black hat USA 2019

AUGUST 3-8, 2019
MANDALAY BAY / LAS VEGAS

Rogue7

Rogue Engineering Station Attacks on Simatic S7 PLCs

Eli Biham. ***Sara Bitan***, Aviad Carmel, Alon Dankner, ***Uriel Malin***, Avishai Wool

Technion – Israel Institute of Technology Tel-Aviv University

Who Are We?

Dr. Sara Bitan

- Senior researcher at the Technion Hiroshi Fujiwara Cyber Security Research Center
- Founder and CEO of CyCloak - Secure System Design and Audit

Uriel Malin

- MSC Student at Tel Aviv University, advised by Prof. Avishai Wool
- Security Researcher at Medigate – Healthcare IoT Security

Talk Topics

- **Uncovered design vulnerabilities in the S7 protocol**
- **An exploit that performs remote stealth programming of an S7-1500 PLC**

The Operator

The Engineer

The Attacker

Industrial Control System

- A distributed computerized system
- Operates and monitors physical devices
- Controls critical infrastructure
  - Power plants
  - Water facilities
  - Transportation systems
  - Chemical plants

Programmable Logic Controller

- PLC - The core of the ICS
- Connected to sensors and active devices
- Runs a control program that periodically samples the sensors and triggers the devices accordingly
- A bridge between the virtual and the kinetic worlds
- The target of our attacks

PLC interfaces

HMI - SCADA

WinCC

TIA

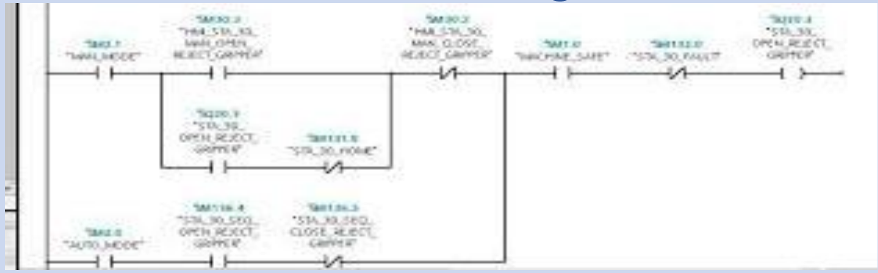
Engineering Workstation

STEP 7

The S7 Protocol

The S7 Protocol

The Control Program



Web Server

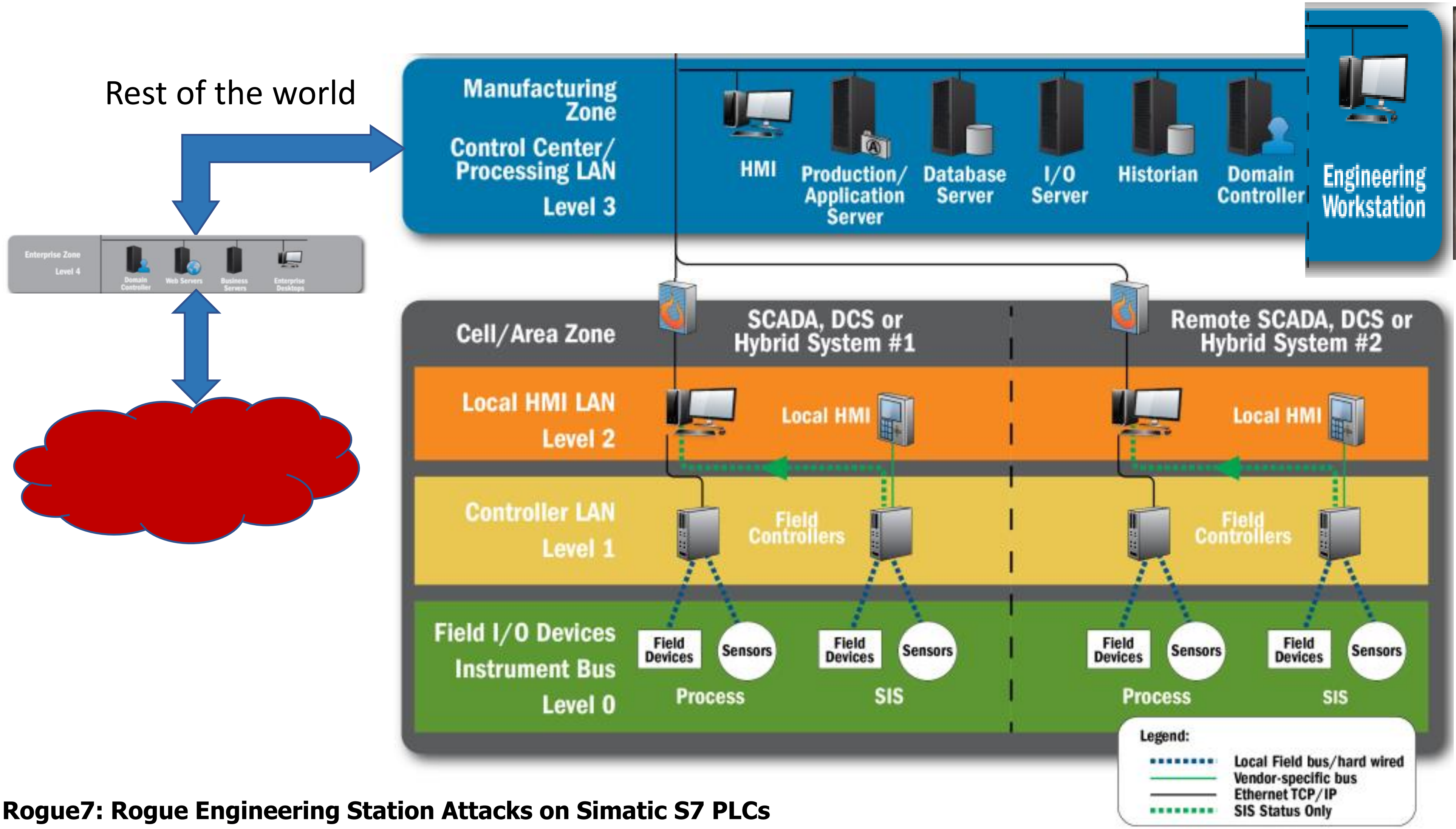
SNMP Server

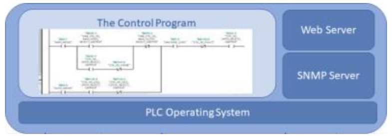
PLC Operating System

PLC

S7-1500

Secure ICS Topology

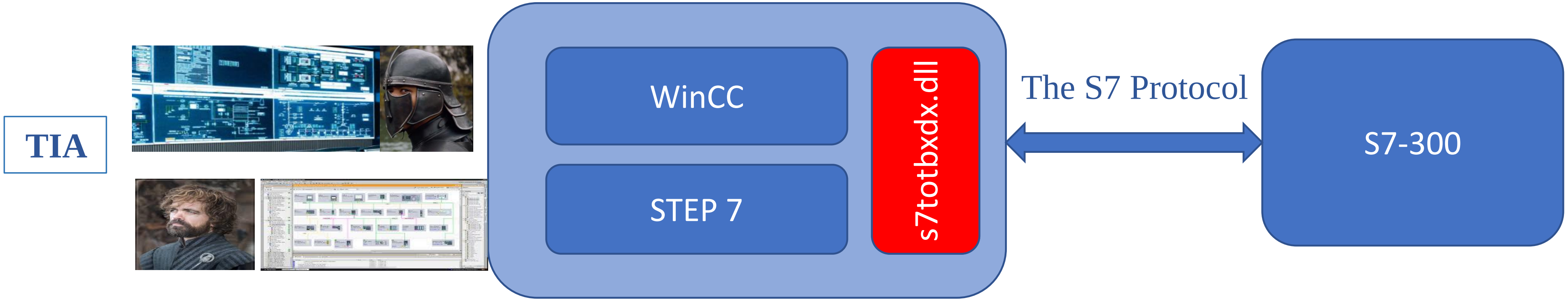




Source: NCCIC IR-18-214

Stuxnet Malware (9/2010)

- The most famous cyber-attack on ICS
- Targeted Siemens S7-300 PLC
- Infected both WinCC and Step 7 packages



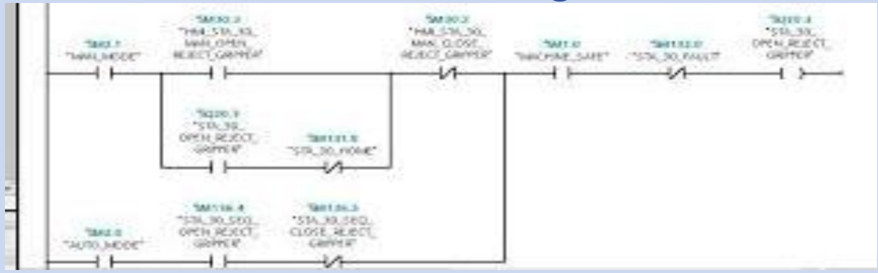
TIA as a Soft Belly

- Typically attacks are exploiting vulnerabilities the engineering station:
  - CVE-2012-3015 : untrusted search path vulnerability in Siemens SIMATIC STEP7 v5.5– July-26-2012
  - CVE-2019-10915: authentication bypass in TIA v15.1 –July-11-19 by Tenable Security

Our Attack

- Exploits vulnerabilities in the PLC Operating System
  - S7 protocol
- Any vulnerable station/ device in the network can serve as an attack machine

The Control Program



Web Server

SNMP Server

The S7 Protocol

PLC Operating System

The S7-1500 PLC

- One of two new members in the SIMATIC PLCs product line
  - S7-1500 is the high-end PLC
  - The other is S7-1200

S7-1500 Security Enhancements

- New version of the S7 protocol
  - Integrity protection of the messages
- Know-how and copy protection
- PLC access control
  - Using passwords
  - Mitigates the program download attacks
  - Not used by customers in practice

Protection level	Access			Access permission	
Protection	HMI	Read	Write	Password	Confirmation
Full access (no protection)	✓	✓	✓		
Read access	✓	✓			
HMI access	✓				
No access (complete protection)					

