




black hat[®]
EUROPE 2023

DECEMBER 4-7

EXCEL LONDON / UK



A Decade After Stuxnet: How Siemens S7 is Still an Attacker's Heaven

Colin Finck and Tom Dohrmann

Who are we?

Colin Finck

✉ c.finck@enlyze.com

🐦 @ColinFinck

- Reverse-engineering industrial control systems at ENLYZE for the past 5 years
- Reverse-engineering Windows internals for the ReactOS Project since 2006
- Rust enthusiast

Tom Dohrmann

✉ t.dohrmann@enlyze.com

🐦 @13erbse

- Hacker and Software Developer
- Interested in Low Level Systems
- Member of the FluxFingers CTF Team



A Short Introduction to PLCs

From a Computer Science perspective: Embedded Computers

- Ethernet ports
- Some even with x86 CPUs



Image source © Siemens AG 2023, All rights reserved



Uses of PLCs



Manufacturing and Processing Industry



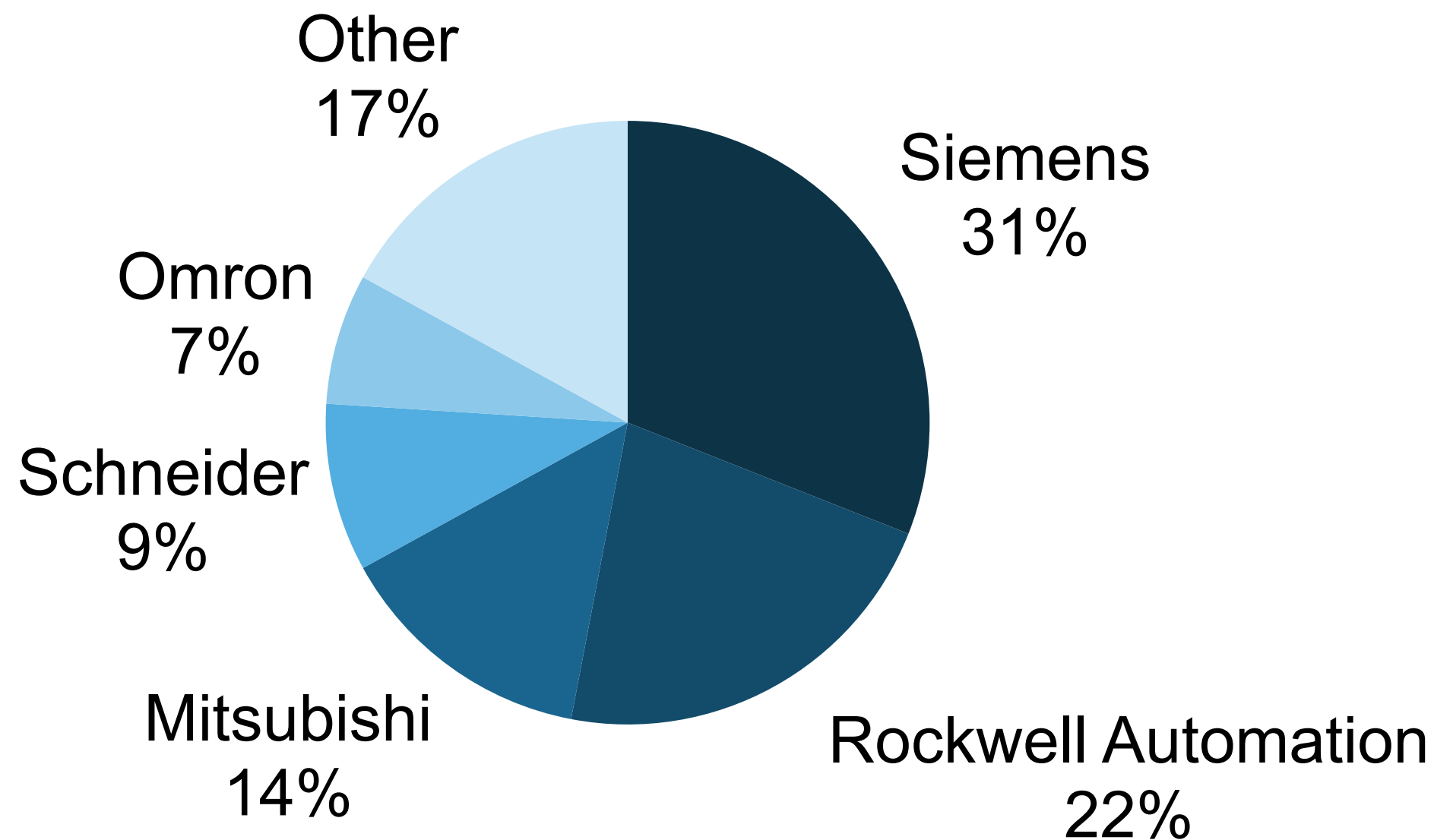
Power Plants, Grids, Pipelines, Water Utilities



Building Automation



Global PLC Market Share





IT world



OT world

Programming is standardized

Vendor-agnostic graphical and textual programming languages

The screenshot shows the Siemens SIMATIC Manager software interface. The title bar indicates the project path: "Siemens - C:\Users\User\Documents\Automatisierung\Cleaning Station Example\Cleaning station\Cleaning station". The menu bar includes "Project", "Edit", "View", "Insert", "Online", "Options", "Tools", "Window", and "Help". The toolbar contains icons for "Save project", "Go online", and "Go offline". The "Project tree" on the left shows the hierarchy: "Cleaning station" > "PLC_1 [CPU 1513-1 PN]" > "Program blocks" > "Simulation" > "Flow_Sim [FC14]". The main workspace displays a ladder logic diagram with the following components:

- Block 1: "Real" block with inputs "IN1" (connected to "V100DB".RbkOut.Value) and "IN2" (connected to "10.0").
- Block 2: ">=" block with input "Value" connected to the output of the "Real" block.
- Block 3: "&" block with inputs connected to the output of the ">=" block and the output of another ">=" block.
- Block 4: ">=" block with input "Value" connected to "P100DB".AutAct.Value.
- Block 5: ">=" block with input "Value" connected to "P101DB".AutAct.Value.
- Block 6: "CalcDose" block (FB2) with "EN" (Enable) connected to the output of the "&" block. It has two outputs: "Dose_Liters" (connected to "CMInterface_DB".Liter.Value) and "VolumeFlow" (connected to "FIT100DB".PV_Out.Value).

The right sidebar contains "Instructions", "Testing", "Tasks", and "Libraries". The bottom left corner has the "Infor" logo.

