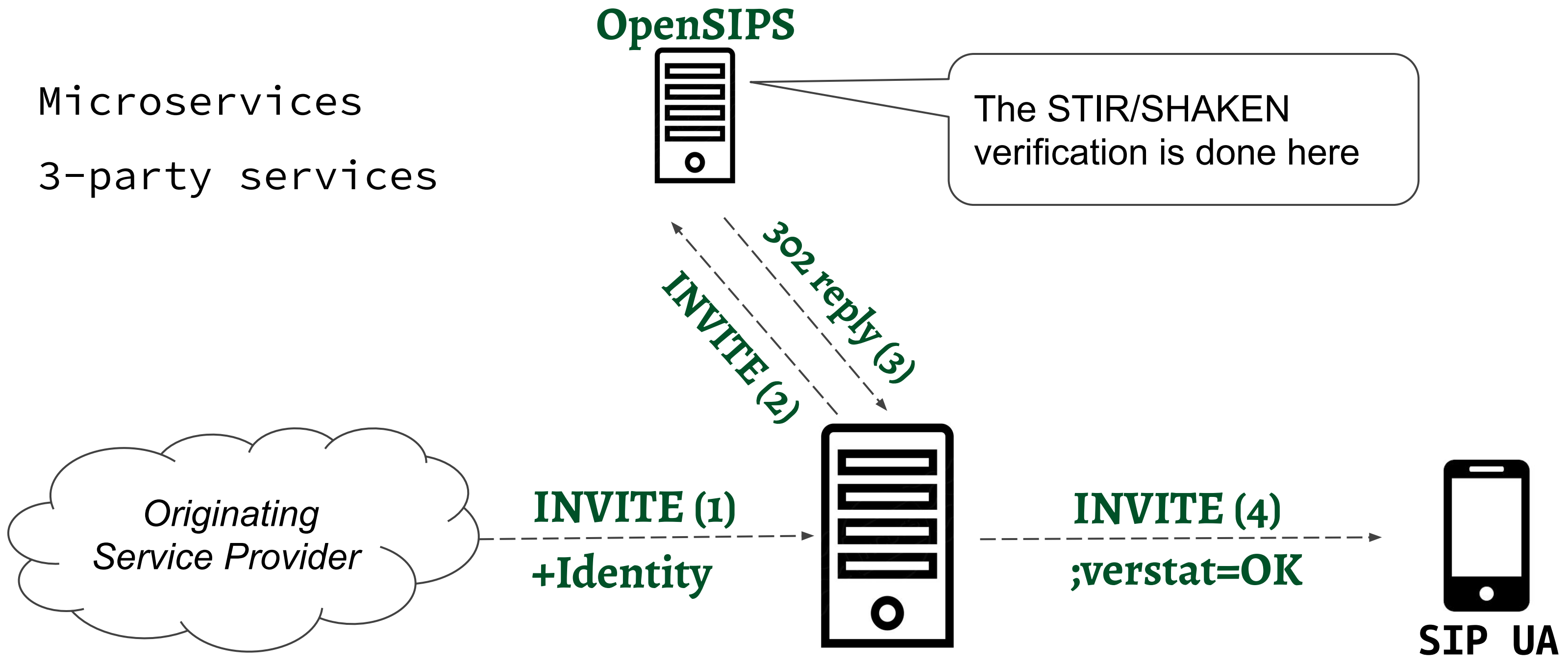
- developed by Vlad Pătrașcu
- is **public** and **open-source**
- support for attaching Identity headers:
  - directly to an INVITE
  - as a 302 redirect

Integration Models - built-in


The STIR/SHAKEN verification is done internally, by main SIP server
Originating Service Provider
INVITE
+Identity
OpenSIPS
INVITE
;verstat=OK
SIP UA