The S7 Protocol

Session oriented. Session begin with a 4-ways handshake

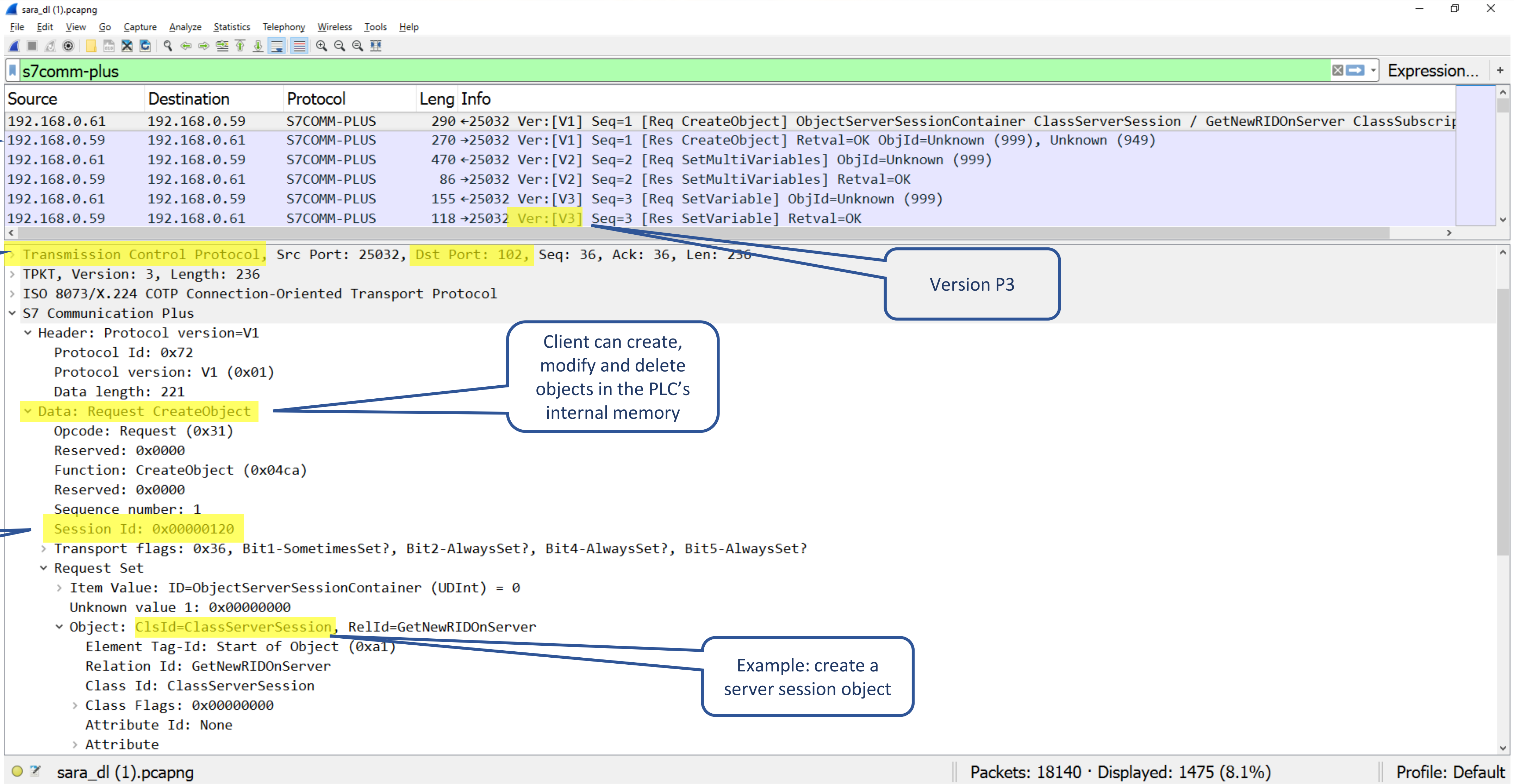
ISO transport over TCP

Version P3

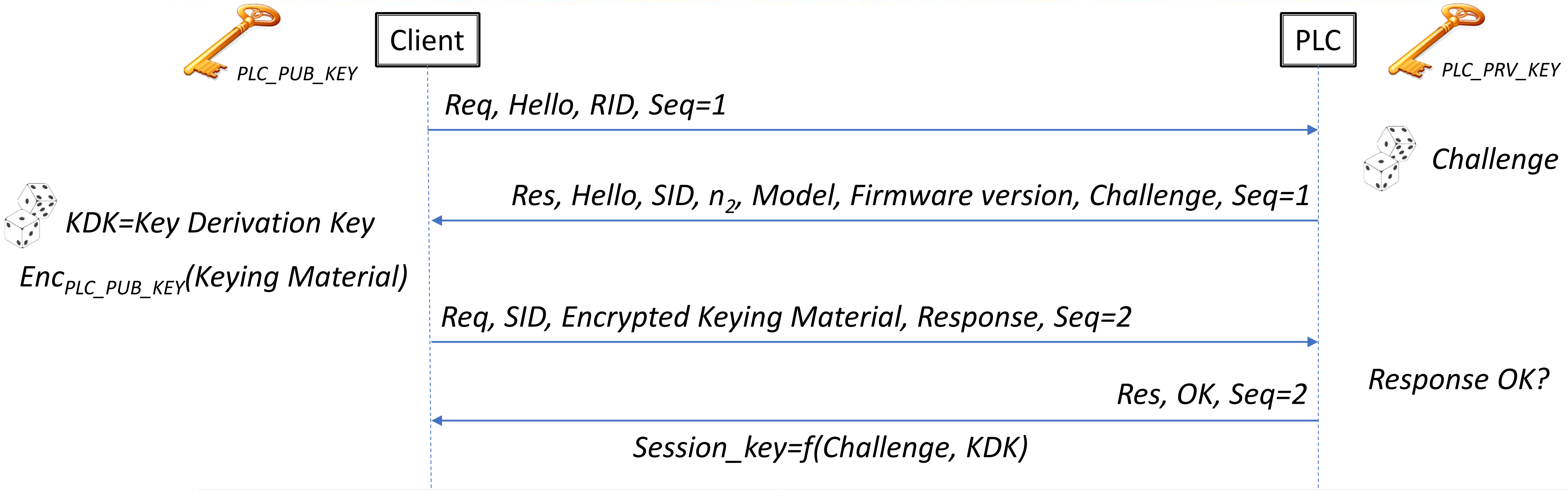
Client can create, modify and delete objects in the PLC's internal memory

Session ID

Example: create a server session object

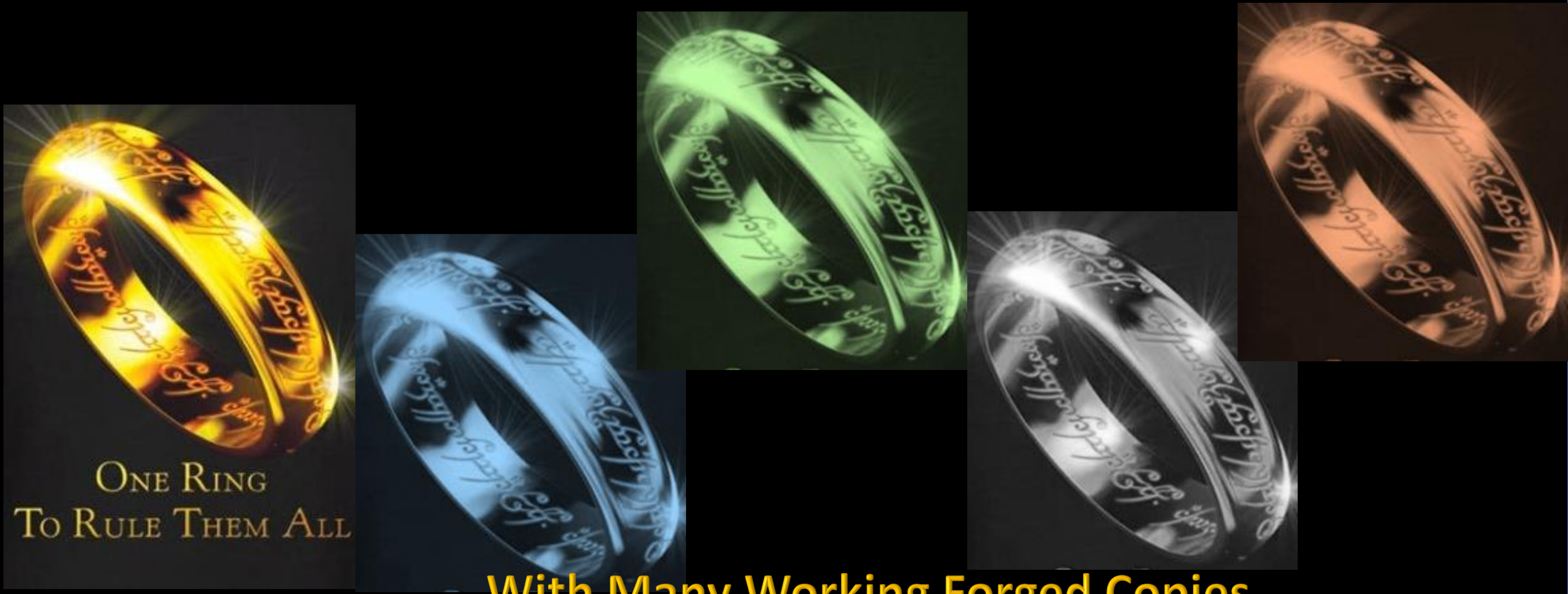


The P3 Handshake Protocol



SSL	S7 P3 Handshake
Server only authentication	Server (PLC) only authentication
Client generates keying material. Encrypts with server public key.	Client generates keying material. Encrypts with server public key.
Different servers use different public keys.	All PLCs from the same model and version use a single key.