Communication is not standardized

Every PLC vendor has their proprietary protocol, classic lock-in

The screenshot displays a network traffic capture in Wireshark. The top pane shows two packets:

No.	Time	Source	Destination	Protocol	Length	Info
12	2021-10-14 10:45:51,270020	127.0.0.1	127.0.0.1	S7COMM-PLUS	463	←1031 Ver:[V2] Seq=2 [Req SetMultiVariables] ObjId=DynObjX7.64.4
13	2021-10-14 10:45:51,270043	127.0.0.1	127.0.0.1	TCP	44	102 → 1031 [ACK] Seq=268 Ack=552 Win=2619136 Len=0

The bottom pane shows the detailed view of the S7COMM-PLUS packet (packet 12). It displays a tree structure of the data:

- In Object Id: DynObjX7.64.4
- Item count: 2
- Item address count: 2
- > AddressList
- ▼ Valuelist
 - Item Value [1]: (Struct) = 1800 (StructSecurityKey)
 - Item Number: 1
 - > Datatype flags: 0x00
 - Datatype: Struct (0x17)
 - Value: StructSecurityKey
 - > Item Value: ID=SecurityKeyVersion (UDInt) = 0
 - > Item Value: ID=SecurityKeySecurityLevel (USInt) = 0
 - > Item Value: ID=SecurityKeyPublicKeyID (Struct) = 1825 (SecurityKeyID)
 - > Item Value: ID=SecurityKeySymmetricKeyID (Struct) = 1825 (SecurityKeyID)
 - ▼ Item Value: ID=SecurityKeyEncryptedKey (Blob) = 0xaddee1feb40000000100000001000000ecdc104910d795830101000000000001a73081f...
 - ID Number: SecurityKeyEncryptedKey
 - > Datatype flags: 0x00
 - Datatype: Blob (0x14)
 - Blob root ID: None
 - Blob size: 180
 - Value: addee1feb40000000100000001000000ecdc104910d795830101000000000001a73081f...

The right pane shows the raw hex and ASCII data for the selected packet, with the S7COMM-PLUS payload highlighted in blue.

Anyway, why can I connect to nearly every S7 without credentials?

Siemens S7-1200/1500 Protocol

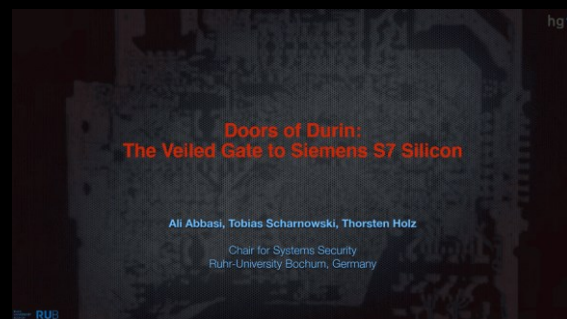
- July 2019: “There must be a single master key. How hard can it be?”
- 6 weeks later: Proof-of-Concept client to connect to most S7-1500

```
Eingabeaufforderung
Running `target\debug\examples\s7plus_network_test.exe 10.20.24.11:102`
#####
CreateObject
#####
Ok(CreateObjectResponse { object_ids: [1883242496, 1883242497], object: Some(Object { relation_id: 288, class_id: 287, class_flags: 0, attribute_id: 0, attributes: [Item { id: 233, value: ItemValue(WString(WString("01:BD426B091F08731A"))) }, Item { id: 299, value: ItemValue(UDInt(UDInt(536870914))) }, Item { id: 301, value: ItemValue(WString(WString("OMSP_11.00.00.06_59.06.00.01"))) }, Item { id: 303, value: ItemValue(Array(Array { datatype: USInt, vec: [USInt(USInt(188)), USInt(USInt(32)), USInt(USInt(188)), USInt(USInt(247)), USInt(USInt(217)), USInt(USInt(15)), USInt(USInt(121)), USInt(USInt(255)), USInt(USInt(110)), USInt(USInt(106)), USInt(USInt(211)), USInt(USInt(233)), USInt(USInt(72)), USInt(USInt(231)), USInt(USInt(75)), USInt(USInt(74)), USInt(USInt(223)), USInt(USInt(221)), USInt(USInt(52)), USInt(USInt(40))] })) }, Item { id: 306, value: ItemValue(Struct(Struct { value: 314, data: Items([Item { id: 315, value: ItemValue(UDInt(UDInt(704))) }, Item { id: 316, value: ItemValue(UDInt(UDInt(640))) }, Item { id: 317, value: ItemValue(UDInt(UDInt(8397120))) }, Item { id: 318, value: ItemValue(UDInt(UDInt(8397120))) }, Item { id: 319, value: ItemValue(WString(WString("1;6ES7 212-1BE40-0XB0 ;V4.4"))) }, Item { id: 320, value: ItemValue(WString(WString("2;576"))) }, Item { id: 321, value: ItemValue(UInt(UInt(3))) }])) })), objects: [] }) })
#####
SetMultiVariables
#####
Ok(SetMultiVariablesResponse { error_values: ErrorValueList({}), integrity_id: 0 })
```

Fast forward to 2023

More publications on the internals of Siemens PLCs – but hardly reproducible

2019



The Veiled Gate to Siemens S7 Silicon
(Abbasi et. al)

2019



Rogue7: Rogue Engineering Station Attacks on Simatic S7 PLCs
(Biham et al.)

2022



sOfT7: Revealing the Secrets of Siemens S7 PLCs
(Bitan & Dankner)

2022



Uncover Siemens SIMATIC S7 Hardcoded Cryptographic Keys
(Team82 at Claroty)

Siemens S7-1500 Software Controller

- Software-only variant of the S7-1500 PLC
- Runs in a VM on an x86 Siemens Industrial PC next to Windows
- Very accessible to the research community



Image source © Siemens AG 2023, All rights reserved

Analyzed Communication Protocol

- Analyzed protocol has been in use since 2015
- TLS handshake and transport introduced in 2022, but most PLCs have not been upgraded
- Concepts are similar, but cryptographic details are different between hardware PLCs and Software Controller

Decrypting the Firmware Image

- Firmware comes as encrypted ELF file
 - along with a self-contained decryptor
- Bitan & Dankner developed a harness for the Intel Pin framework to use the decryptor standalone
 - but not released to the general public
- We reimplemented the harness and released it at <https://github.com/enlyze/EnlyzeS7SoftwareControllerDecoder>

Decrypting the Firmware Image

For more information on this method, check out



s0fT7: Revealing the Secrets of the Siemens S7 PLCs

Sara Bitan | Alon Dankner

Joint work with **Professor Eli Biham, Maxim Barsky** and **Idan Raz**
Faculty of Computer Science, Technion – Israel Institute of Technology

Dynamic Analysis



Multiboot

Multiboot header exists, but at the wrong location.