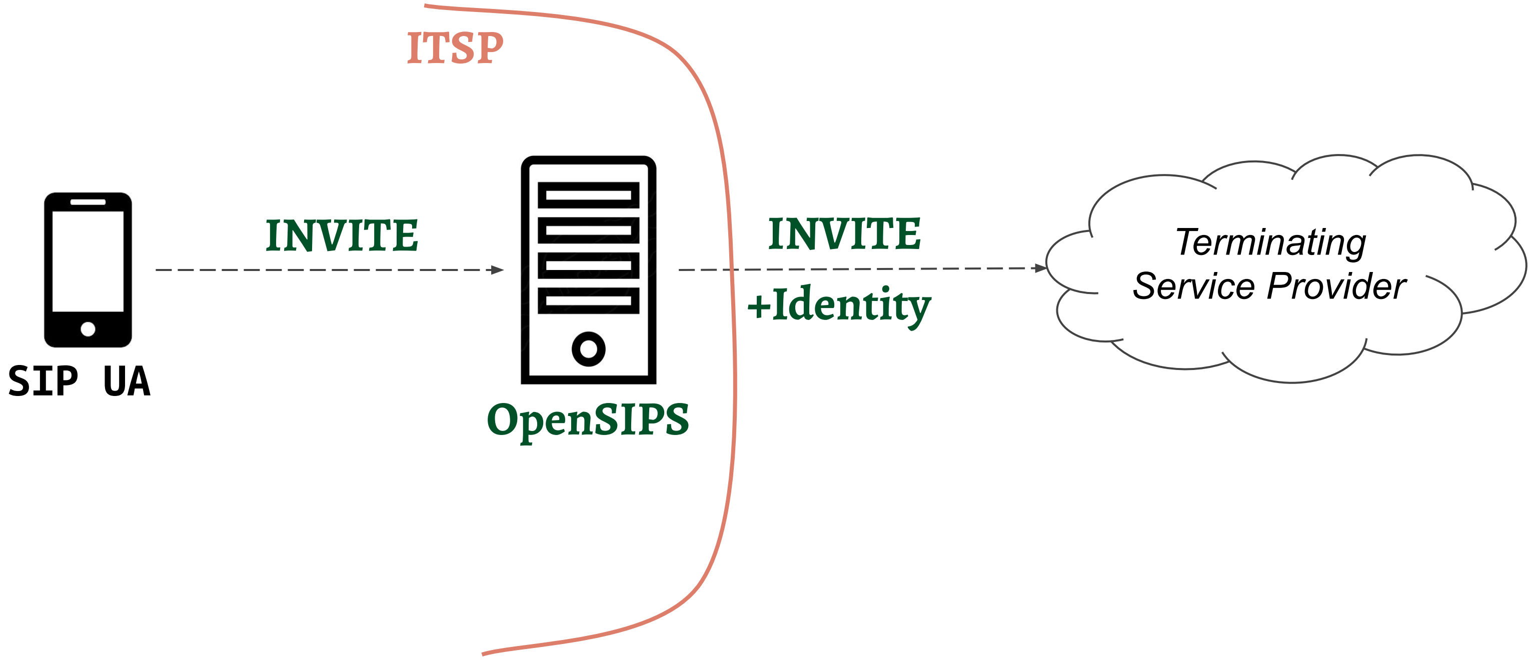
Integration Models - external

- Microservices
- 3-party services



Usage Samples

Authorization



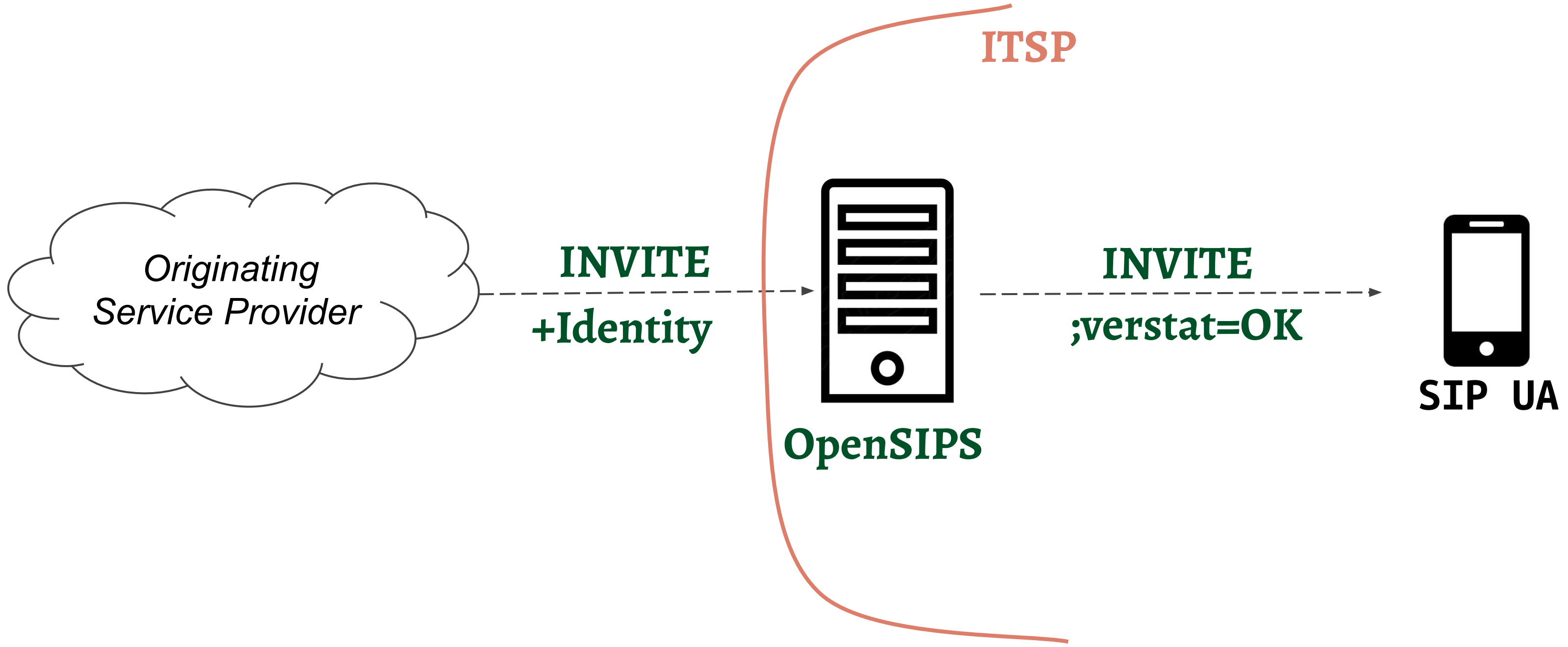
Authorization : opensips.cfg

```
loadmodule "stir_shaken.so"




$var(rc) = stir_shaken_auth("A", "$var(oid)", "$var(cert)", "$var(pkey)",
"https://cert.example.org/passport.cer"[,”$var(orig)”,”$var(dest)”]);
if ($var(rc) < 0) {
    xlog(“stir_shaken_auth() failed with $var(rc)\n”);
    send_reply(500, “Server Internal Error”);
    exit;
}
```

Verification



Verification: opensips.cfg

```
loadmodule "stir_shaken.so"
modparam("stir_shaken", "ca_list", "/etc/pki/opensips/passport.cer")



$var(rc) = stir_shaken_verify("$var(cert)", $var(code), $var(reason));
if ($var(rc) < 0) {
    xlog("stir_shaken_verify() failed: $var(rc), $var(code), $var(reason));
    send_reply($var(code), $var(reason));
    exit;
}