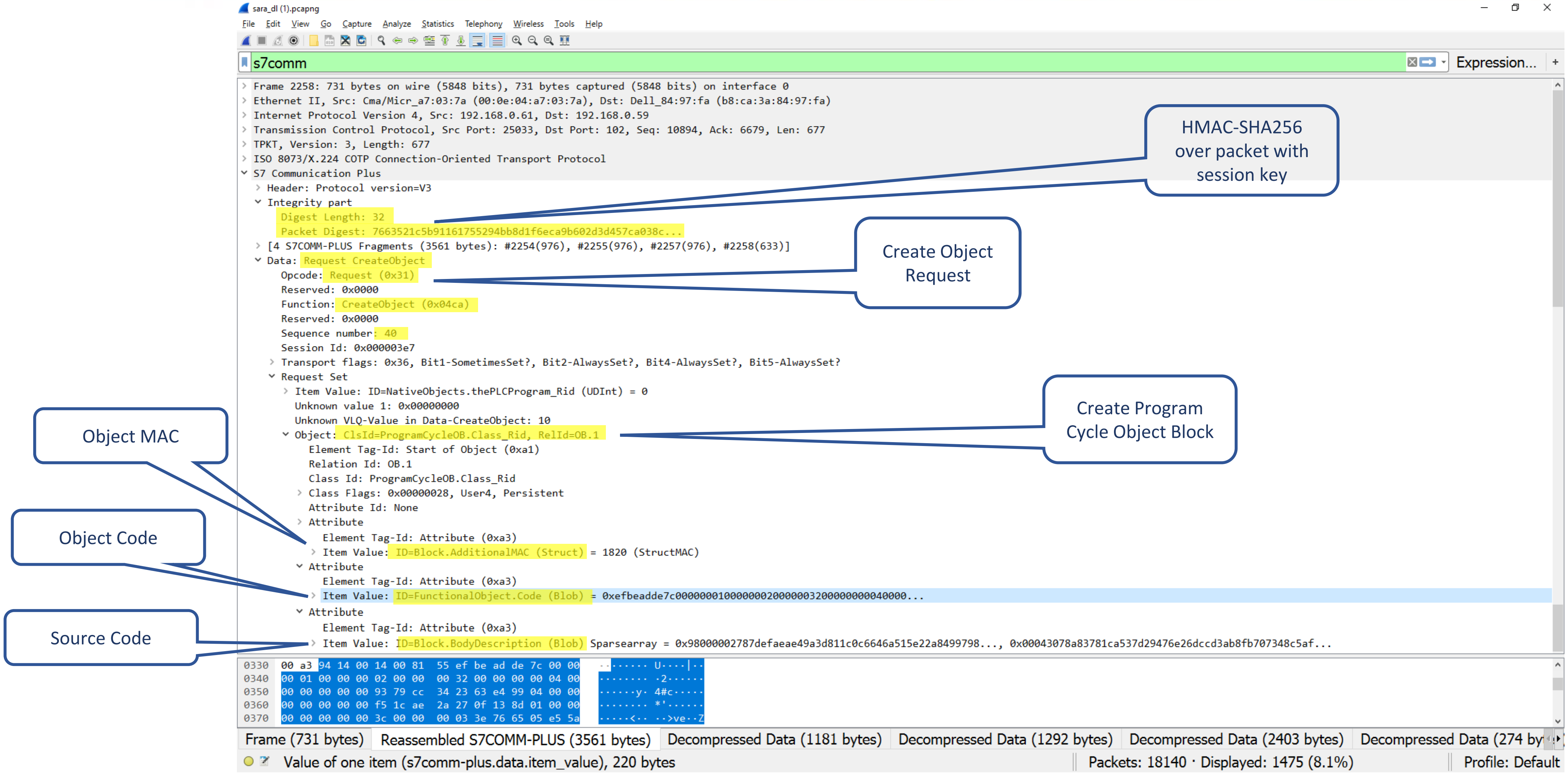
P3 Handshake Design Vulnerabilities



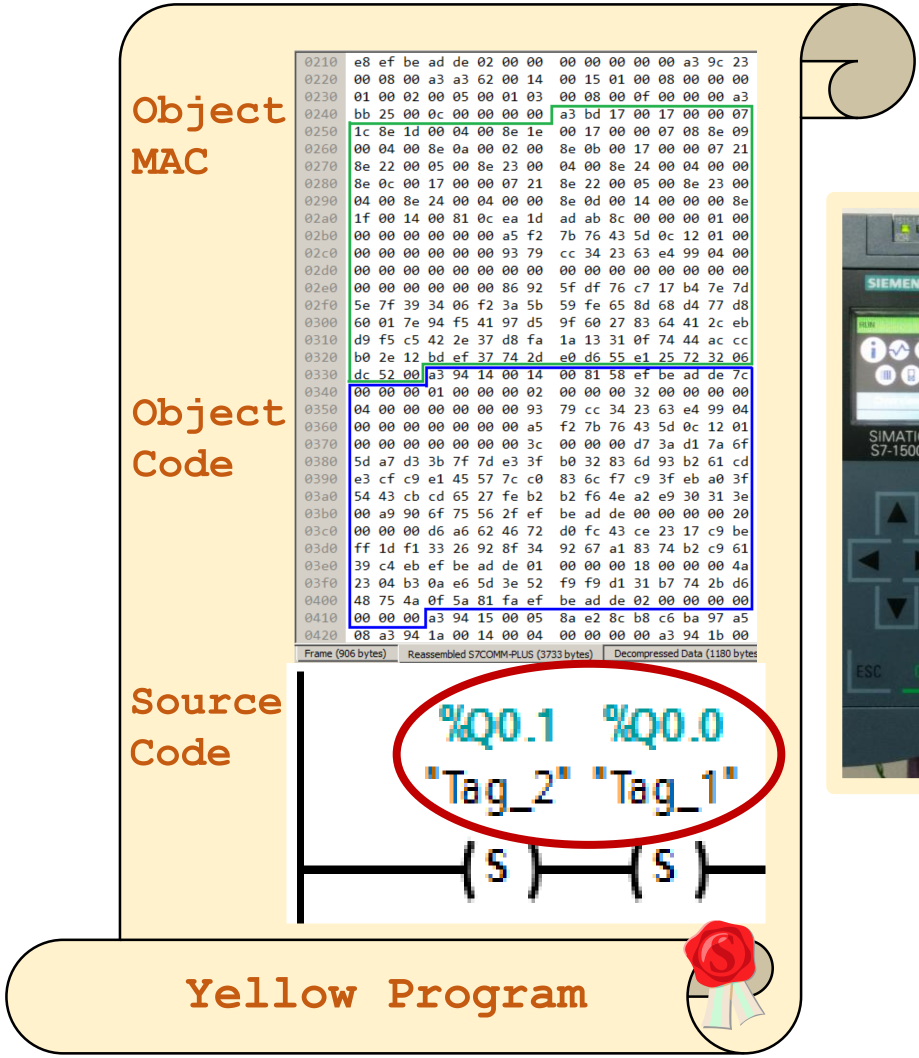
With Many Working Forged Copies

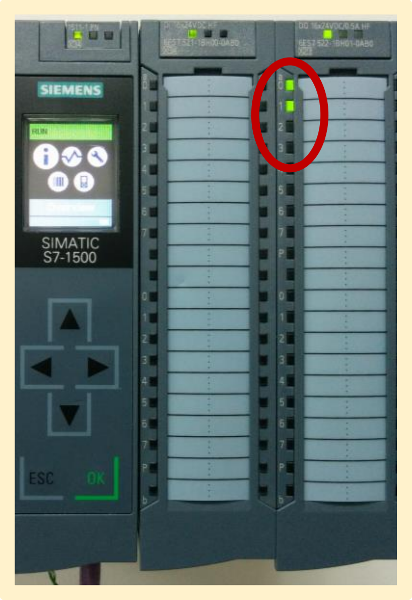
Attacking the P3 program download exchange

