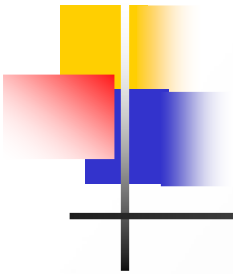


SPIIRAS

Development of Entrusting Protocol

Vasily Desnitsky, Sergey Reznik

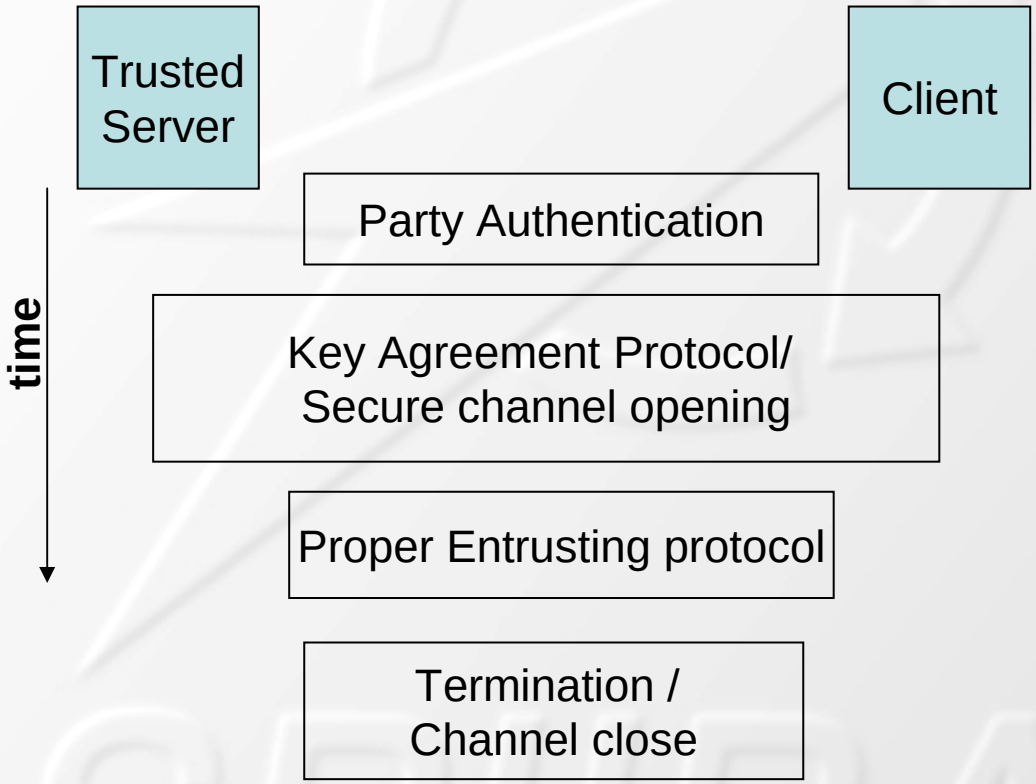
Computer Security Research Group,
St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences

Entrusting Protocol Requirements

- Authentication of parties — Authentication protocol
- Key Agreement — Key Agreement protocol
- Confidentiality
- Data Authentication
- Message Loss

  Proper Entrusting protocol
- Timeliness — Controlled on application level

Entrusting protocol structure



Party Authentication

- Trusted Server Authentication
- Client Authentication (?)