```

Inspection

$identity(header)

$identity(x5u)

$identity(payload)

$identity(attest)

$identity(origid)

$identity(dest)

$identity(orig)

$identity(iat)

The Risks

Assessment

SIP specific risks:

- Network risks

STIR/SHAKEN specific risks, derived from the Certificate Managing side (valid only for the verification part):

- Performance risks
- Security risks

Network Risks - UDP

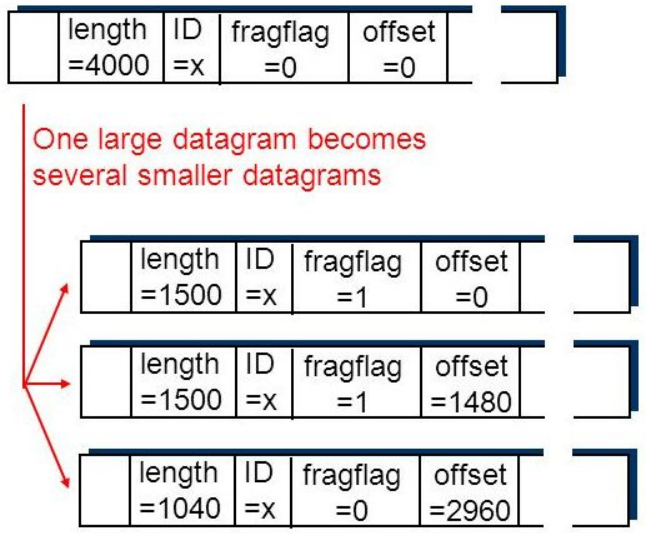
The passport is quite large leading to large SIP packages.

Over UDP protocol

- max 65K, but usual MTU 1.5K
- => fragmentation

Risks:

- Losing fragments on the way
- Unable to re-assemble



Network Risks - TCP

Over TCP protocol

- No limit as payload

Risks

- Bottleneck at TCP conn level
- Performance issue at OS and application layers

Performance Risks

An ***on-demand certificate download*** introduces a time penalty of 10x compared to the SIP call setup.