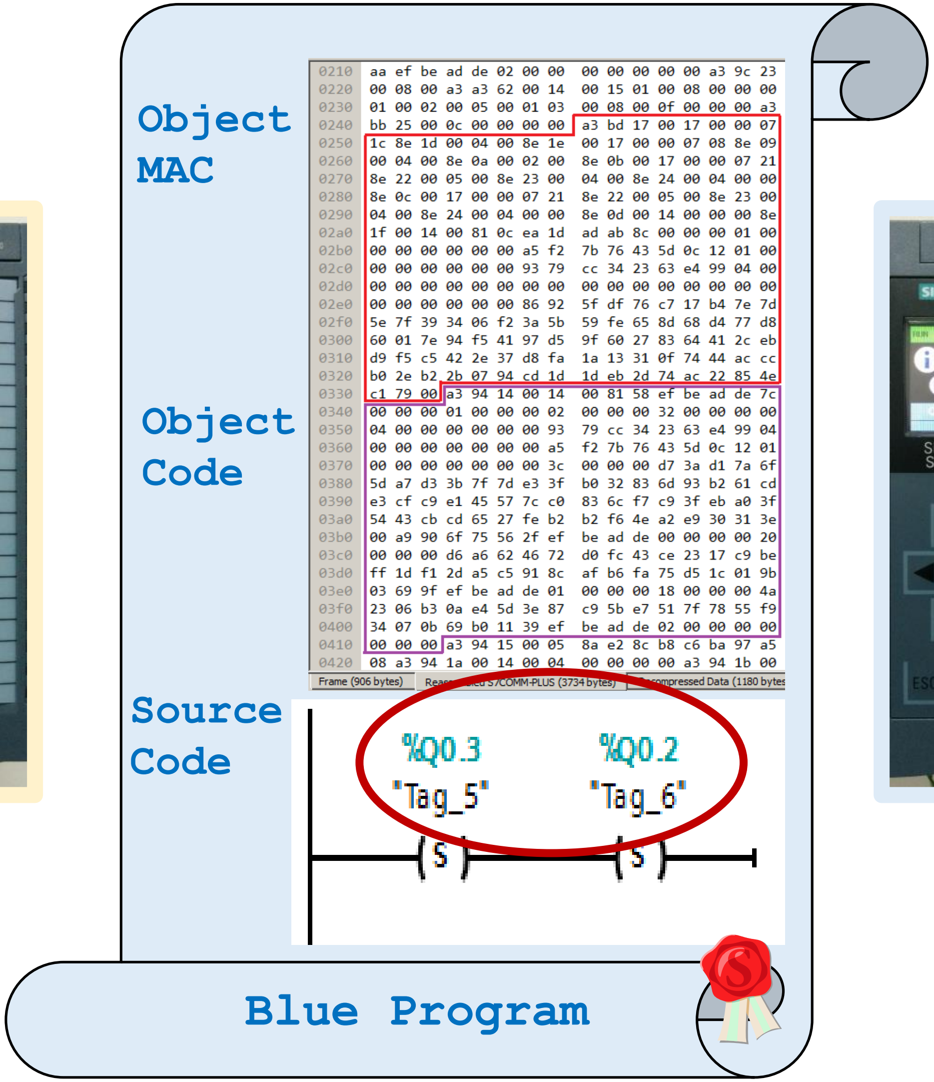
Control Program Create Message

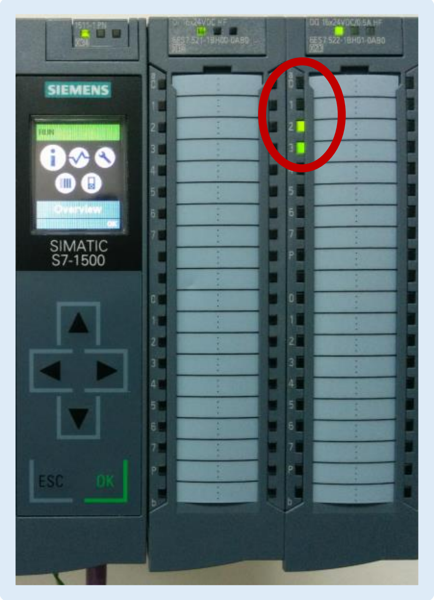


Control Program Representation

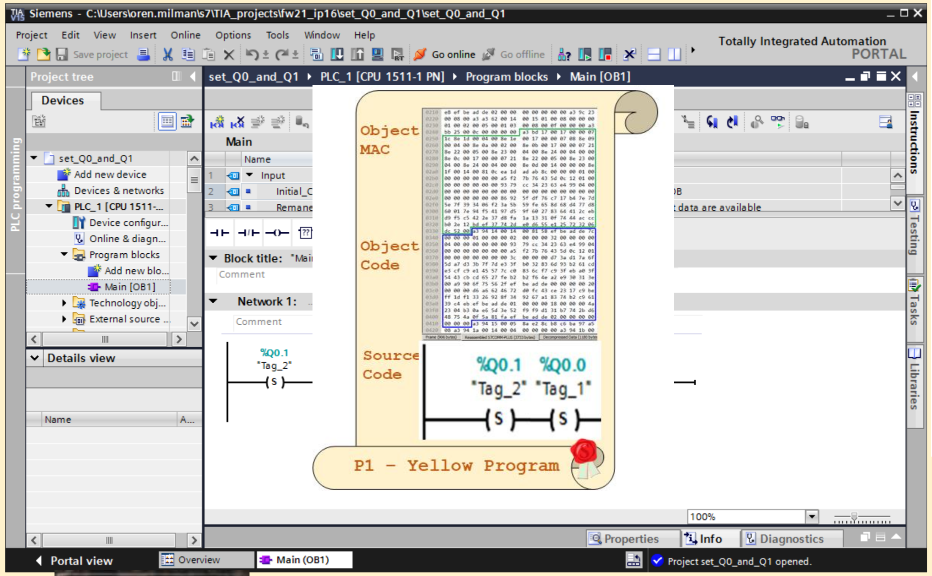


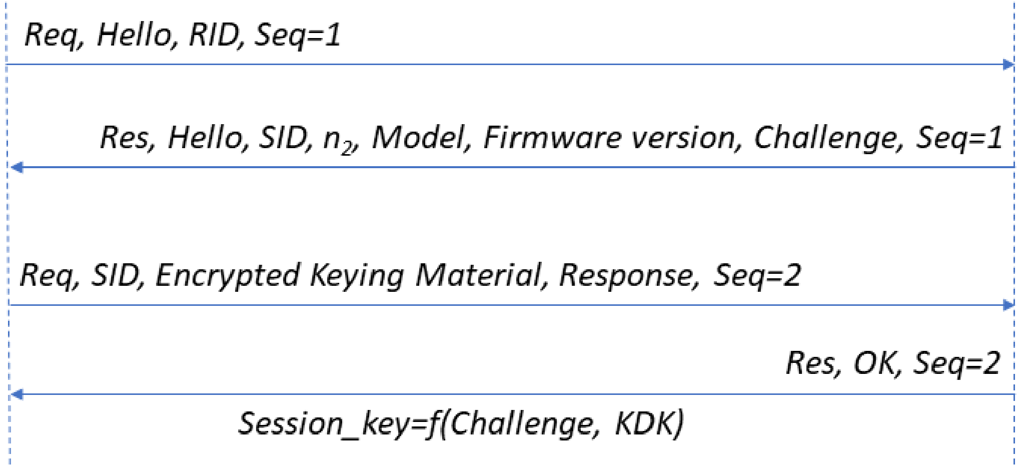






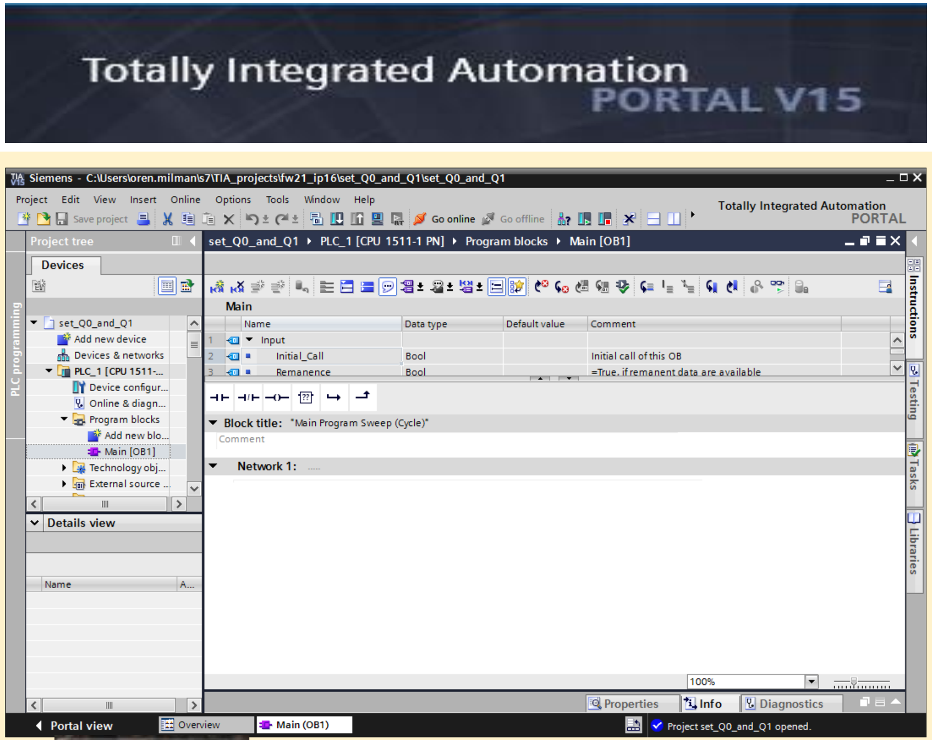
Program Download





Download

Program Upload



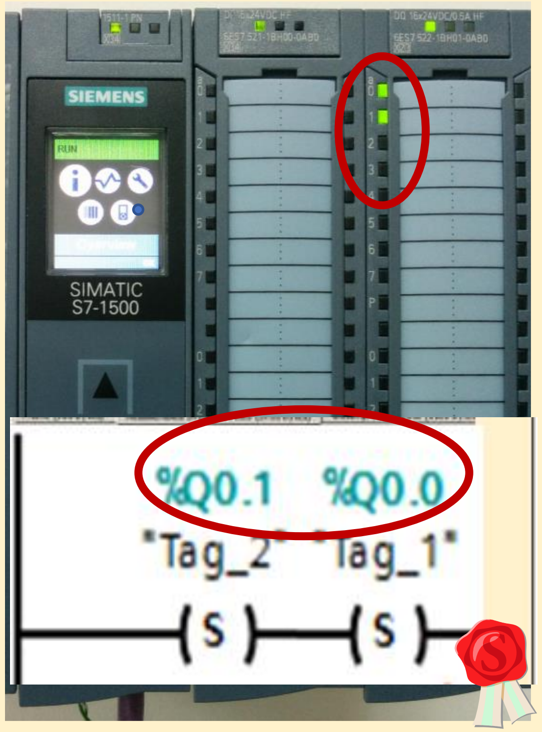
Req, Hello, RID, Seq=1

Res, Hello, SID, n_2, Model, Firmware version, Challenge, Seq=1

Req, SID, Encrypted Keying Material, Response, Seq=2

Res, OK, Seq=2

Session_key=f(Challenge, KDK)



Upload

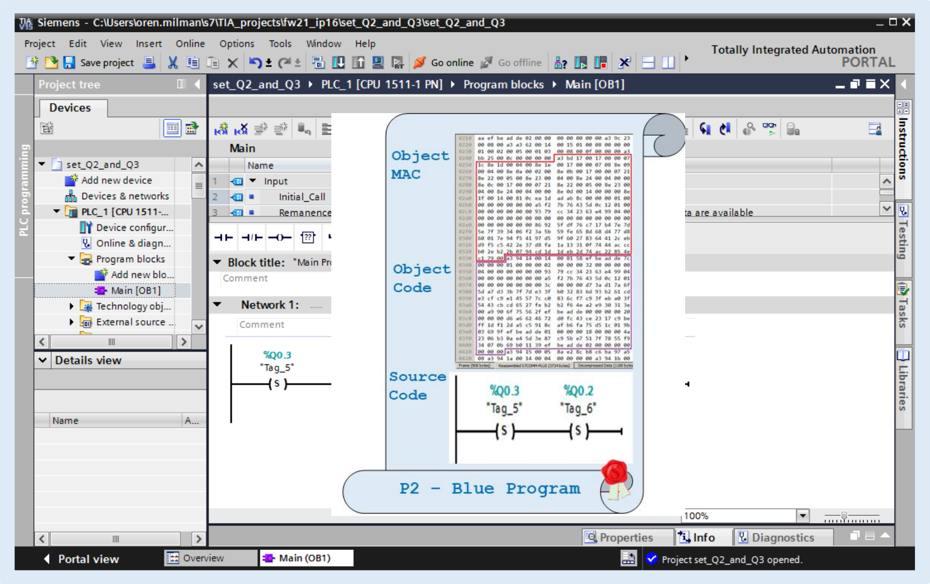
Rogue7: Rogue Engineering Station Attacks on Simatic S7 PLCs

The Rogue TIA

- No engineering station in the production network.
- The attacker brings it with him.
- Rogue TIA: The attack system consists of a modified legitimate TIA and a malicious proxy.

