Hochschule Konstanz
Fakultät Informatik

Software Security

AIN

Hanno Langweg

06 Secure Software Development Lifecycle

Hochschule Konstanz
Fakultät Informatik

Managing Security Activities in the Software Development Lifecycle

Building Security In Maturity Model – 78 firms

The 12 practices are:

Governance	1. Strategy & Metrics (SM) 2. Compliance & Policy (CP) 3. Training (T)
Intelligence	4. Attack Models (AM) 5. Security Features & Design (SFD) 6. Standards & Requirements (SR)
SSDL Touchpoints	7. Architecture Analysis (AA) 8. Code Review (CR) 9. Security Testing (ST)
Deployment	10. Penetration Testing (PT) 11. Software Environment (SE) 12. Configuration Management & Vulnerability Management (CMVM)

Building Security In Maturity Model – 78 firms

TWELVE CORE ACTIVITIES "EVERYBODY" DOES	
ACTIVITY	DESCRIPTION
[SM1.4]	Identify gate locations and gather necessary artifacts
[CP1.2]	Identify PII obligations
[T1.1]	Provide awareness training
[AM1.2]	Create a data classification scheme and inventory
[SFD1.1]	Build and publish security features
[SR1.1]	Create security standards
[AA1.1]	Perform security feature review
[CR1.4]	Use automated tools along with manual review
[ST1.3]	Drive tests with security requirements and security features
[PT1.1]	Use external penetration testers to find problems
[SE1.2]	Ensure host and network security basics are in place
[CMVM1.2]	Identify software bugs found in operations monitoring and feed them back to development

Comparison with peers

EARTH SPIDER CHART

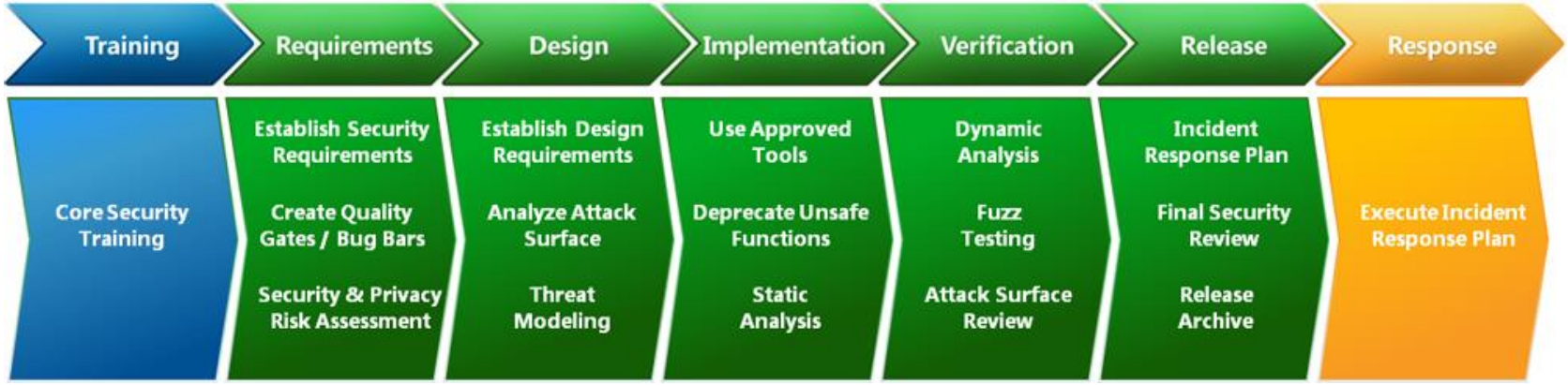
Microsoft's Security Development Lifecycle

MS SDL

- Collection of practices employed and recommended by Microsoft
- First used internally, published in 2008