→ We implemented a UEFI-based bootloader to load the image.

```
00003770 ff ff ff ff 01 01 01 01 ff ff ff ff ef be ad de |.....|
00003780 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
00003790 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
000037c0 02 b0 ad 1b 03 00 00 00 fb 4f 52 e4 00 00 00 00 |.....OR.....|
000037d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000037e0 20 57 61 72 6e 69 6e 67 3a 20 66 6f 75 6e 64 20 | Warning: found |
```

Early Boot Logging

Early boot logs a lot :)

```
TD_sprintf(acStack_e8, "    Using GPIO table index #%d, table is at 0x%08x.\n", param3,  
           (uint)(&PTR_DAT_18dd32c4)[param3 * 2]);  
CF_puts(acStack_e8);
```

But puts implementation was stubbed out :(

```
void CF_puts(char *param_1)  
  
{  
    return;  
}
```

Early Boot Logging

➔ Patched the functions in our custom bootloader

```
patcher.set_pc(0x10c072a0);  
// mov    dx,0x3f8  
patcher.place_instruction(bytes: &[0x66, 0xba, 0xf8, 0x03]);  
let label: Label = patcher.label();  
// mov    al,BYTE PTR [rdi]  
patcher.place_instruction(bytes: &[0x8a, 0x07]);  
// out    dx,al  
patcher.place_instruction(bytes: &[0xee]);  
// inc    rdi  
patcher.place_instruction(bytes: &[0x48, 0xff, 0xc7]);  
// test   al,al  
patcher.place_instruction(bytes: &[0x84, 0xc0]);  
patcher.jne(label);  
// ret  
patcher.place_instruction(bytes: &[0xc3]);
```

Early Boot Logging

```
Checking mlfb 'default' against index 0, mlfb='6ES7 672-5DC11-0YA0 '  
Checking mlfb 'default' against index 1, mlfb='6ES7 672-5SC11-0YA0 '  
Checking mlfb 'default' against index 2, mlfb='6ES7 672-5VC11-0YA0 '  
Checking mlfb 'default' against index 3, mlfb='6ES7 672-5WC11-0YA0 '  
Checking mlfb 'default' against index 4, mlfb='default'  
Using GPIO table index #4, table is at 0x18dd32e8.  
Setting up Local APIC...  
found IO-APIC 0 at 0xfec00000 (version 0x20) with 24 entries  
setting IA32_EFER.NXE  
Initializing IPC...  
  prepare local structures...  
    - setting ISR attributes  
    - initializing wait elements  
    - initializing spinlocks and memory  
prepare own notification info...  
do architecture specific init...  
  
ADONIS boot successful, starting first user thread...
```

Hypercalls

The kernel tries to communicate with the hypervisor via hypercalls:

```
[root@desktop:/sys/kernel/debug/tracing]# echo 1 > events/kvm/kvm_hypercall/enable
```

```
[root@desktop:/sys/kernel/debug/tracing]# cat trace_pipe
```

```
<...>-1914303 [000] ..... 24024.416368: kvm_hypercall: nr 0x401 a0 0x0 a1 0x0 a2 0xffffffff a3 0x10002230
<...>-1914303 [000] ..... 24024.416372: kvm_hypercall: nr 0x401 a0 0x0 a1 0x1 a2 0xffffffff a3 0x2c
qemu-system-x86-1914303 [000] ..... 24024.967093: kvm_hypercall: nr 0x504 a0 0x10c006a8 a1 0x0 a2 0xffffffff a3 0x2c
qemu-system-x86-1914303 [000] ..... 24024.968275: kvm_hypercall: nr 0x102 a0 0xffffffff a1 0xffffffff a2 0xffffffff a3 0x2c
qemu-system-x86-1914303 [000] ..... 24024.968278: kvm_hypercall: nr 0x503 a0 0x28 a1 0xfffffc18 a2 0x0 a3 0x2c
qemu-system-x86-1914303 [000] ..... 24025.019161: kvm_hypercall: nr 0x101 a0 0x68747541 a1 0x444d4163 a2 0x69746e65 a3 0x0
qemu-system-x86-1914303 [000] ..... 24025.019164: kvm_hypercall: nr 0x102 a0 0x100199ac a1 0x444d4163 a2 0x69746e65 a3 0xffffffff
qemu-system-x86-1914303 [000] ..... 24025.019938: kvm_hypercall: nr 0x204 a0 0xfffffc18 a1 0x1 a2 0x1 a3 0xfffffc18
```

Query Memory Region

Find Memory Region

Read IO APIC Register

➔ Switched to QEMU TCG and modified VMMCALL instruction

PCI Devices

- Identified two required PCI devices
 - wsync
 - com_trc
- Started implementing them in QEMU.
- Couldn't make progress, eventually gave up.
Further research is needed.

<https://github.com/enlyze/s7-1500-software-controller-loader>

<https://github.com/enlyze/qemu/tree/soft-sps>



Static Analysis

Decompiler woes

The firmware is a 32-bit ELF running 64-bit code but uses 32-bit pointers.

Ghidra aggressively casts between integers and pointers and loses type information.

Other decompilers suffer from similar problems.

```
31 c0          XOR          EAX,EAX
85 f6          TEST         ESI,ESI
74 1c          JZ           LAB_16845302
66 2e 0f      NOP
1f 84 00
00 00 00 00

                                LAB_168452f0
67 44 8b      MOV         R8D,dword ptr [EDI + EAX*0x4]
04 87
67 44 89      MOV         dword ptr [EDX + EAX*0x4],R8D
04 82
48 83 c0 01   ADD         RAX,0x1
39 c6          CMP         ESI,EAX
77 ee          JA          LAB_168452f0

                                LAB_16845302
f3 c3          RET
```

```
uVar1 = 0;
if (param_2 != 0) {
    do {
        *(undefined4 *) (long)(int)(param_3 + uVar1 * 4) =
            *(undefined4 *) (long)(int)(param_1 + uVar1 * 4);
        uVar1 = uVar1 + 1;
    } while (uVar1 < param_2);
}
return;
```


Custom Processor Definitions to the Rescue

We forked Ghidra's x86-64 processor definitions and changed the pointer size.

```
moffs32: segWide^[imm64] is addrsize=2 & highseg=1 & segWide & imm64 { tmp:8 = segWide + imm64; export *:4 tmp; }  
moffs32: segWide^[imm64] is addrsize=2 & highseg=1 & segWide & imm64 { tmp:4 = segWide:4 + imm64:4; export *:4 tmp; }
```


```
<default_pointer_alignment value="8" />  
<pointer_size value="8" />  
<default_pointer_alignment value="4" />  
<pointer_size value="4" />
```

```
uVar1 = 0;  
if (count != 0) {  
    do {  
        dest_3[uVar1] = src[uVar1];  
        uVar1 = uVar1 + 1;  
    } while (uVar1 < count);  
}  
return;
```