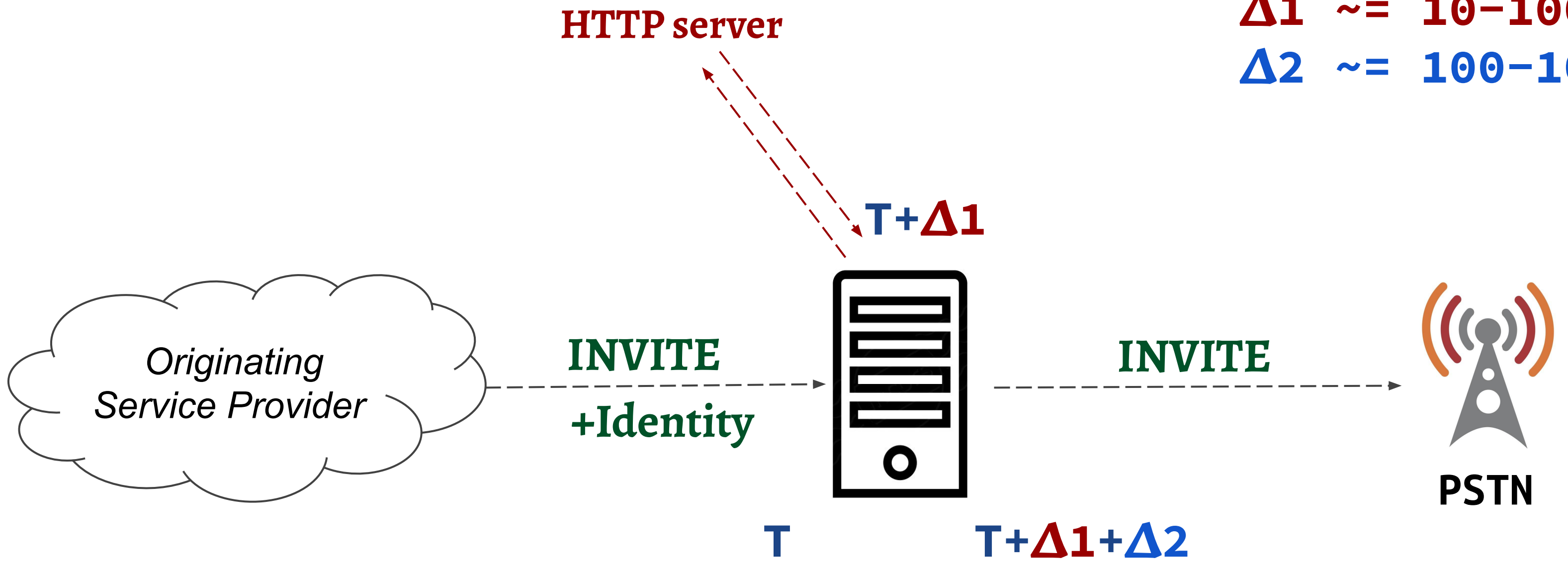
Besides the huge impact on the PDD, the problem may escalate, due the multi-processes architecture of OpenSIPS (tens of calls are handled in parallel)

Definitely there is a need to use the async support (when fetching the certificate) to avoid blocking.