- Goals
  - Increased reliability of software
  - Reduced maintenance costs

MS SDL: 7 phases



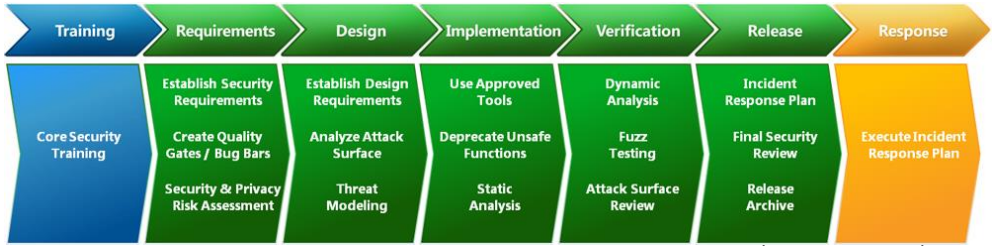
("Phases" no longer promoted - everbody is "agile" today)

1. Training
2. Requirements
3. Design
4. Implementation
5. Verification
6. Release
7. Response

MS SDL: 12 practices

1. Provide training
2. Define security requirements
3. Define metrics and compliance reporting
4. Perform threat modeling
5. Establish design requirements
6. Define and use cryptography standards
7. Manage the security risk of using third-party components
8. Use approved tools
9. Perform static analysis security testing (SAST)
10. Perform dynamic analysis security testing (DAST)
11. Perform penetration testing
12. Establish a standard incident response process

MS SDL: phases/practices



1. Provide training
2. Define security requirements
3. Define metrics and compliance reporting
4. Perform threat modeling
5. Establish design requirements
6. Define and use cryptography standards
7. Manage the security risk of using third-party components
8. Use approved tools
9. Perform static analysis security testing (SAST)
10. Perform dynamic analysis security testing (DAST)
11. Perform penetration testing
12. Establish a standard incident response process

MS SDL: practices/CC

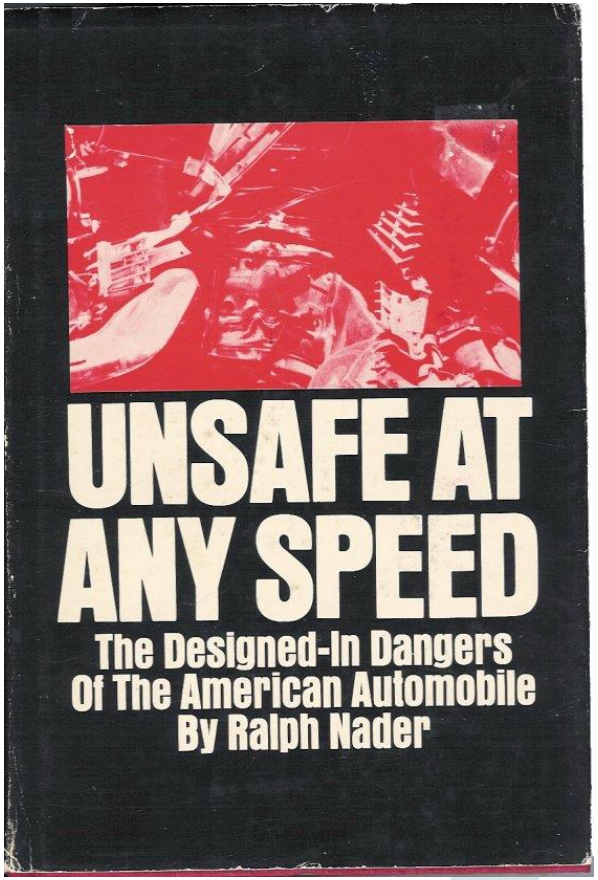
1. Provide training *Site visit*
2. Define security requirements ASE_SPD
3. Define metrics and compliance reporting *Attack potential*
4. Perform threat modeling ASE_REQ
5. Establish design requirements ADV_ARC
6. Define and use cryptography standards ADV_TDS
7. Manage the security risk of using third-party components ALC_DVS
8. Use approved tools ALC_TAT
9. Perform static analysis security testing (SAST) ATE_IND
10. Perform dynamic analysis security testing (DAST) ATE_IND
11. Perform penetration testing AVA_VAN
12. Establish a standard incident response process ALC_FLR