Custom Processor Definitions to the Rescue

<https://github.com/enlyze/ghidra-adonis-processor>

 **ghidra-adonis-processor**

 main ▾

 1 branch

 1 tag

Go to file

Add file ▾

 Code ▾



Freax13 add scripts

bd1af4c on Sep 11  7 commits



data

change long size to 4

last month



ghidra_scripts

add scripts

last month



LICENSE

fork ghidra x86 processor files

3 months ago



Module.manifest

fork ghidra x86 processor files

3 months ago



NOTICE

modify for use with ADONIS kernel

last month

RTTI

- Ghidra-Cpp-Class-Analyzer by Andrew Strelsky
- Required small fix
- Identified about 8000 classes

```

*****
* typeid for WinAC_Asymm... *
*****
WinAC_Asymmetric_Decryp...XREF[... 19feb5c4(*)
...eb5b0 3c    __si...
          0d
          2a ...
...eb5b0 3c 0d    __cl...          super_...          XREF[... 19feb5c4(*)
          2a 1a
          90 b5 ...
...eb5b0 3c 0d    type...          super_...          XREF[... 19feb5c4(*)
          2a 1a
          90 b5 ...
...eb5b0 3c 0d    void...PTR_ARRAY_1..._vptr          XREF[... 19feb5c4(*)
          2a 1a
...eb5b4 90 b5    ds * typeid-na...__name = "27WinA...
          fe 19
...eb5b8 b0 b0    __cl...Asymmetric_...__base...
          fe 19
...eb5bc 00    ??    00h
...eb5bd 00    ??    00h
...eb5be 00    ??    00h
...eb5bf 00    ??    00h

```

Static Analysis Helpers

Auto-renaming functions based on logging calls

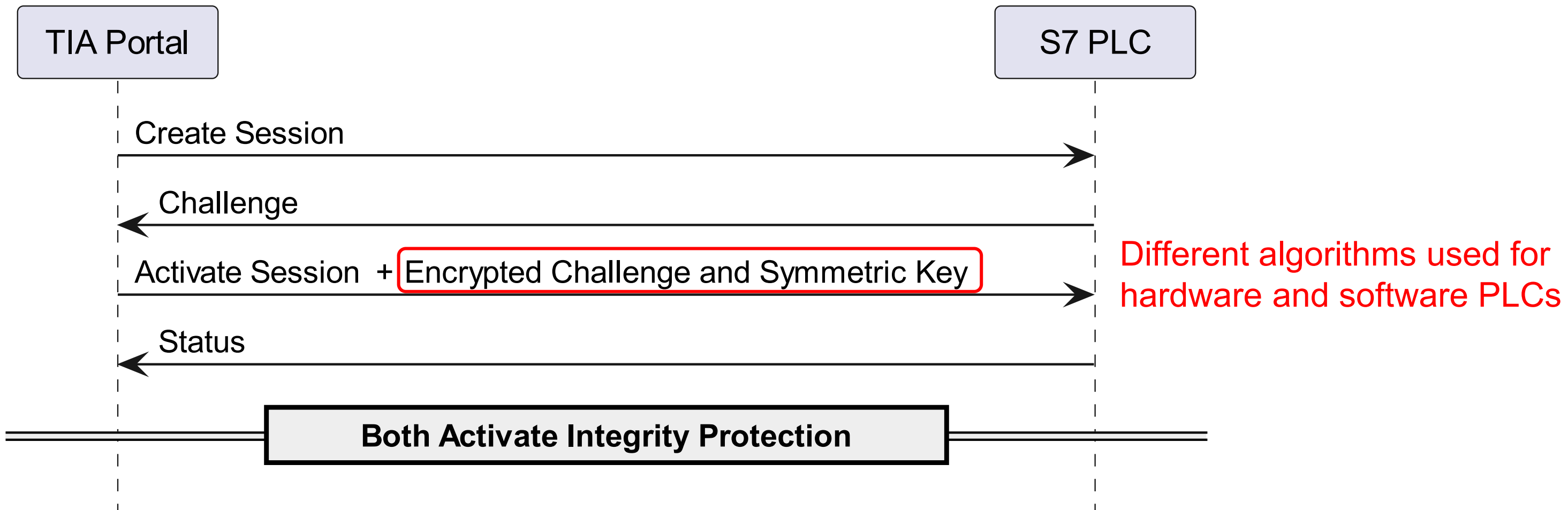
```
TD_debug_enter_function(0xdb, "AcpiFindRootPointer", "tbxfroot", 8);  
TD_debug_enter_function(0x1c1, "AcpiTerminate", "utxface", 1);  
TD_debug_enter_function(0xdc, "HwDerivePciId", "hwpci", 0x10);  
TD_debug_enter_function(0xa3, "PsGetNextPackageLength", "psargs", 0x20);
```

Auto-decoding error codes based on Wireshark dissector

```
if (*(int *) (param_1 + 0x6c) == -1) {  
    /* OMS Error: GeneralIntegrity/IntegrityError */  
    return 0x80414c0001defea1;  
}
```

Cryptographic Details of the Handshake

30,000-foot View of the Handshake





Software PLC Handshake

1. Asymmetric Key Exchange
2. Shared Key Derivation
3. Encryption of Challenge & Symmetric Key

Asymmetric Key Exchange

A shared secret between client and PLC is derived using Elliptic Curve Diffie-Hellman.

Curve parameters:

- 192-bit over a prime field with
 $p = 0xff13$
- $a = -1$
- $b = 0x6241e52b7bd8790514ebe1e51c8368cd9d56e1ae21de9cbc$
- $G_x = 0x6f74ce776d67b1d7a49f8cf0e26b77bc677cf771962e4427$
- $G_y = 0x7eaa7f6516d614857b4cda3e3f2fb5c642fc8285fb86575f$

Asymmetric Key Exchange

A shared secret between client and PLC is derived using Elliptic Curve Diffie-Hellman.

PLC public key parameters:

- $x = 0x8e6d4846b080f387e3d48858c54a40b7fb28dc02b706e25f$
- $y = 0x12fe2110375f5e3627148ac04f1c5473042275e4b1091567$

Asymmetric Key Exchange

```
/* This code calculates  $x * x * x - x + \text{constant} - (y * y)$   
   This fits the equation of an elliptic curve:  $y*y=x*x*x+ax+b$   
   This code checks that the public key is on the curve. */
```

```
TD_square_192bit(&local_2a8, (TD_prime_field_value *)&public_key);  
TD_mult_192bit(&local_2a8, &local_2a8, (TD_prime_field_value *)&public_key);  
TD_sub_192bit(&local_2a8, &local_2a8, (TD_prime_field_value *)&public_key);  
TD_add_192bit(&local_2a8, &local_2a8, &TD_curve_b);  
TD_square_192bit(&local_288, public_key_y);  
TD_sub_192bit(&local_2a8, &local_2a8, &local_288);  
TD_truncate_192bit(&local_2a8, &local_2a8);  
iVar1 = TD_all_zero(&local_2a8);
```

Asymmetric Key Exchange

Quick refresher on Elliptic Curve Diffie-Hellman:

1. Generate random nonce
2. Multiply nonce with G to get the client's public key
→ The client's public key is sent to the PLC
3. Multiply nonce with PLC public key to get the shared secret

```
TD_generate_random_number(0,&nonce,0x18);  
TD_EC_MULT(&client_public_key,&TD_G,&nonce,6);  
TD_EC_MULT(&derived_shared_secret,&server_public_key,&nonce,6);
```



Software PLC Handshake

1. Asymmetric Key Exchange
- 2. Shared Key Derivation**
3. Encryption of Challenge & Symmetric Key

Shared Key Derivation

Generates two 128-bit shared keys from the shared secret.