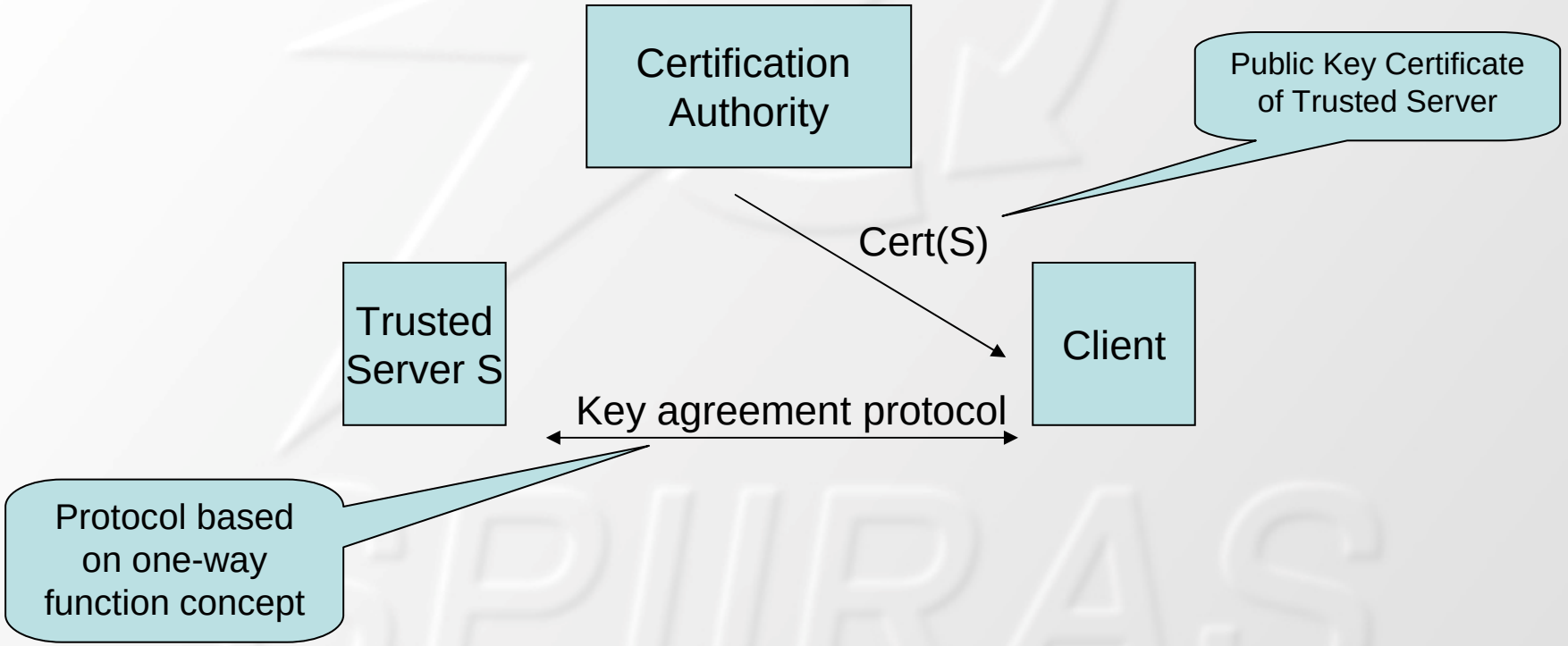
- Drawbacks of password-based authentication
  - (-) need of secure password transfer
  - (-) lack of password freshness
  - (-) need of different password for each program copy