MS SDL: practices/SOFTSEC

1. Provide training SOFTSEC
2. Define security requirements *INITSEC*
3. Define metrics and compliance reporting *INITSEC*
4. Perform threat modeling 02 Offensive Sec., *INITSEC*
5. Establish design requirements 03 Secure Programming
6. Define and use cryptography standards *INITSEC*
7. Manage the security risk of using third-party components 04 SCA
8. Use approved tools 03 Secure Programming
9. Perform static analysis security testing (SAST) 04 SCA
10. Perform dynamic analysis security testing (DAST) 05 Testing
11. Perform penetration testing 02 Offensive Security
12. Establish a standard incident response process *INITSEC*

Traceability, evidence, liability

Liability



- Insecure software has little to **no legal consequences** for supplier (today)
- **This will change** (when?) – why should IT be special?

- Liability requires **traceability**
  - Development – who who modified it?
  - Deployment – where did it come from?
  - Operation – who configured it?
- Attribution, proactive forensics, debugging

Liability: Version control as evidence

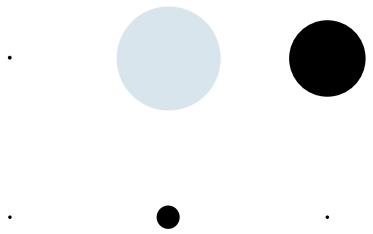
- **Everything necessary to reproduce** artefact
  - Source code: application, libraries; compilers?
  - Images, translations, initial values, deployment configuration
- **Version control** systems
  - Log modifications, history
  - Roll back mistakes, attacks
- Applicable
  - Traditional development
  - Configuration of hosted services, infrastructure as code

Build process

- Transformation source code ➔ binary
- Decision about source artefacts
  - Source code: files, versions, libraries, references
  - Compiler(s), compiler configuration
  - Data compiled into binary file
- Build environment
  - Operating system
  - Cached artefacts
- Output: deliverable binary package