Rogue TIA

Rogue TIA Stealth Program Injection
Setup Phase



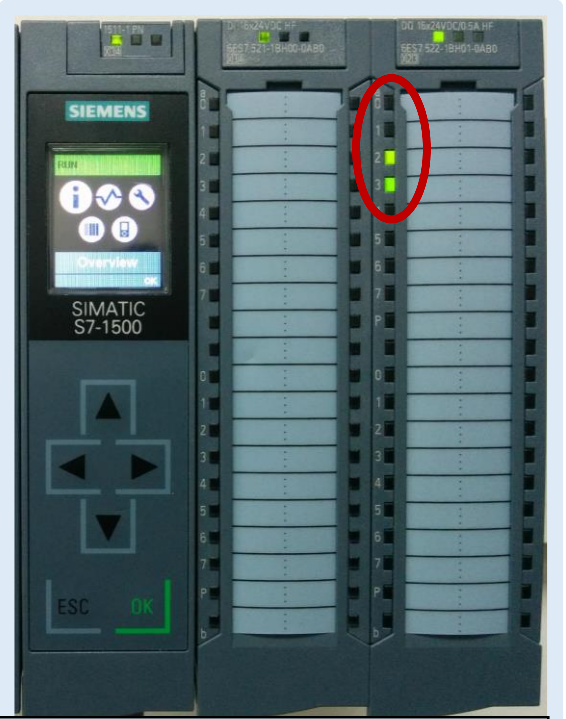
Req, Hello, RID, Seq=1

Res, Hello, SID, n_2, Model, Firmware version, Challenge, Seq=1

Req, SID, Encrypted Keying Material, Response, Seq=2

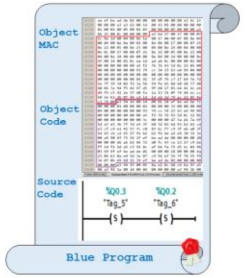
Res, OK, Seq=2

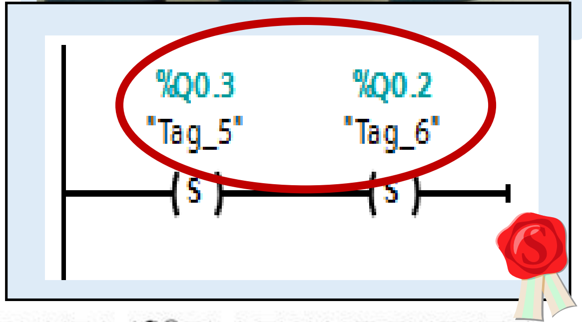
Session_key=f(Challenge, KDK)



Download

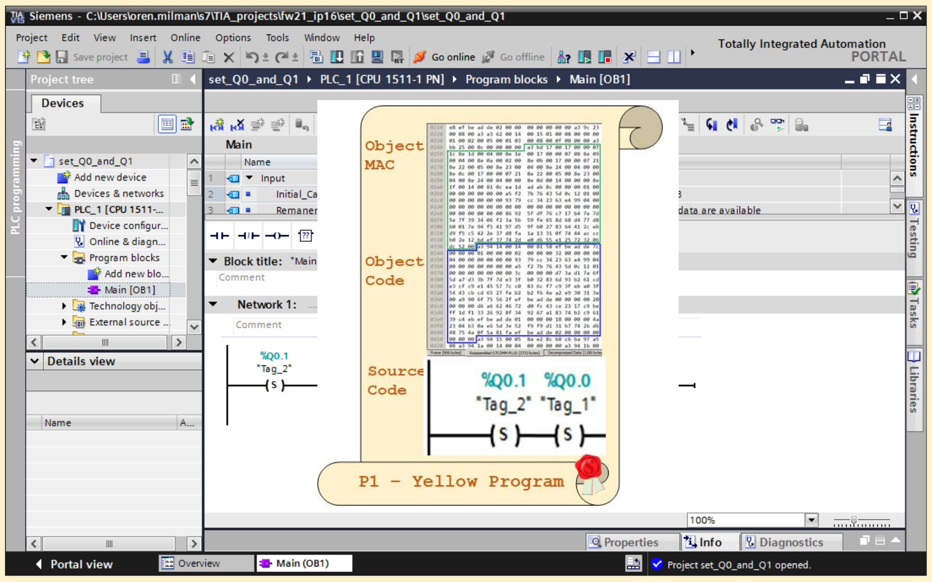
Record

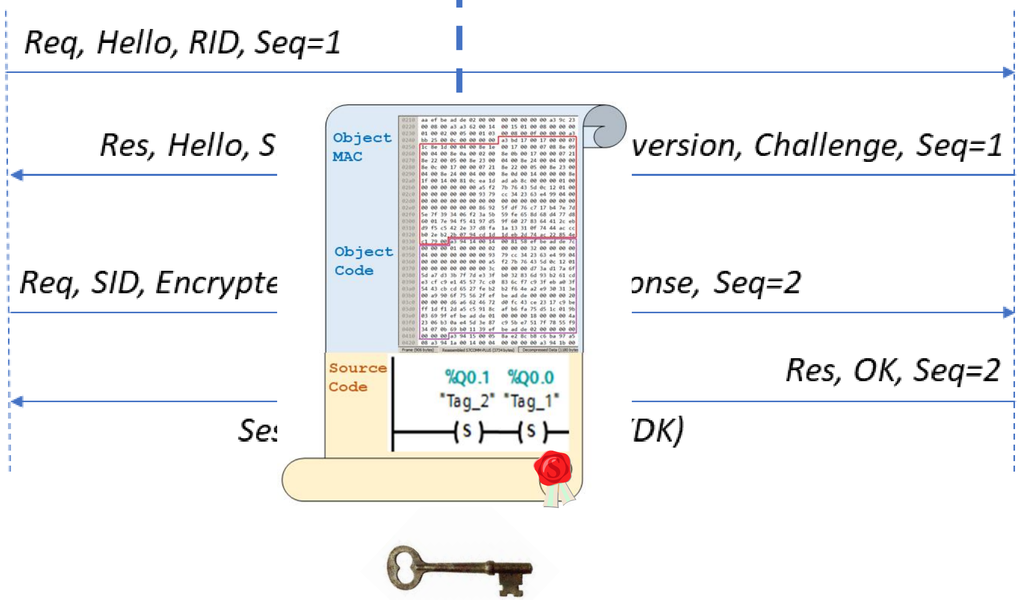


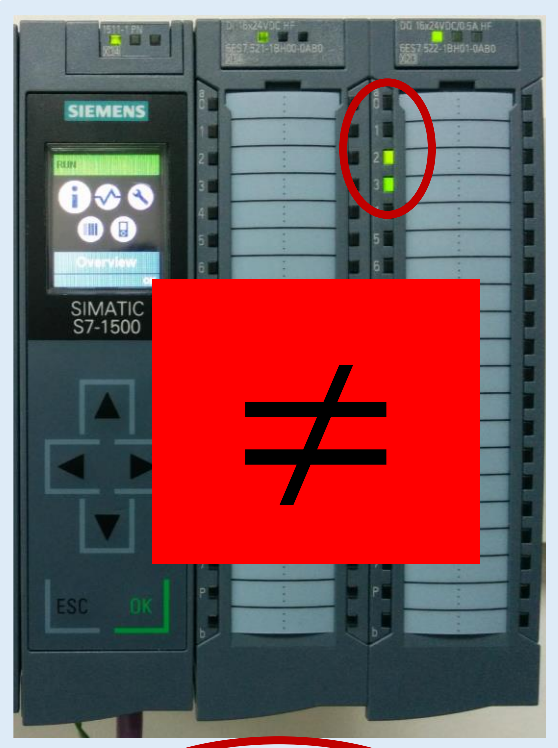


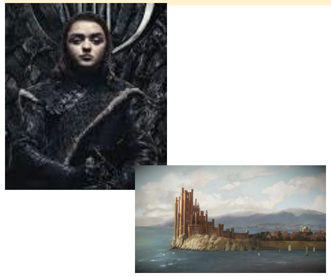
Rogue7: Rogue Engineering Station Attacks on Simatic S7 PLCs

Rogue TIA Stealth Program Injection Attack Phase



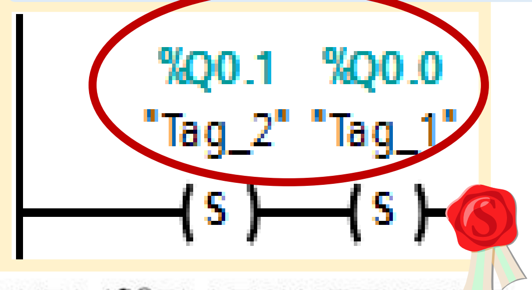






Download

Re-calc integrity

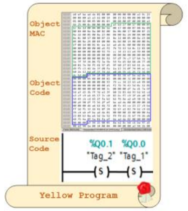


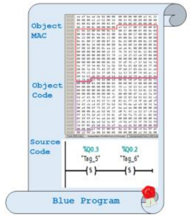
But we cannot always bring our own TIA with us....

Introducing

The Rogue Engineering Station

Rogue Engineering Station



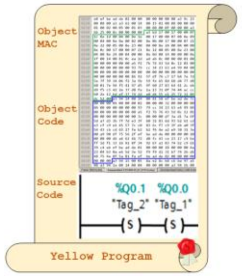


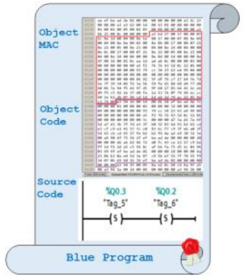
Rogue Engineering Station

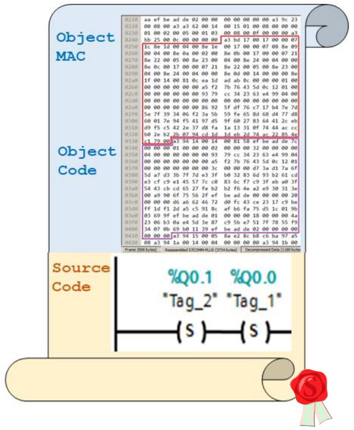
- A Python attack script that impersonates a TIA
- Inputs: PLC's IP address; Yellow and blue programs download pcap files

- Runs a live download session with the PLC
  - Four-way hand-shake
  - All S7 message fields are taken from the yellow pcap file
  - Except, the object code and MAC that are taken from blue pcap file
  - Fixes message SID, integrity protection and additional cookies

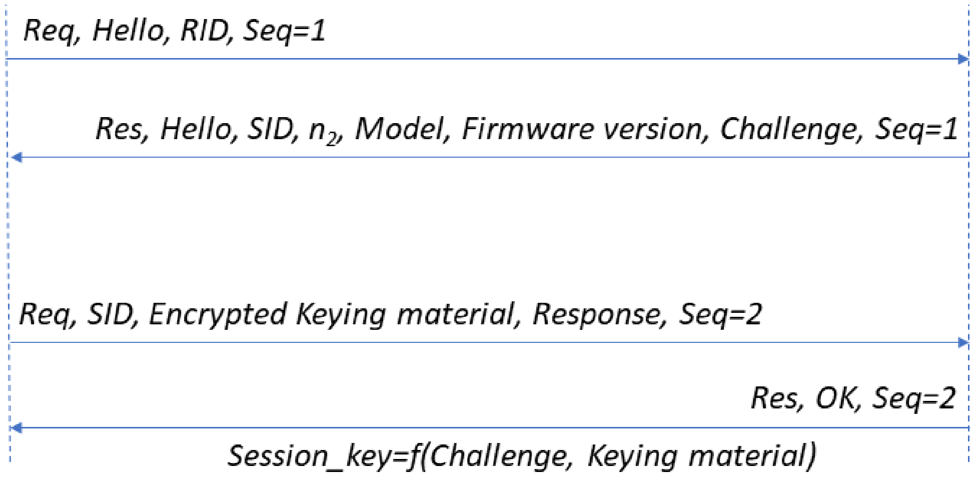
Rogue Engineering Workstation
Stealth Program Injection