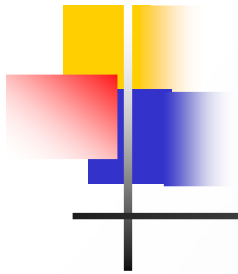
- Authentication based on Public Key Certification

Trusted Server Authentication

- Public Key Certification

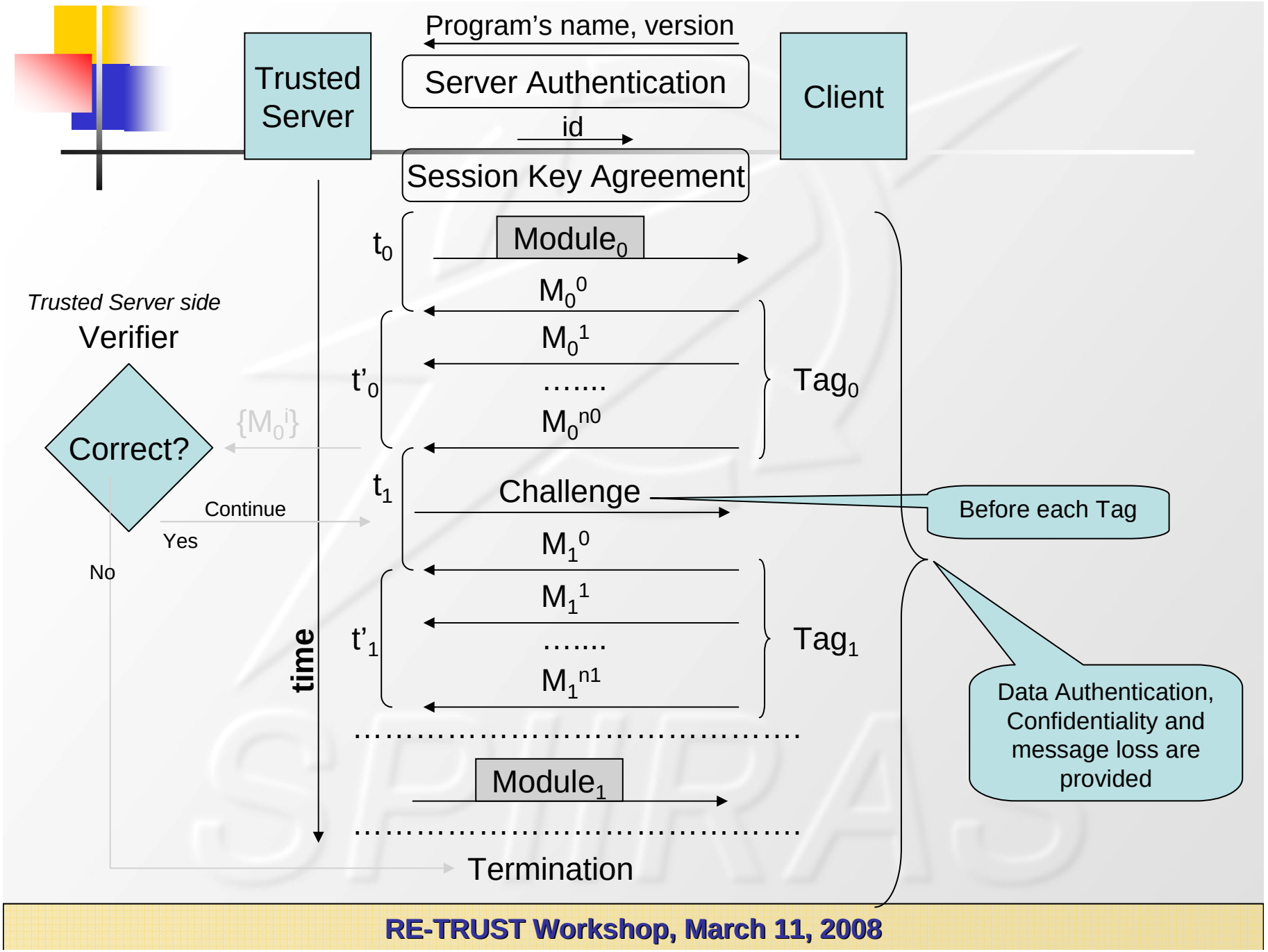
  (-) need of Public Key Certification Infrastructure