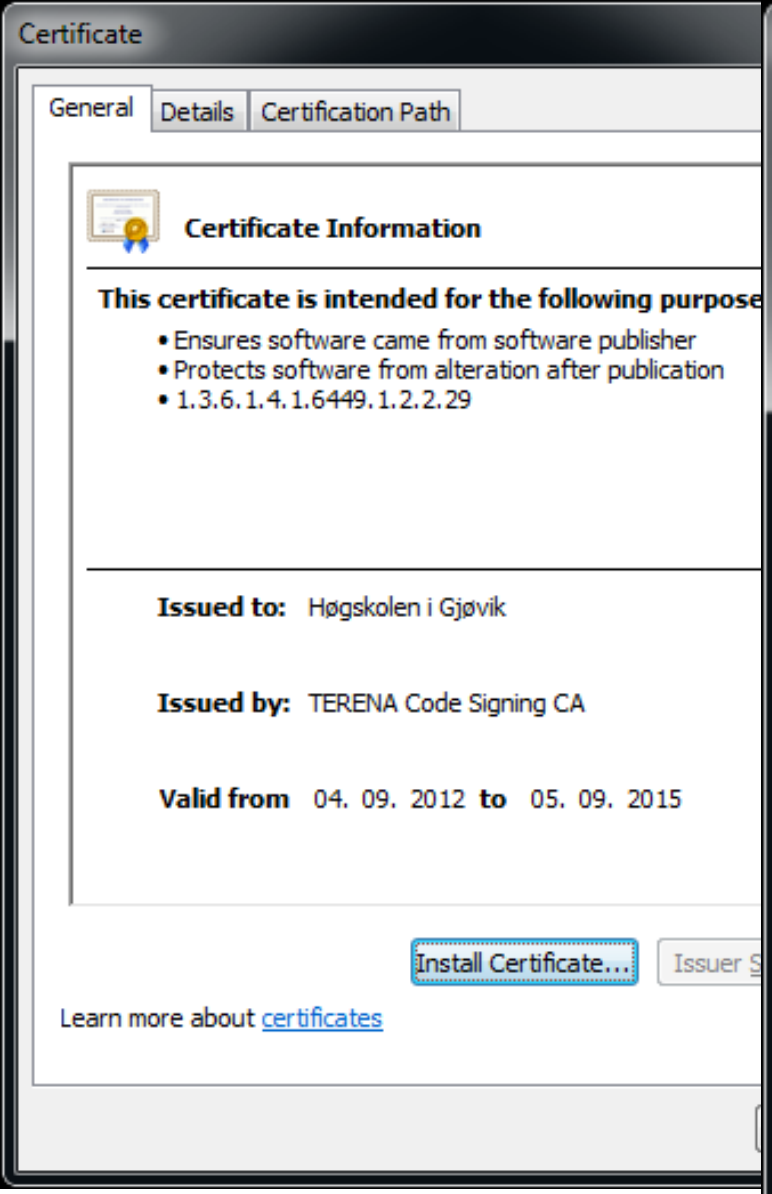
Access to *build process* input

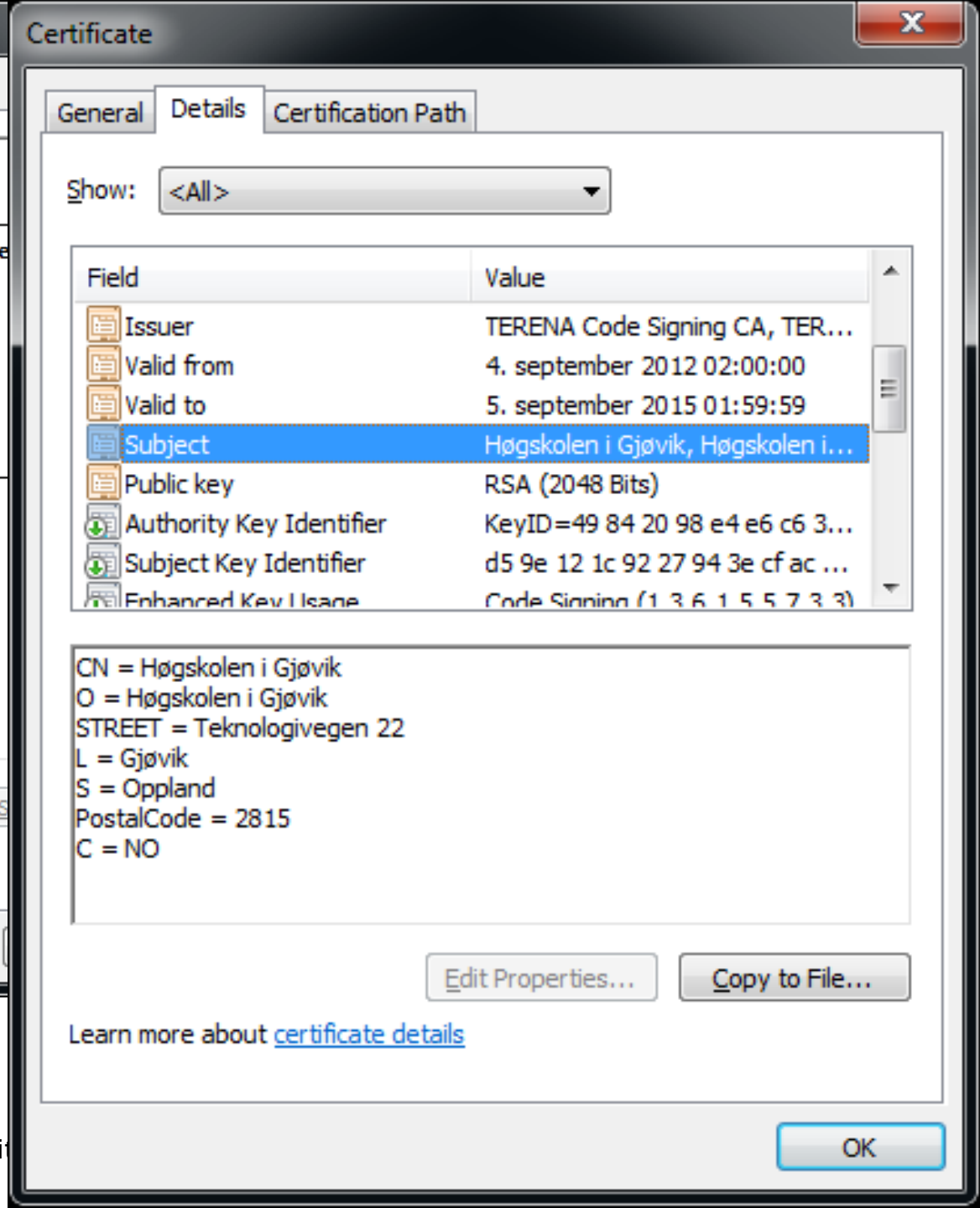
- Developers
- Designers, translators
- Software integrators
- IT operations staff
- Marketing?