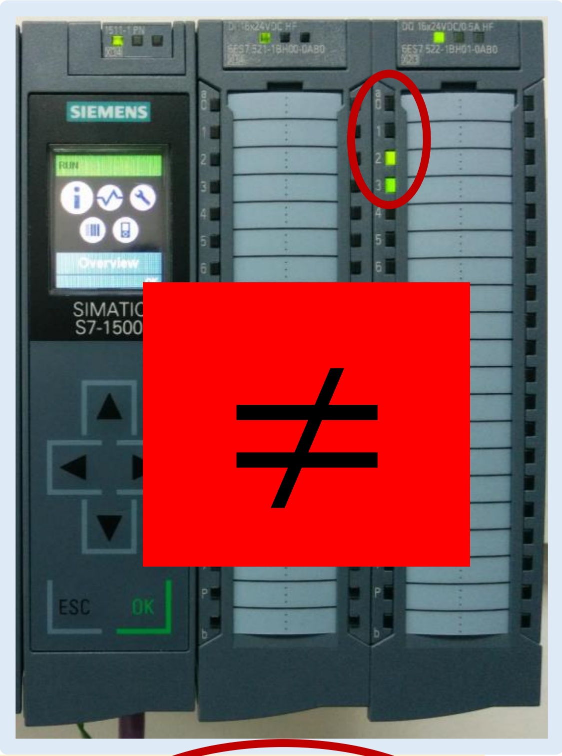


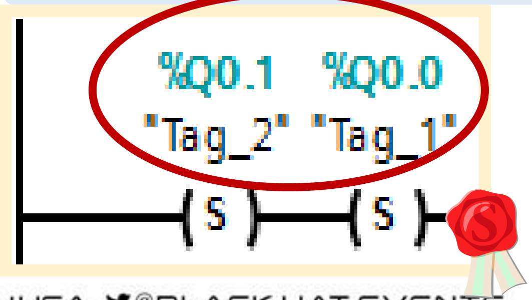




Fix SID
Fix other cookies
Re-calc integrity







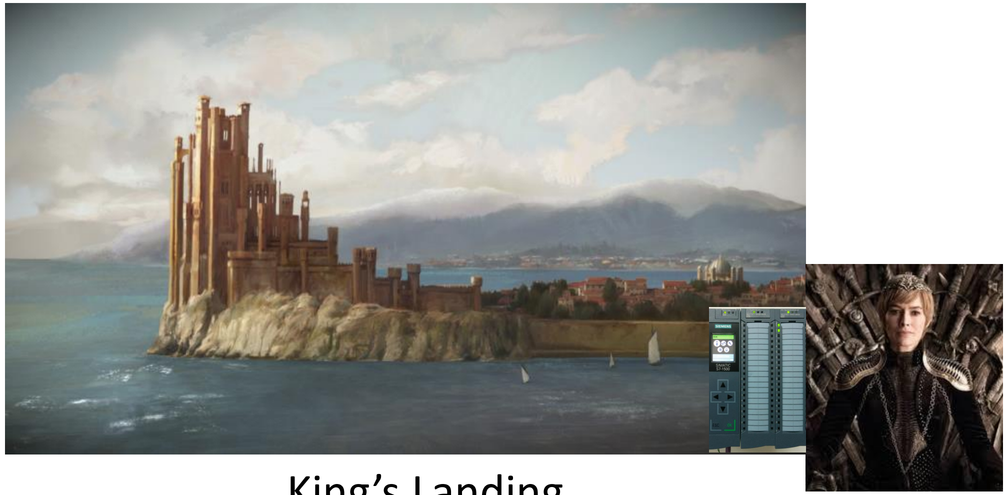
Download

Rogue7: Rogue Engineering Station Attacks on Simatic S7 PLCs

Step7 Impersonation

My Lab

The Wall

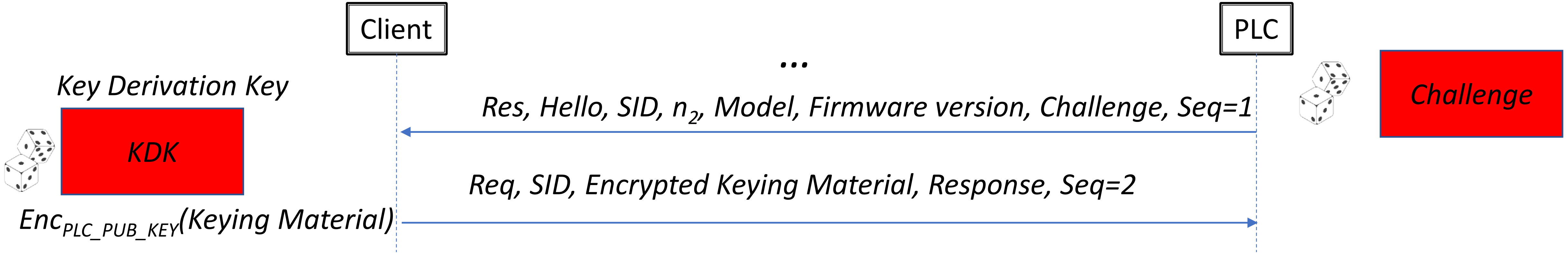


King's Landing

P3 Handshake Details

Deep technical details not for the faint-hearted

P3 Session Key Establishment



- Recall that:

$$Session\_key = f(Challenge, KDK)$$

- What is this *EncryptedKeyingMaterial* Structure?

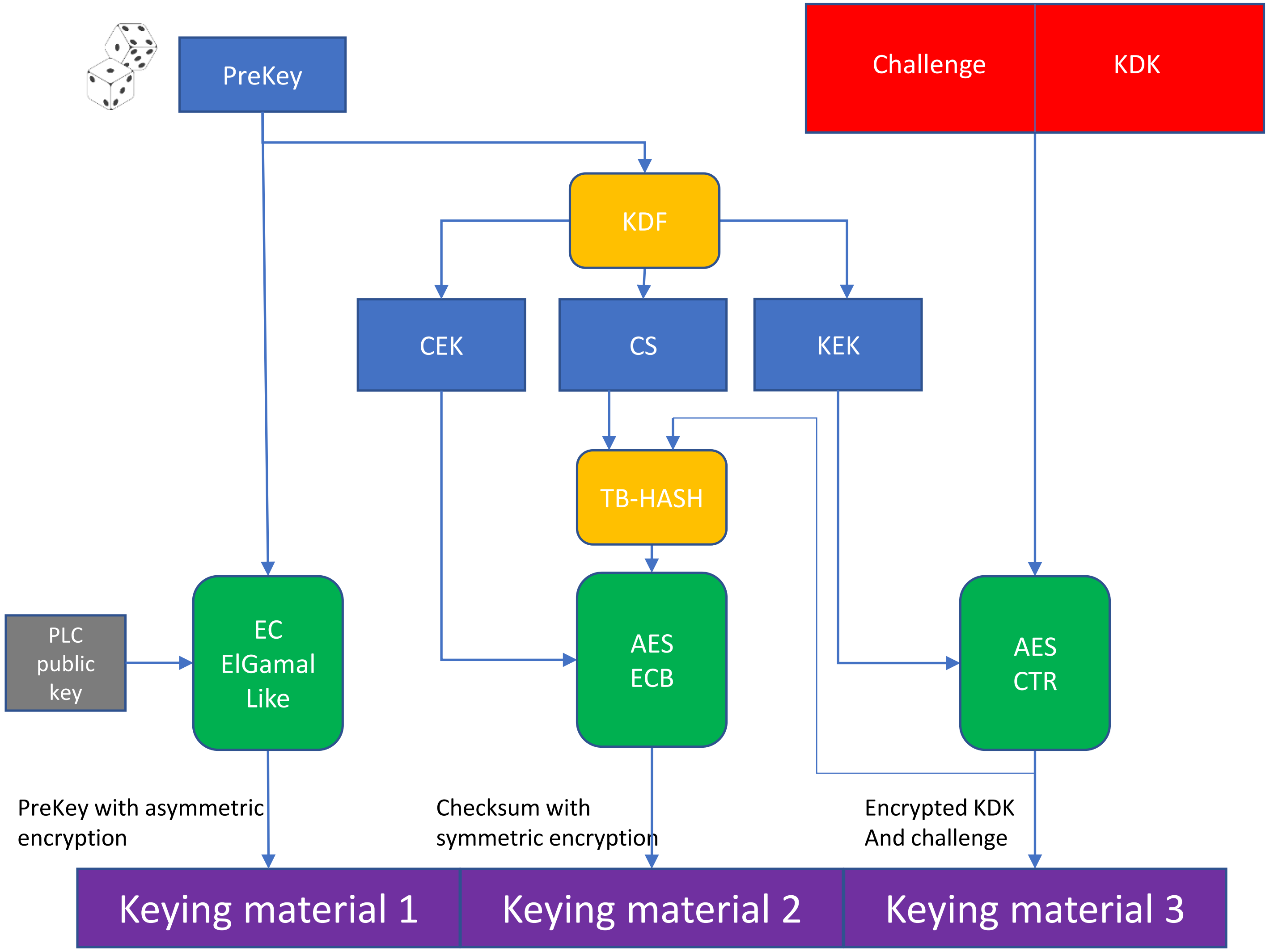
P3 Handshake
Cryptographic Primitives in Use

- Public-private key based asymmetric encryption
  - Elliptic curve: EC-ElGamal
- Symmetric encryption
  - AES
    - Electronic Code Book – ECB Mode
    - Counter mode
- Key Derivation Function: KDF

- Non-cryptographic checksum: Tabulation hash

P3 – KDK Sharing

1. Generate 20 bytes PreKey
    1. Encrypt it using EC-ElGamal–like encryption with the plc public key and add it to Keying material
2. Calculate KDF on PreKey and get
    1. Checksum Encryption Key (CEK)
    2. Checksum Seed (CS)
    3. Key Encryption Key (KEK)
3. Concatenate the KDK to the challenge, encrypt them using AES-CTR with the KEK, and add to Keying material
4. Initiate the Tabulation Hash with CS and calculate checksum over (3)
5. Encrypt (4) using AES-ECB with CEK and add to Keying material