1. A constant 2x2 matrix M is raised to the x component of the shared secret.
2. The result is encoded as little-endian value and hashed using SHA256.
3. The first 24 bytes of the digest are hashed again using SHA256.
4. The resulting digest is split into two parts. Each part is separately encrypted using a *modified* AES algorithm and returned as a shared key.

Shared Key Derivation

Generates two 128-bit shared keys from the shared secret.

$M_{0,0}$ = 0xa5e873221ea059a595ba61bf27f9cdd5954ef57a747978e2

$M_{0,1}$ = 0x71ded36d796ac873a589cfe8e2831af1297e7e279053186c

$M_{1,0}$ = 0x55136a2069fe9c09984dcb47174c5b77d9c8b4a3db52cd7e

$M_{1,1}$ = 0x5a178cdde15fa65a6a459e40d806322a6ab10a858b868633

Shared Key Derivation

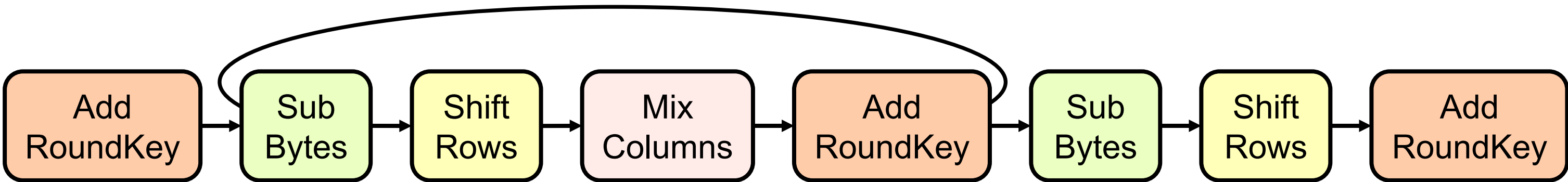
```
TD_matrix_exp_192bit(buffer1,buffer1,shared_secret);  
do {  
    dest = (int *)((int)buffer2 + offset);  
    src = (int *)((int)buffer1[0].value + offset);  
    offset = offset + 0x18;  
    TD_copy_ints(src,6,dest);  
} while (offset != 0x60);  
TD_SHA256_DIGEST(digest,(byte *)buffer2,0x60);  
TD_copy_ints((int *)digest,6,(int *)output);
```

```
TD_SHA256_DIGEST(sha_output,(byte *)sha_input,0x18);  
TD_modified_aes_encrypt(sha_output,output);  
TD_modified_aes_encrypt(auStack_38,output + 0x10);
```