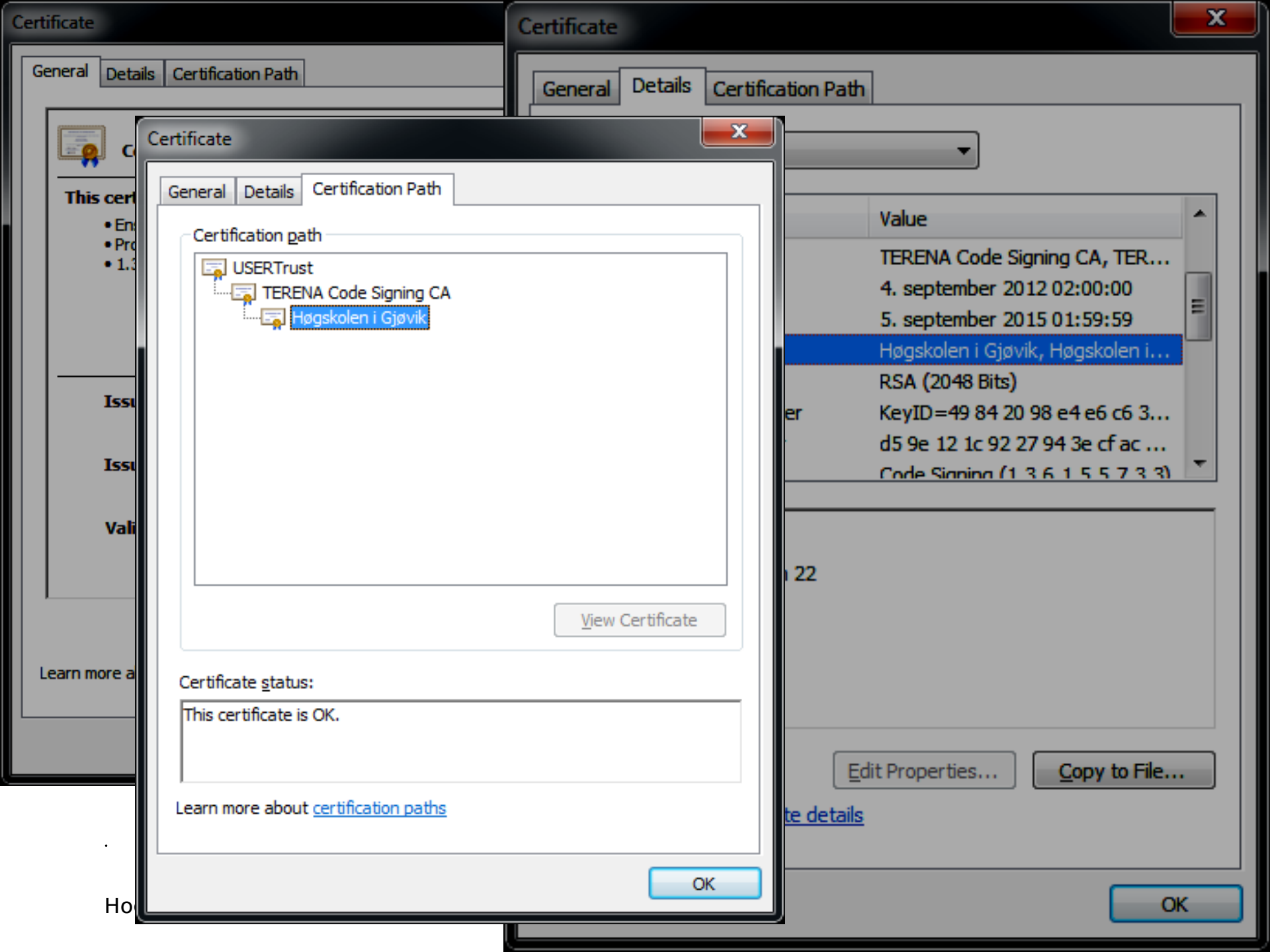


Code signing

- Prove origin (authenticity) + integrity of transfer
- Chain of trust: Where is the root?
  - Verisign? Symantec? DigiCert? Thawte? Buypass? UserCert? Deutsche Telekom?
  - In-house? (And what to check before signing in-house?)
- Validity of certificate