P3 – Asymmetric Keys

- The public keys are stored in compressed .key files at
  [TIA INSTALLATION]\Data\Hwcn\Custom

- Each key file contains
  - Metadata (version, key type, key family, etc.)
  - **Key data – PLC public key for the EC-ElGamal-like encryption**

P3 – An Example .key File

version: 1

orderNumber: s71500-connection

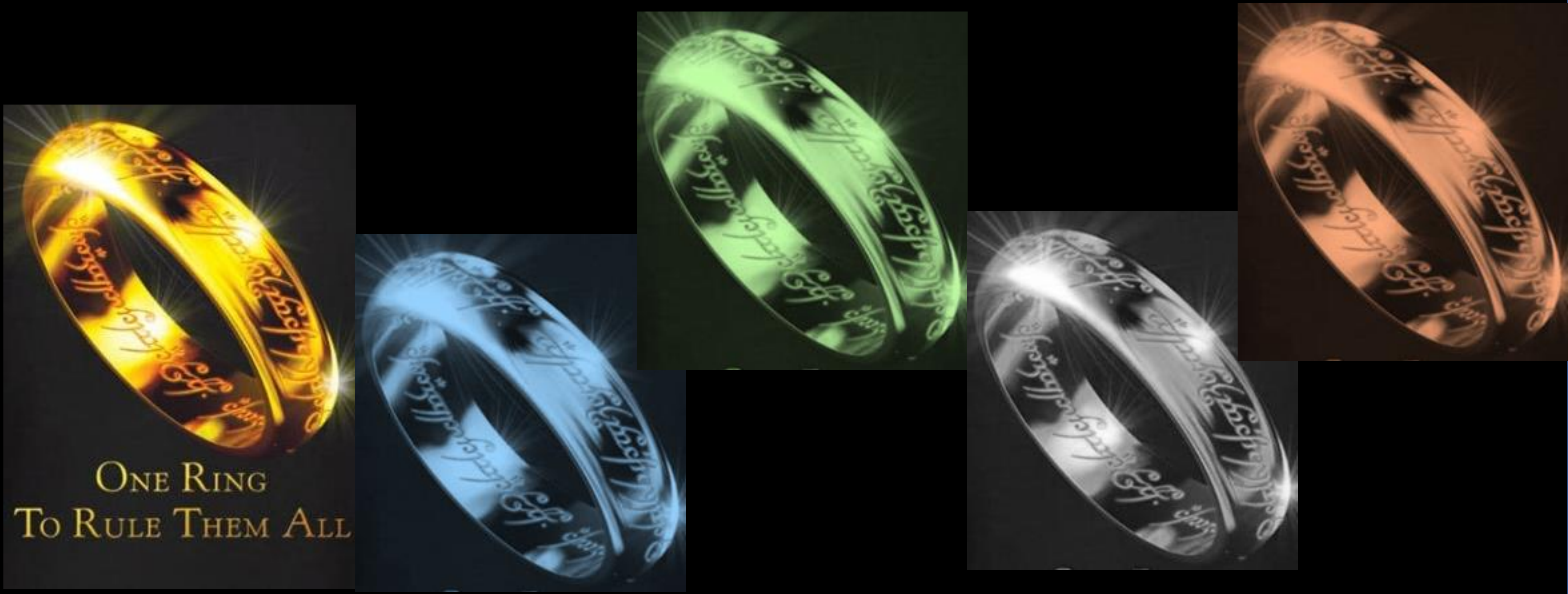
firmwareVersion:

keyType: connection

familyType: S7-1500

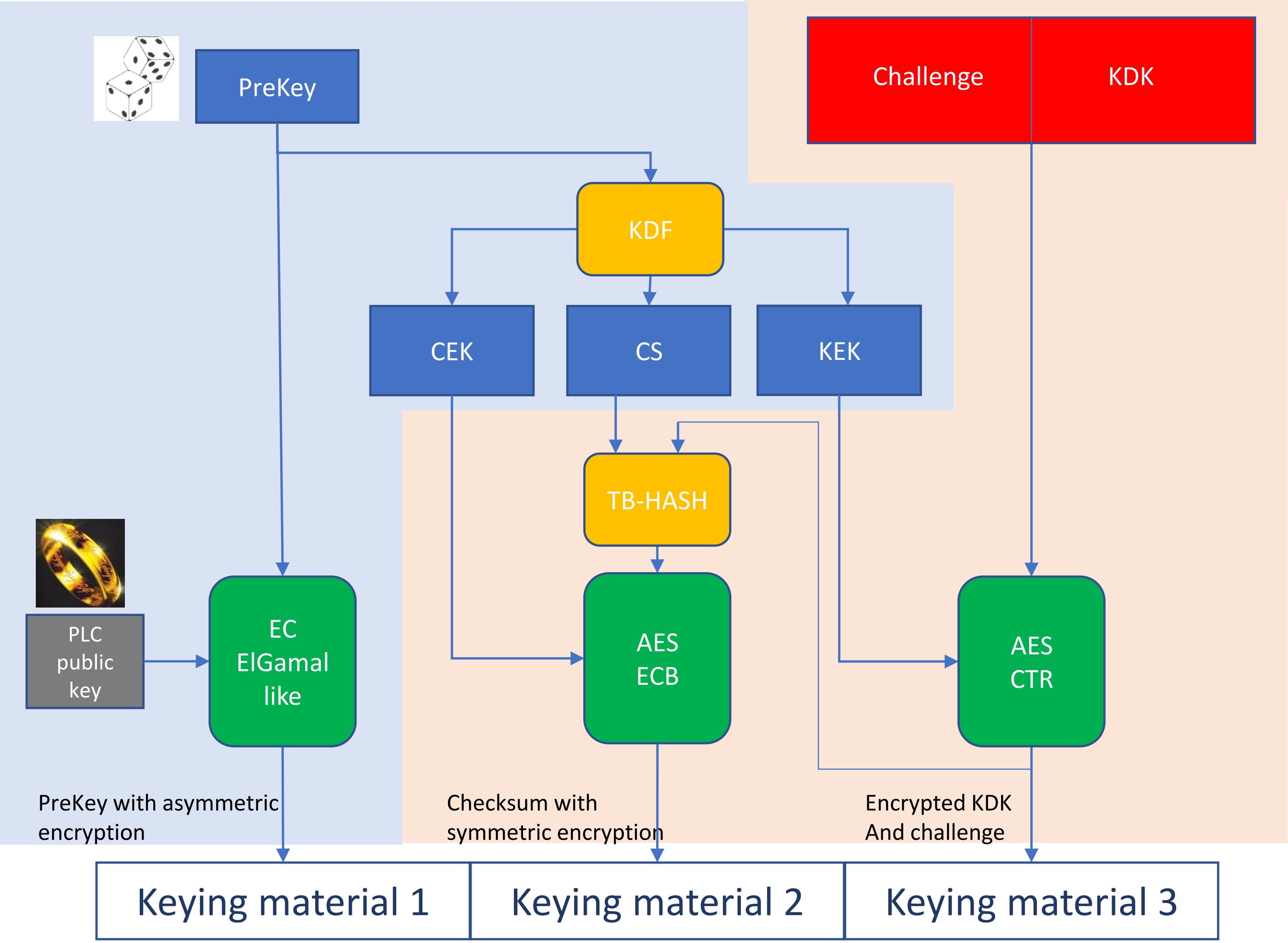
key data: 8456…

P3 Handshake Design Vulnerabilities



Rogue Engineering Station Session Key Establishment

1. Pre-calculate PreKey encryption and CEK, CS, KEK
2. Let Python do the symmetric encryptions and the checksum calculation to build keying material 2 & 3
3. The Python script wraps the session key derivation function from the relevant dll
   - We didn't reverse the session key derivation function *f*, due to lack of time



REing the TIA Handshake

- TIA is huge – find your target files
  - OMSp_core_managed.dll – main S7 communication DLL
  - Mixed mode DLL
    - Managed CIL bytecode – C#
    - Native (unmanaged) x86/x64 opcodes – C++
- Choose your tools
  - Managed code (.NET) - Reflector/dnSpy
  - Native C++ code - IDA Pro

- Improve your reverse engineering starting point

REing the TIA Handshake

- TIA is huge – find your target files
  - OMSp_core_managed.dll – main S7 communication DLL
  - Mixed mode DLL
    - Managed CIL bytecode – C#
    - Native (unmanaged) x86/x64 opcodes – C++
- Choose your tools
  - Managed code (.NET) - Reflector/dnSpy
  - Native C++ code - IDA Pro

- Improve your reverse engineering starting point

Tip #1 – Identify Native Code Entry Points

- omsp_core_managed.dll managed part opened in dnSpy

```
public unsafe static Error SetServerPublicKeyGlobal(uint keyType, byte[] key)
{
    if (key != null && key.Length != 0)
    {
        ref byte byte& = ref key[0];
        byte* ptr = ref byte&;
        uint num = (uint)key.Length;
        Blob blob;
        <Module>.OMS.Blob.{ctor}(ref blob);
        Error result;
        try
        {
            long num2 = <Module>.OMS.StructValue.copy_data(ref blob, (void*)ptr, num, 0u, false);
            if ((((num2 & -9223372036854775808L) != 0L) ? 1 : 0) != 0)
            {
                Error error = new Error(ref num2);
                result = error;
            }
            else
            {
                num2 = <Module>.OMS.ClientSession.s_set_server_public_key(keyType, ref blob);
                Error error2 = new Error(ref num2);
                result = error2;
            }
```

- Let's click on *s_set_server_public_key*

Tip #1 – Identify Native Code Entry Points

```
// Token: 0x0600037A RID: 890 RVA: 0x00109780 File Offset: 0x00108B80
[SuppressUnmanagedCodeSecurity]
[MethodImpl(MethodImplOptions.Unmanaged | MethodImplOptions.PreserveSig)]
internal unsafe static extern long s_set_server_public_key(uint, OMS.Blob*);
```

- Wait…Where is the function body???
- Looks like it's an unmanaged function…
- How can we find its implementation?