Shared Key Derivation

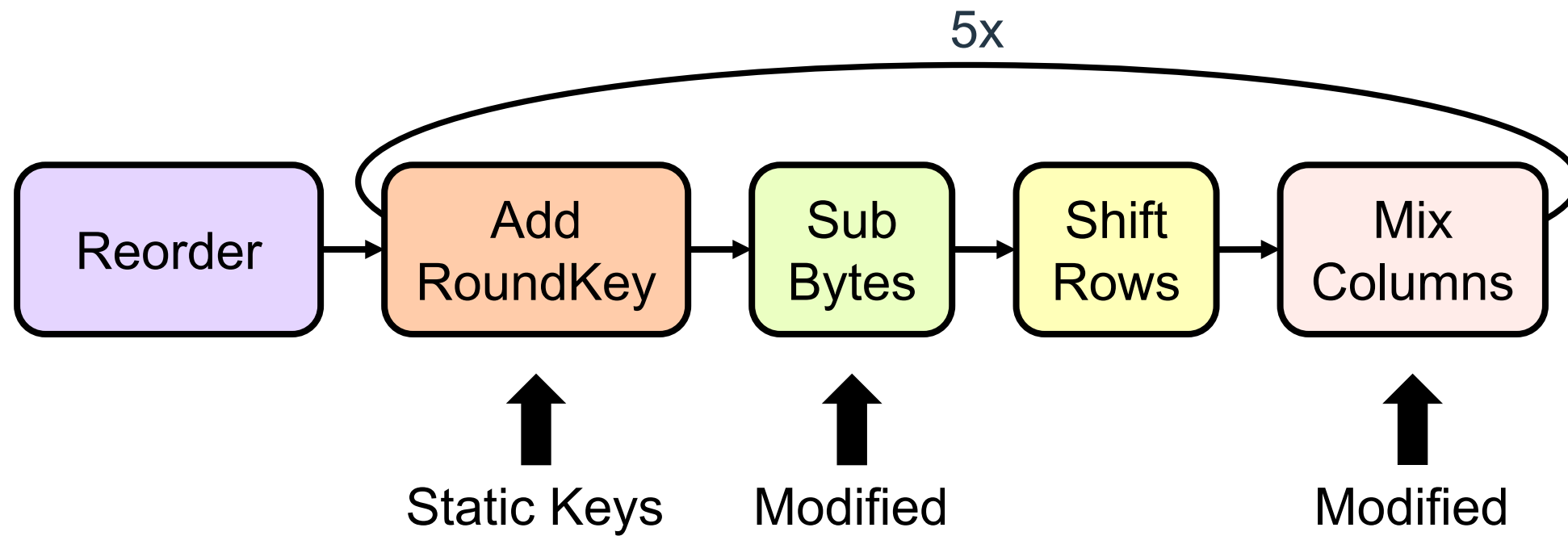
Standard AES

9x



Shared Key Derivation

Modified AES

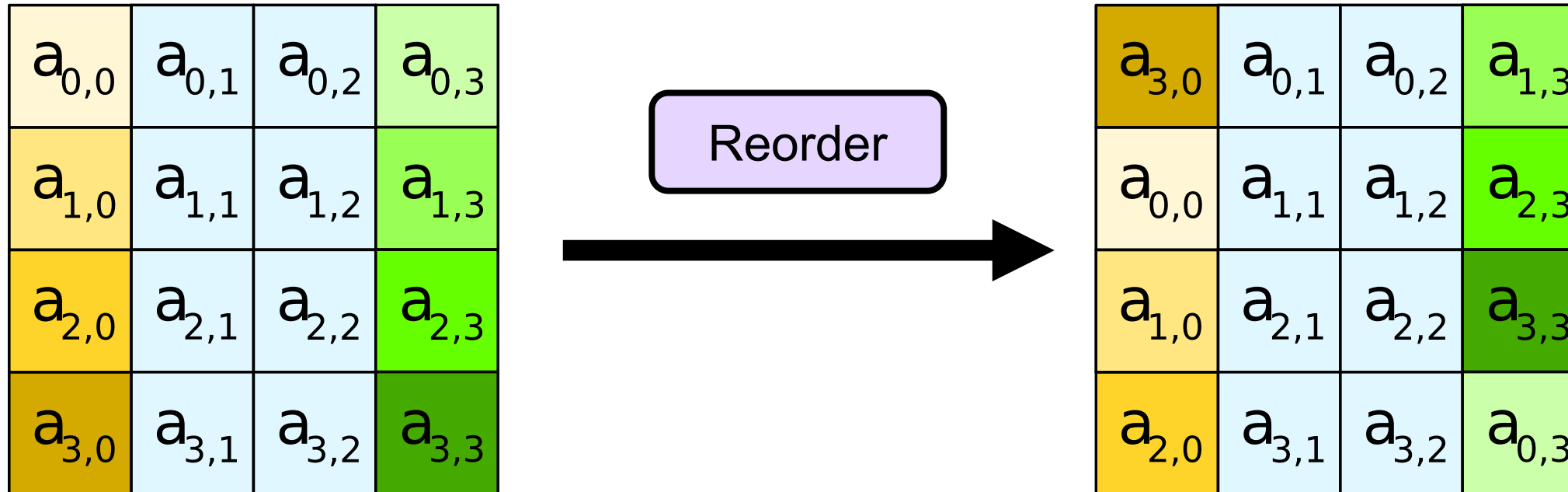


We have AES at Home!