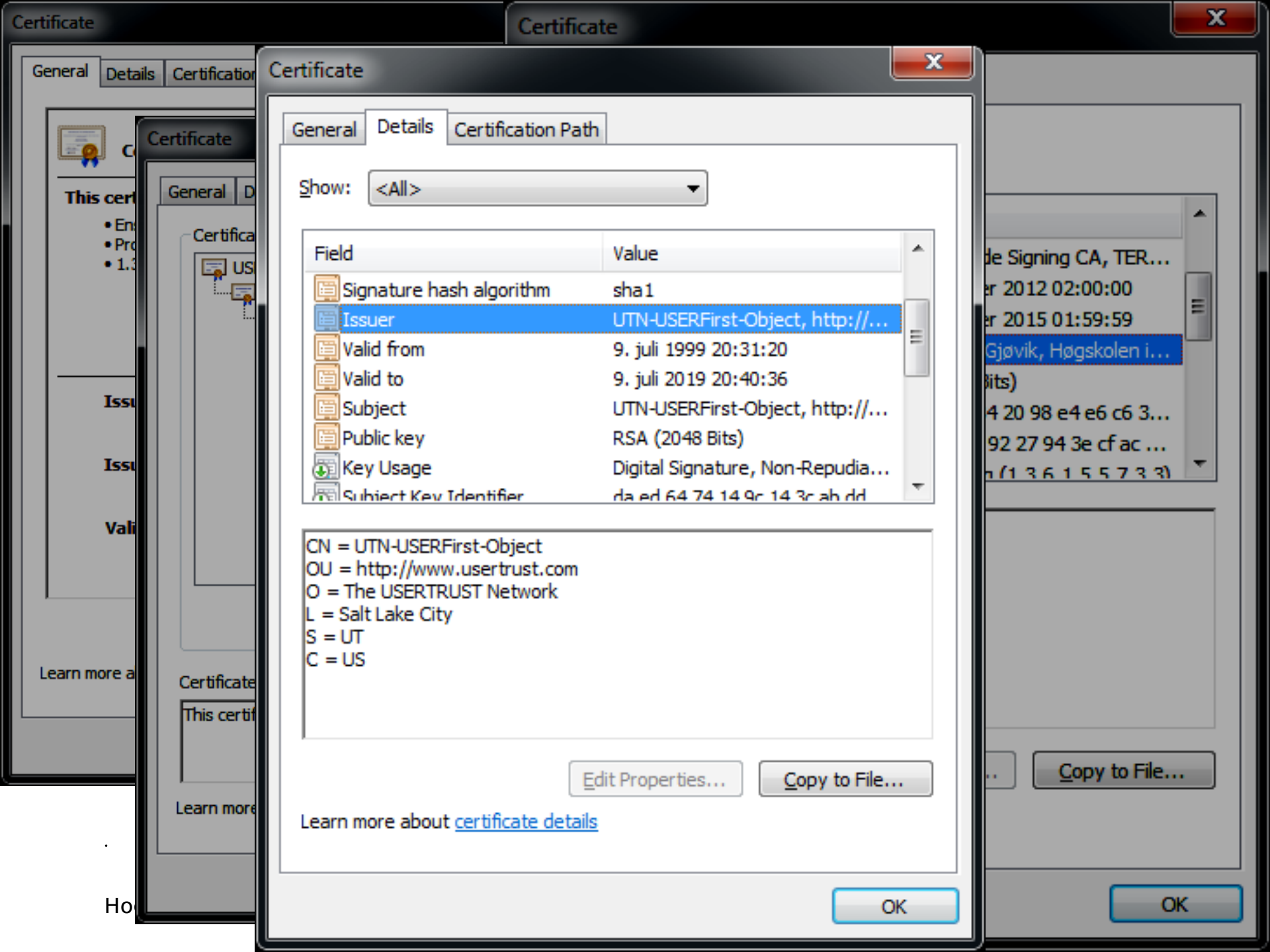

Certificate
General
Details
Certification Path
Certificate Information
This certificate is intended for the following purpose
• Ensures software came from software publisher
• Protects software from alteration after publication
• 1.3.6.1.4.1.6449.1.2.2.29
Issued to: Høgskolen i Gjøvik
Issued by: TERENA Code Signing CA
Valid from 04. 09. 2012 to 05. 09. 2015
Install Certificate...
Learn more about certificates


Certificate
General
Details
Certification Path
Show: <All>
Field
Value
Issuer
TERENA Code Signing CA, TER...
Valid from
4. september 2012 02:00:00
Valid to
5. september 2015 01:59:59
Subject
Høgskolen i Gjøvik, Høgskolen i...
Public key
RSA (2048 Bits)
Authority Key Identifier
KeyID=49 84 20 98 e4 e6 c6 3...
Subject Key Identifier
d5 9e 12 1c 92 27 94 3e cf ac ...
CN = Høgskolen i Gjøvik
O = Høgskolen i Gjøvik
STREET = Teknologivegen 22
L = Gjøvik
S = Oppland
PostalCode = 2815
C = NO
Edit Properties...
Copy to File...
Learn more about certificate details
OK