PDD degradation

Δ1 ~= 10-100ms

Δ2 ~= 100-1000us



Security Risks

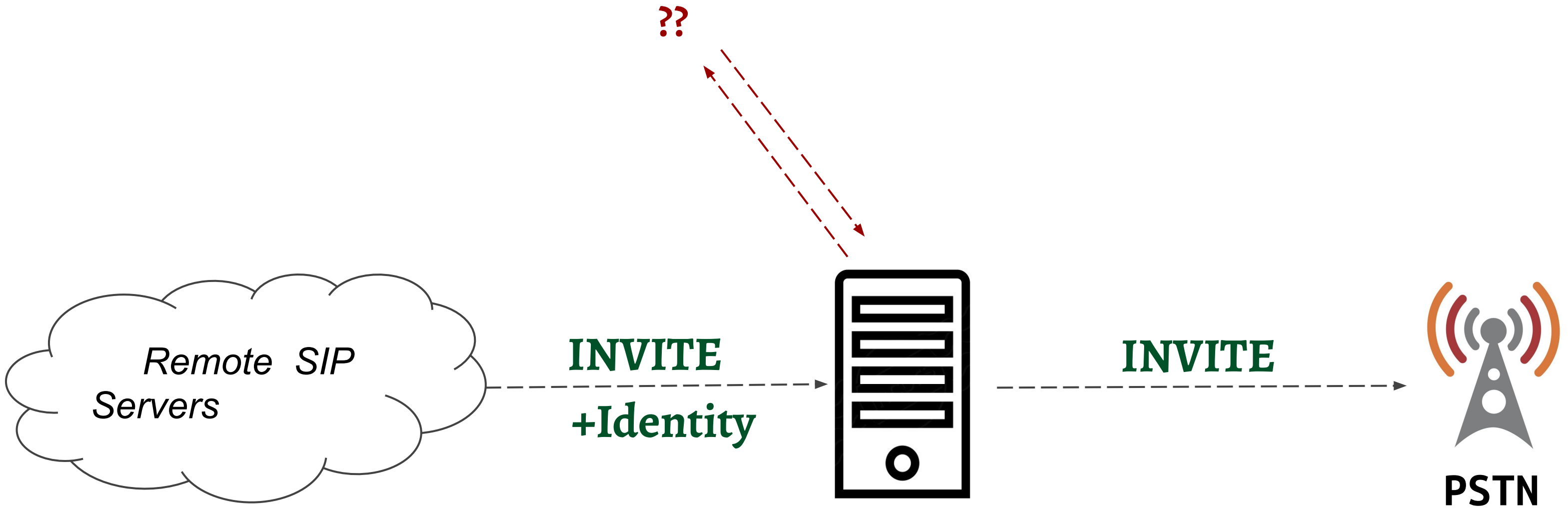
There are no ways to verify and trust the HTTP URLs provided by passports.

You need to fetch the certificate (follow the HTTP link) before validating the content 😢 .

- Malicious URL attacks (DOS)
- DDOS attacks

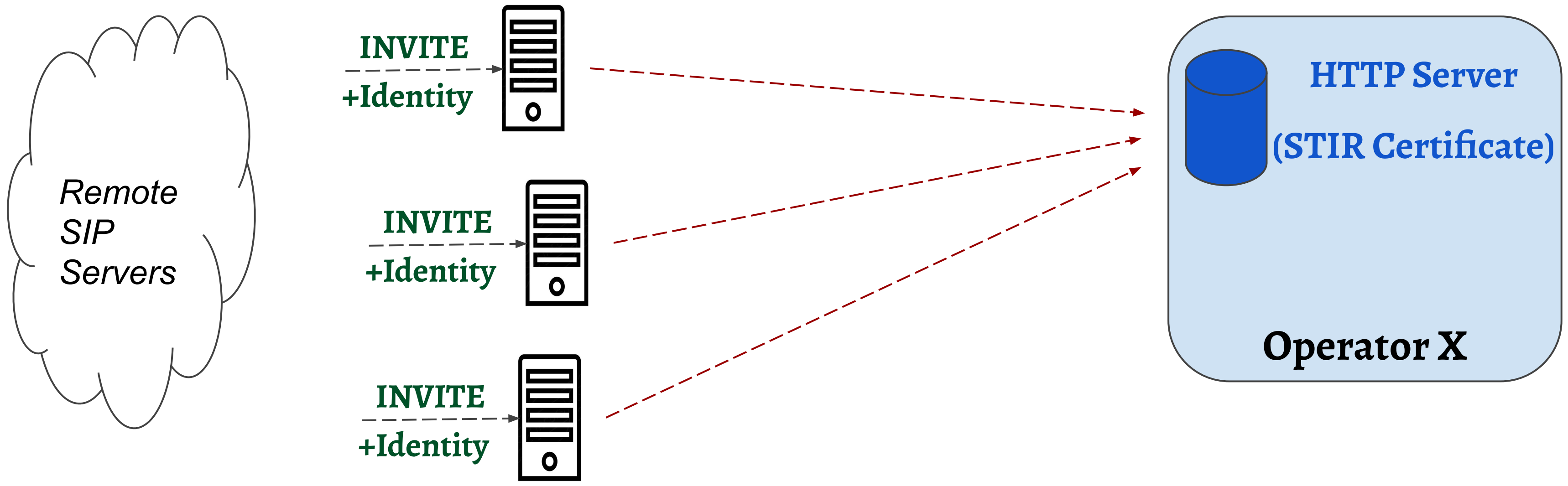
Security Risks - DOS

Malicious HTTP URLs presented in the passport



Security Risks - DDOS

Many operators are flooded with passports pointing to an attacked URL.



Learn more on what we did

Go Agnostic before things are settling down

- Bogdan Iancu
  - OpenSIPS Project: www.opensips.org
  - Email: bogdan@opensips.org