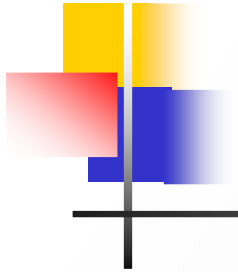




Key Agreement

- Protocol based on one-way function concept
- E.g. Diffie-Hellman principle – it subject to man-in-the-middle attack
  - Prior party authentication


Trusted Server
Client
Program's name, version
Server Authentication
id
Session Key Agreement
Module0
t0
M0^0
t'0
M0^1
.....
M0^n0
Tag0
Trusted Server side
Verifier
Correct?
{M0^i}
Continue
Yes
No
t1
Challenge
M1^0
t'1
M1^1
.....
M1^n1
Tag1
Before each Tag
Data Authentication, Confidentiality and message loss are provided
time
Module1
Termination



Entrusting protocol verification (1/3)

- *Construct target entrusting protocol with support of variety of verification tool*
- *AVISPA verification of simplified entrusting protocol*

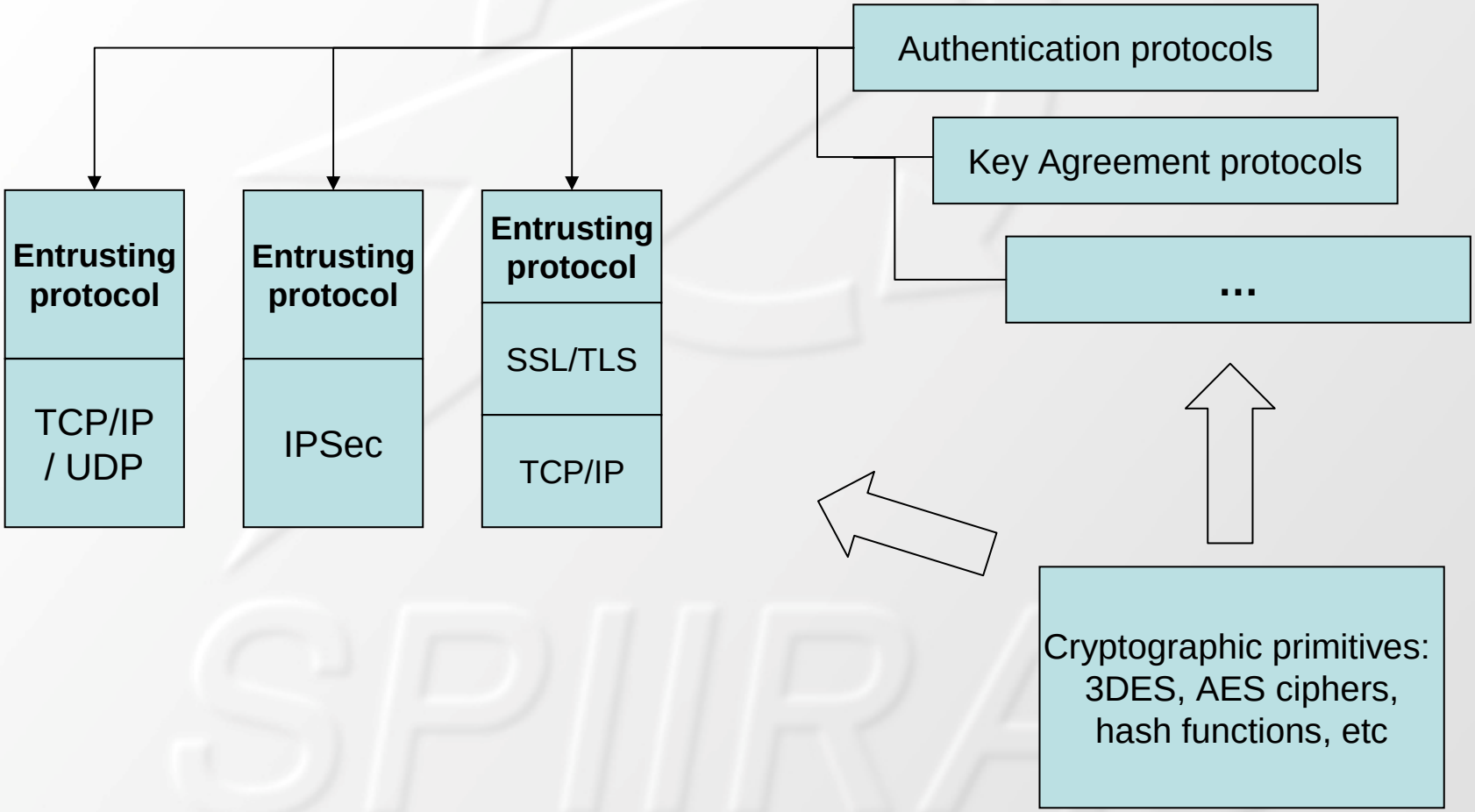
Entrusting protocol verification (2/3)

```
role server (S, C: agent, K : symmetric_key, SND, RCV: channel (dy))
played_by S def=
local State : nat,
        Module: message,
        Tag: text,

init State := 0
transition
    0.  State  = 0 /\ RCV(start) =|>
               State':= 2 /\  SND({Module}_K)
    2. State = 2 /\ RCV({Tag}_K) =|>
               State':=2  /\  Module SND:=new()  /\  ({Module}_K)
end role

role client(S, C: agent, K : symmetric_key, SND, RCV: channel (dy))
played_by C def=
local State : nat,
        Module: message,
        Tag: text,

init State := 1
transition
    1.  State  = 1  /\ RCV({Module}_K) =|>
                State':=3 /\ SND({Tag}_K) /\ secret(Tag, p, {S,C})
    3. State = 3 /\ RCV({Module}_K) =|>
                State':=3 /\  Tag:=new()  /\  SND({Tag}_K) /\ secret(Tag, p, {S,C})
end role
```

Entrusting protocol verification (3/3)

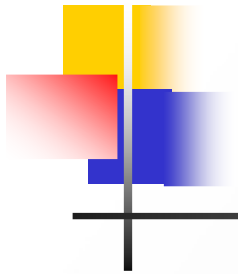
```
role session(S, C: agent, K : symmetric_key) def=
local SS, RS, SC, RC: channel (dy)
composition
    server(S,C, K,SS,RS) /\ client  (S,C,K,SC, RC)
end role

role environment() def=
const
        p : protocol_id,
        ksc,ksi,kic : symmetric_key,
        s,c : agent

intruder_knowledge = {s,c,ksi,kic}

composition
             session(s,c,ksc)
        /\  session(s,i,ksi)
        /\  session(i,c,kic)
end role

goal
  secrecy_of p
end goal

environment()
```

Proposals for prototype implementation



Prior proposal analysis

- IPSec
  - Protection on network layer
  - Too much flexibility and hence complexity
  - [Ferguson, Schneier] : advice to use *tunnel mode / ESP* only
  - Encryption and authentication order in IPSec implementations
    - Horton principle – “the protocol should authenticate what was meant, not what was said”
- SSL/TLS
  - Protection on application layer
  - Parties authentication based on X.509 certificates



Conclusion

- Future work
  - Specification of entrusting protocol model
  - Entrusting protocol verification using variety of verification tools
  - Analysis of integration of entrusting protocol with existing Internet security protocols