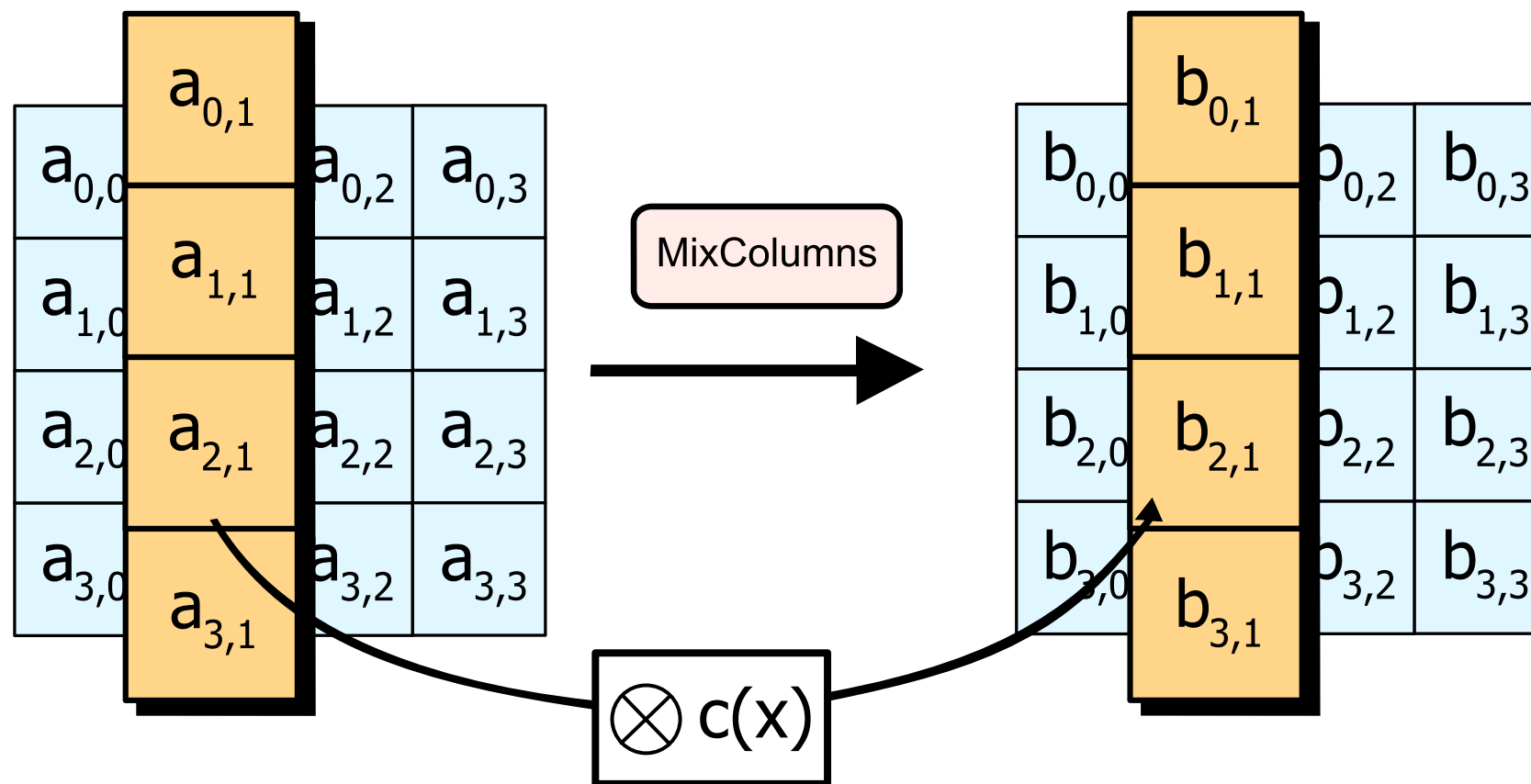
Shared Key Derivation

Added Reorder Step



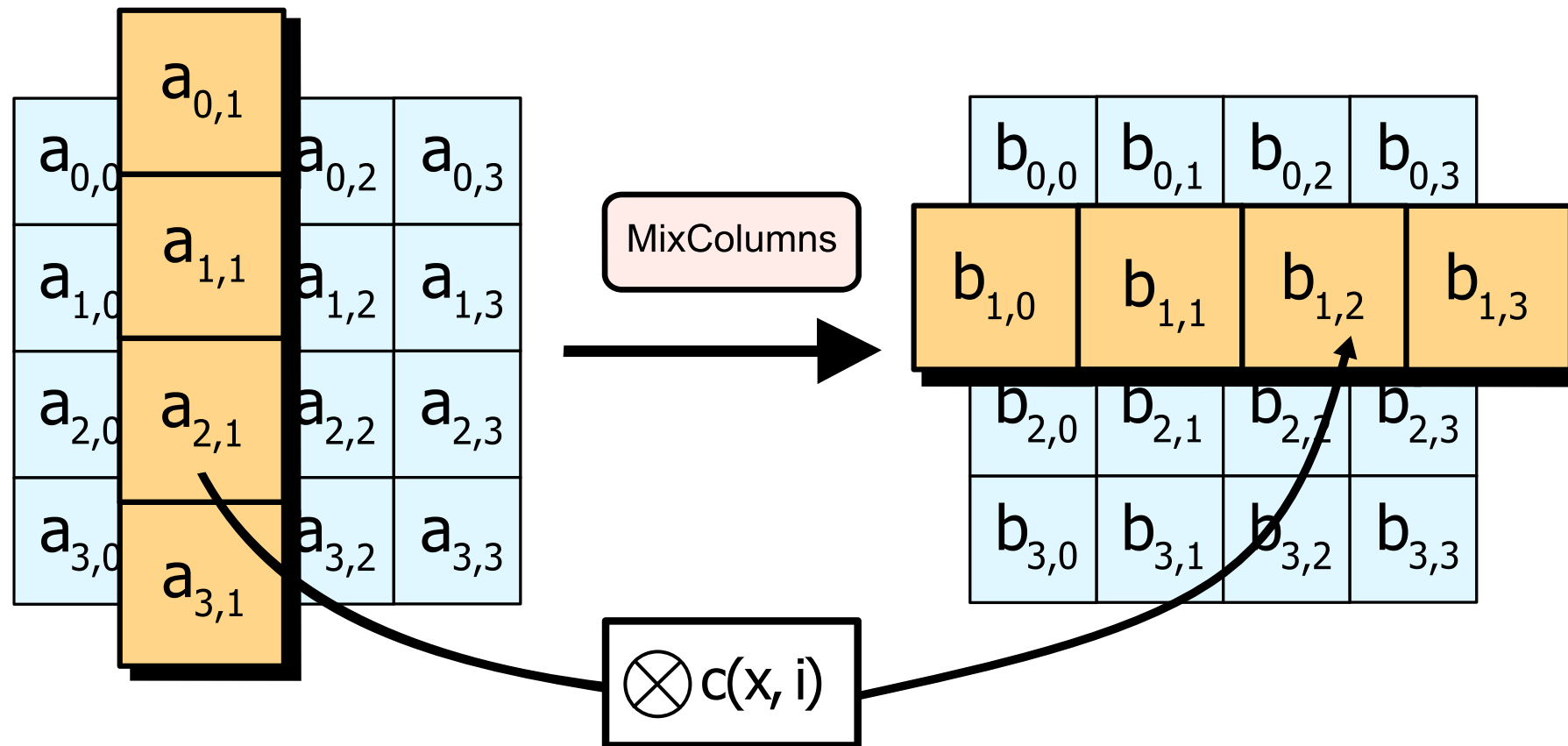
Shared Key Derivation

Standard MixColumns Step



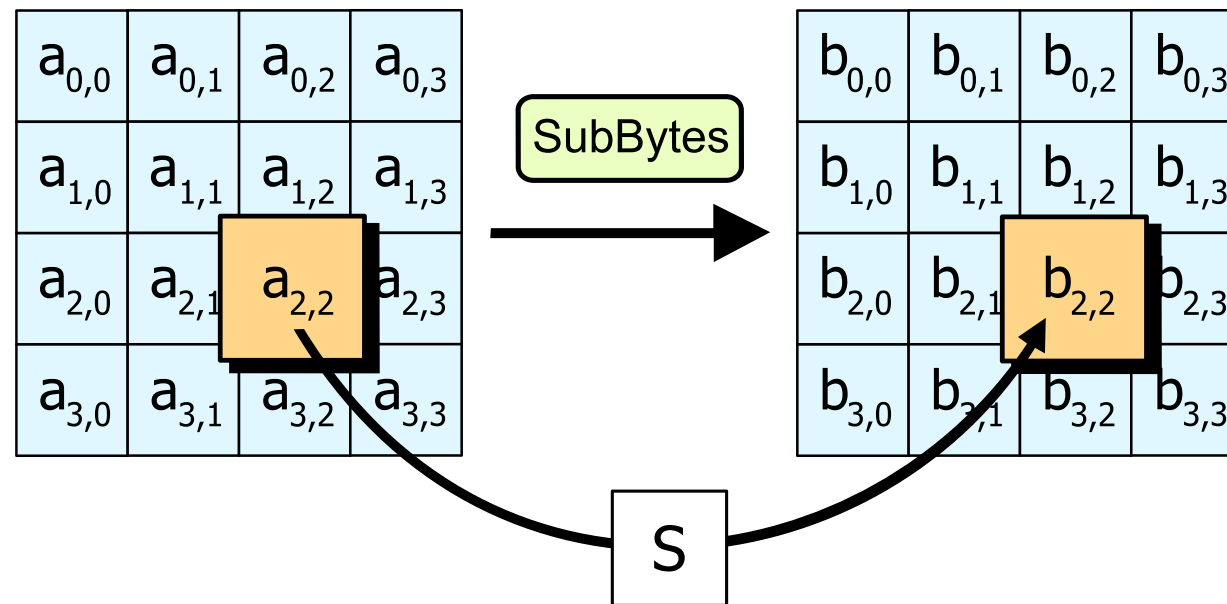
Shared Key Derivation

Modified MixColumns Step



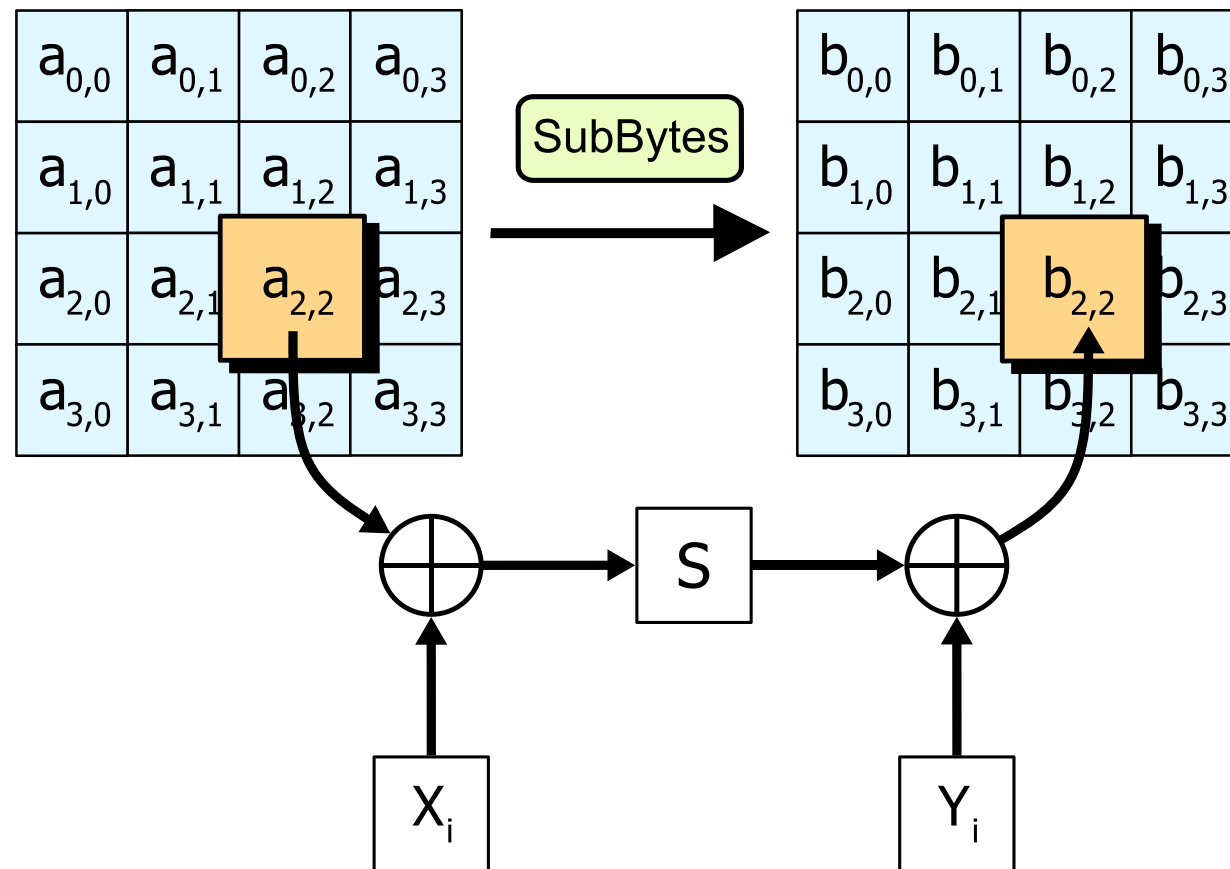
Shared Key Derivation

Standard SubBytes Step



Shared Key Derivation

Modified SubBytes Step





Software PLC Handshake

1. Asymmetric Key Exchange
2. Shared Key Derivation
- 3. Encryption of Challenge & Symmetric Key**

Challenge & Symmetric Key

- The two shared keys are used to transmit another ephemeral AES key
 - AES-encrypt ephemeral key with the first shared key
 - Hash the ciphertext using SHA256
 - AES-encrypt the digest with the second shared key
- Challenge and symmetric key are encrypted with the ephemeral key using AES-GCM



Software PLC Handshake

1. Asymmetric Key Exchange
2. Shared Key Derivation
3. Encryption of Challenge & Symmetric Key

Blob Structure

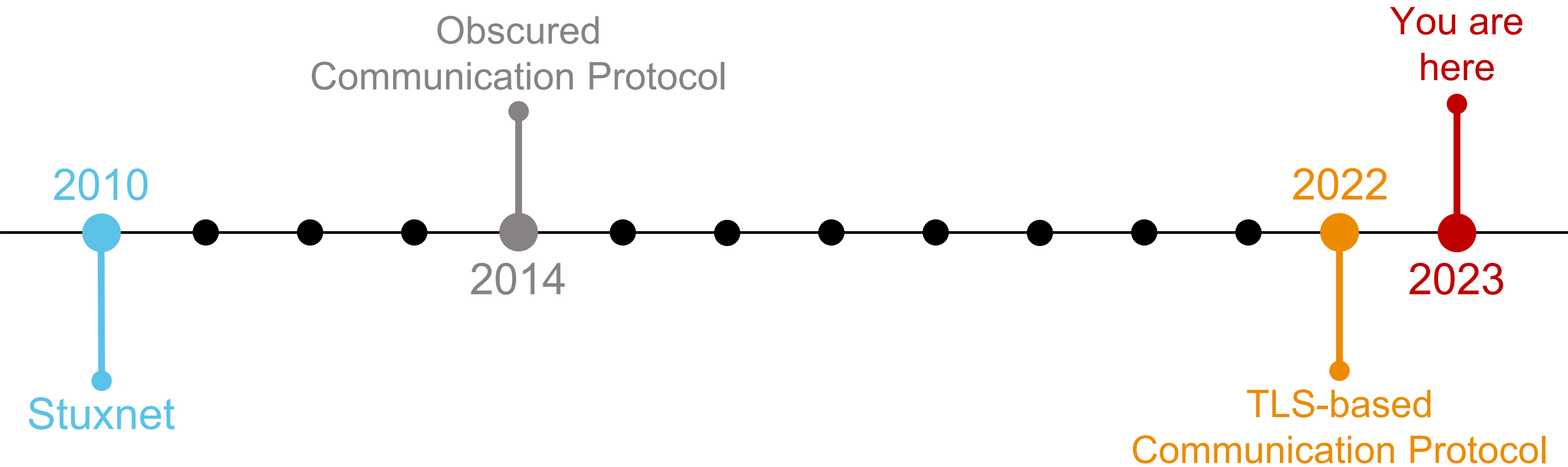
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31								
0xfee1dead				Length (200)				1				1				Symmetric Key Checksum								Symmetric Key Flags															
PLC Public Key Checksum								PLC Public Key Flags								<i>ClientPublicKey_x</i>																							
<i>ClientPublicKey_x</i>								<i>ClientPublicKey_y</i>																															
<i>EncryptedSK</i>																<i>EncryptedDigest</i>																							
<i>FirstIV</i>																<i>EncryptedChallenge</i>																							