Tip #1 – Identify Native Code Entry Points

- RVA for the native part

Image Base
(0x180000000)
+ RVA
(0x109780)

=

0x180109780

```
// Token: 0x0600037A RID: 890 RVA: 0x00109780 File Offset: 0x00108B80
[SuppressUnmanagedCodeSecurity]
[MethodImpl(MethodImplOptions.Unmanaged | MethodImplOptions.PreserveSig)]
internal unsafe static extern long s_set_server_public_key(uint, OMS.Blob*);
```

```
.text:0000000180109780 OMS__ClientSession__s_set_server_public_key proc near
.text:0000000180109780
.text:0000000180109780 var_38          = dword ptr -38h
.text:0000000180109780 var_28          = xmmword ptr -28h
.text:0000000180109780 arg_0           = qword ptr  8
.text:0000000180109780 arg_8           = qword ptr  10h
.text:0000000180109780 arg_10          = qword ptr  18h
.text:0000000180109780 arg_18          = qword ptr  20h
.text:0000000180109780
.text:0000000180109780 ; __unwind { // j___CxxFrameHandler3
.text:0000000180109780                 mov     rax, rsp
.text:0000000180109783                 push    rdi
.text:0000000180109784                 push    r14
.text:0000000180109786                 push    r15
.text:0000000180109788                 sub     rsp, 50h
.text:000000018010978C                 mov     qword ptr [rax-48h], 0FFFFFFFFFFFFFFFEh
.text:0000000180109794                 mov     [rax+8], rbx
.text:0000000180109798                 mov     [rax+10h], rbp
.text:000000018010979C                 mov     [rax+20h], rsi
.text:00000001801097A0                 mov     r15, rdx
.text:00000001801097A3                 mov     ebp, ecx
.text:00000001801097A5                 and     ebp, 0FF00h
.text:00000001801097AB                 or      ebp, 10h
.text:00000001801097AE                 mov     r14, cs:18096D0B0h
.text:00000001801097B5                 add     r14, 2C8h
.text:00000001801097BC                 mov     [rax-40h], r14
.text:00000001801097C0                 mov     rax, [r14]
.text:00000001801097C3                 mov     r8, [rax+8]
.text:00000001801097C7                 lea     rcx, OMS_Recursive_Thread_Mutex_vtable
```

Tip #2: C++ and RTTI

- Run-Time Type Information
- Allow C++ programmers to examine object types dynamically
- The information must be inside the binary
- We could use it

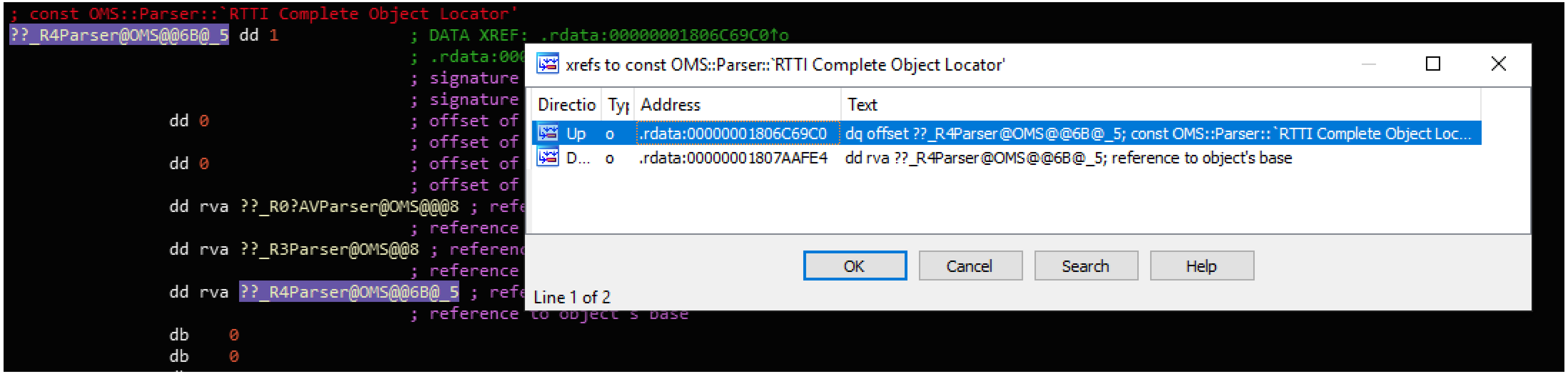
Tip #2: C++ and RTTI – Virtual Tables

1. Find RTTI Objects
2. Locate the relevant virtual tables

```
; const OMS::Parser::`RTTI Complete Object Locator'
??_R4Parser@OMS@@6B@_5 dd 1                ; DATA XREF: .rdata:00000001806C69C0↑o
                                           ; .rdata:00000001807AAFE4↓o
                                           ; signature
                                           ; signature
                dd 0                       ; offset of this vtable in complete class (from top)
                                           ; offset of this vtable in complete class (from top)
                dd 0                       ; offset of constructor displacement
                                           ; offset of constructor displacement
                dd rva ??_R0?AVParser@OMS@@@8 ; reference to type description
                                           ; reference to type description
                dd rva ??_R3Parser@OMS@@8 ; reference to hierarchy description
                                           ; reference to hierarchy description
                dd rva ??_R4Parser@OMS@@6B@_5 ; reference to object's base
                                           ; reference to object's base
                db    0
                db    0
```

Tip #2: C++ and RTTI – Virtual Tables

1. Find RTTI Objects
2. Locate the relevant virtual tables



Tip #2: C++ and RTTI – Virtual Tables

1. Find RTTI Objects
2. Locate the relevant virtual tables
3. Rebuild them

```
.rdata:00000001806C69C0                  dq offset ??_R4Parser@OMS@@6B@_5 ; const OMS::Parser::`RTTI Complete Object Locator'
.rdata:00000001806C69C8 off_1806C69C8    dq offset sub_1800ACA20 ; DATA XREF: sub_1800AC380+131↑o
.rdata:00000001806C69C8                                          ; sub_1800ACAB0+1C↑o ...
.rdata:00000001806C69D0                  dq offset sub_1800AD020
.rdata:00000001806C69D8                  dq offset sub_1800CC970
.rdata:00000001806C69E0                  dq offset sub_1800ABC40
.rdata:00000001806C69E8                  dq offset sub_1800AB8C0
.rdata:00000001806C69F0                  dq offset sub_1800AB8D0
.rdata:00000001806C69F8                  dq offset sub_1800ACD10
.rdata:00000001806C6A00                  dq offset sub_1800CC820
```

Tip #2: C++ and RTTI – Virtual Tables

1. Find RTTI Objects
2. Locate the relevant virtual tables
3. Rebuild them

```
.rdata:00000001806C69C0                   dq offset ??_R4Parser@OMS@@6B@_5 ; const OMS::Parser::`RTTI Complete Object Locator'
.rdata:00000001806C69C8 ParserOOMS_vtable ParserOOMS::vtable <offset ParserOOMS__sub_1800ACA20, \
.rdata:00000001806C69C8                                         ; DATA XREF: ParserOOMS__ParserOOMS+131↑o
.rdata:00000001806C69C8                                         ; sub_1800ACAB0+1C↑o ...
.rdata:00000001806C69C8                                       offset ParserOOMS__sub_1800AD020, \
.rdata:00000001806C69C8                                       offset ParserOOMS__sub_1800CC970, \
.rdata:00000001806C69C8                                       offset ParserOOMS__sub_1800ABC40, \
.rdata:00000001806C69C8                                       offset ParserOOMS__sub_1800AB8C0, \
.rdata:00000001806C69C8                                       offset ParserOOMS__sub_1800AB8D0, \
.rdata:00000001806C69C8                                       offset ParserOOMS__sub_1800ACD10, \
.rdata:00000001806C69C8                                       offset ParserOOMS__sub_1800CC820>
```

Combine Them All

- Scan the .NET metadata and grab the native entry points
- Scan your .idb and grab the relevant RTTI data
- Use IDAPython to add this information to your .idb

Combine Them All - Example

```
signed __int64 __fastcall sub_1800ABC80(DWORD *a2, char a1)
{
  DWORD *v2; // rdi
  char v3; // si
  __int64 v4; // rcx
  __int64 v5; // rcx
  DWORD *v6; // rbx
  DWORD *v7; // rax

  v2 = a2;
  v3 = a1;
  v4 = *((_QWORD *)a2 + 385);
  if ( !v4 || (*(unsigned __int8 (**)(void))(*(_QWORD *)v4 + 8i64))() != a1 )
  {
    v5 = *((_QWORD *)v2 + 385);
    v6 = 0i64;
    if ( v5 )
    {
      v6 = *(DWORD **)(v5 + 32);
      (**(void (__fastcall ***)(__int64, signed __int64))v5)(v5, 1i64);
    }
    v7 = (DWORD *)sub_18017DBE0(v3, v2);
    *((_QWORD *)v2 + 385) = v7;
    if ( !v7 )
      return -6701032989083762692i64;
    if ( v6 )
      *((_QWORD *)v7 + 4) = v6;
  }
  return 0i64;
}
```

Combine Them All - Example

```
signed __int64 __fastcall ParserOOMS::createScanner(ParserOOMS *this, char scanner_type)
{
  PackedScannerOOMS *v4; // rcx
  PackedScannerOOMS *old_scanner; // rcx
  ACE_Message_Block *v6; // rbx
  PackedScannerOOMS *v7; // rax

  v4 = this->packedScanner;
  if ( !v4 || (unsigned __int8)v4->vtable->PackedScannerOOMS::get_default_scanner_type() != scanner_type )
  {
    old_scanner = this->packedScanner;
    v6 = 0i64;
    if ( old_scanner )
    {
      v6 = old_scanner->base.message_block;
      old_scanner->vtable->PackedScannerOOMS::destructor(old_scanner, 1);
    }
    v7 = f_create_scanner(scanner_type, this);
    this->packedScanner = v7;
    if ( !v7 )
      return -6701032989083762692i64;
    if ( v6 )
      v7->base.message_block = v6;
  }
  return 0i64;
}
```

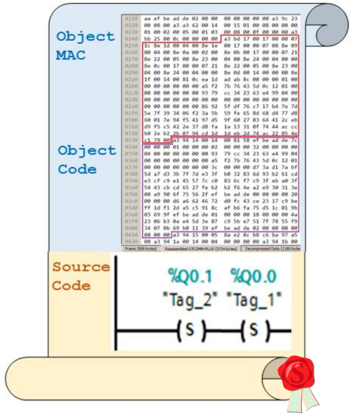
- Much better, isn't it?

Rogue Engineering Station Stealth Program Injection

Attack Demo

Summary

- Vulnerabilities in the S7 protocol – P3
  - TIA is not authenticated
  - "One Ring to Rule them All"
- A Python attack tool that impersonates TIA
  - Download a recorded program to any S7-1500 PLC
  - Stealth program injection attack



Thank You!