<i>EncryptedChallenge</i>																								<i>AuthenticationTag</i>															
<i>AuthenticationTag</i>																																							



What do we learn from all this?



What do we learn from all this?



**We have a cultural,
not a technical problem**

A call to the industry

To PLC vendors:

- Your PLC is a networked computer and potential hacker target.
- Security by obscurity has never been a solution to these threats.
- Get your update processes fixed.

A call to the industry

To machine manufacturers:

- Your machine is a computer and needs regular updates.
- Pass them down to your customers.
- The job is not done after you sold the machine.

A call to the industry

To customers:

- Keep the company and machine networks separated.
- Don't trust your machines to withstand cyberattacks.
- Demand updates from your machine and PLC vendors.

A call to the industry

To fellow researchers:

- Follow our example and share *reproducible* research.
- You have all the tools now to build up on our research.
- Sharing is the only way to advance the state of PLC security.



**Modern automation products are just
embedded computers**

**and they need to be subjected to the same
cybersecurity standards as the rest of the IT industry**



Thank you for your attention!

Colin Finck

✉ c.finck@enlyze.com

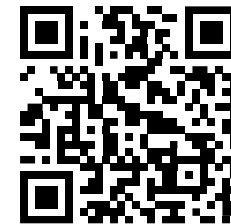
🐦 @ColinFinck

Tom Dohrmann

✉ t.dohrmann@enlyze.com

🐦 @13erbse

Whitepaper at
<https://files.enlyze.com/bheu23>



Shoutout to Alexander Gladis, Manuel 'HonkHase' Atug, German Federal Office for Information Security (BSI), and Siemens for reviewing our paper