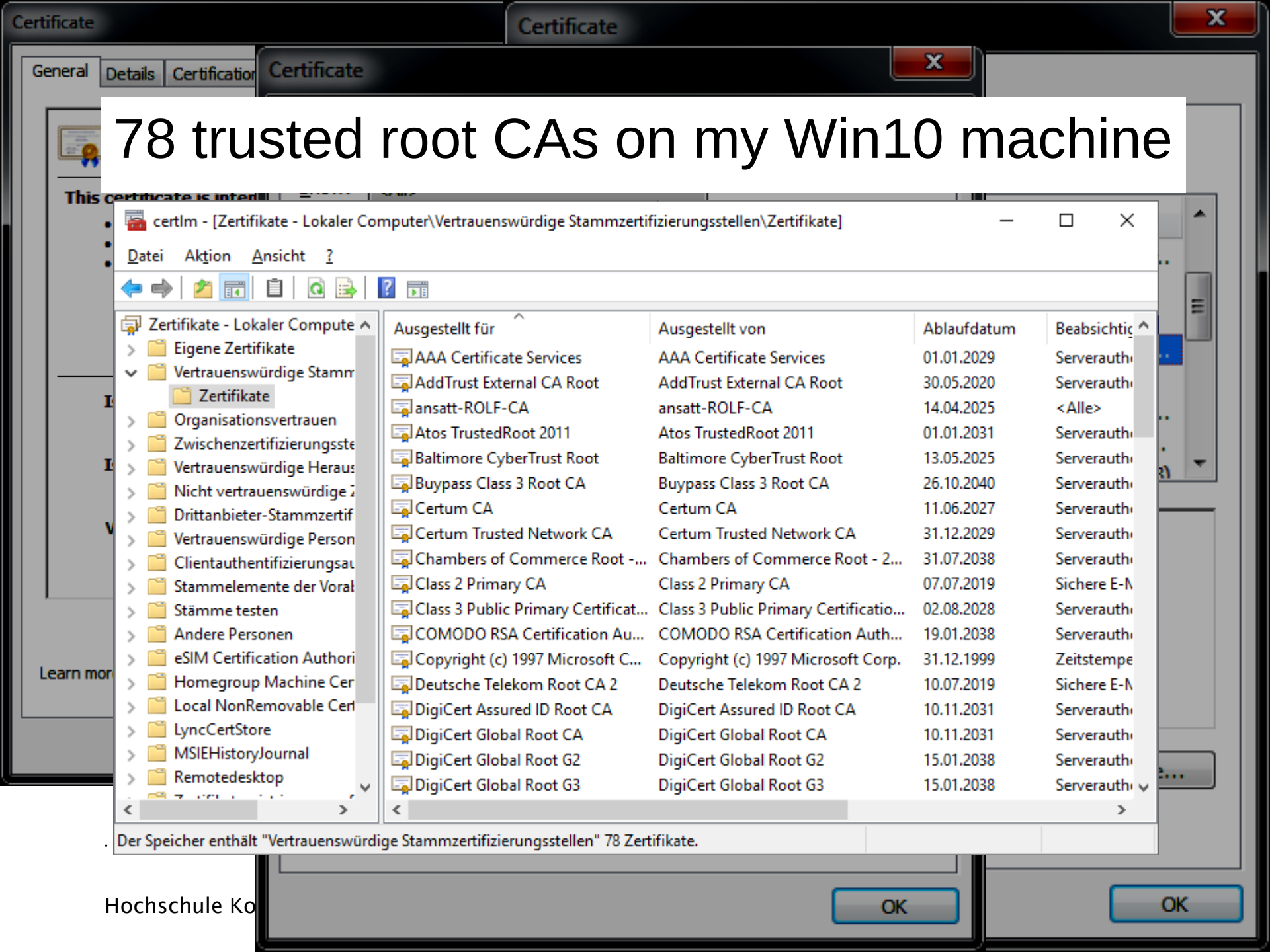

Certificate
General
Details
Certification Path
Certification path
USERTrust
TERENA Code Signing CA
Høgskolen i Gjøvik
View Certificate
Certificate status:
This certificate is OK.
Learn more about certification paths
OK
Value
TERENA Code Signing CA, TER...
4. september 2012 02:00:00
5. september 2015 01:59:59
Høgskolen i Gjøvik, Høgskolen i...
RSA (2048 Bits)
KeyID=49 84 20 98 e4 e6 c6 3...
d5 9e 12 1c 92 27 94 3e cf ac ...
Edit Properties...
Copy to File...
OK

.

Ho


Certificate
General
Details
Certification Path
Show:
<All>
Field
Value
Signature hash algorithm
sha1
Issuer
UTN-USERFirst-Object, http://...
Valid from
9. juli 1999 20:31:20
Valid to
9. juli 2019 20:40:36
Subject
UTN-USERFirst-Object, http://...
Public key
RSA (2048 Bits)
Key Usage
Digital Signature, Non-Repudia...
CN = UTN-USERFirst-Object
OU = http://www.usertrust.com
O = The USERTRUST Network
L = Salt Lake City
S = UT
C = US
Edit Properties...
Copy to File...
Learn more about certificate details
OK

78 trusted root CAs on my Win10 machine

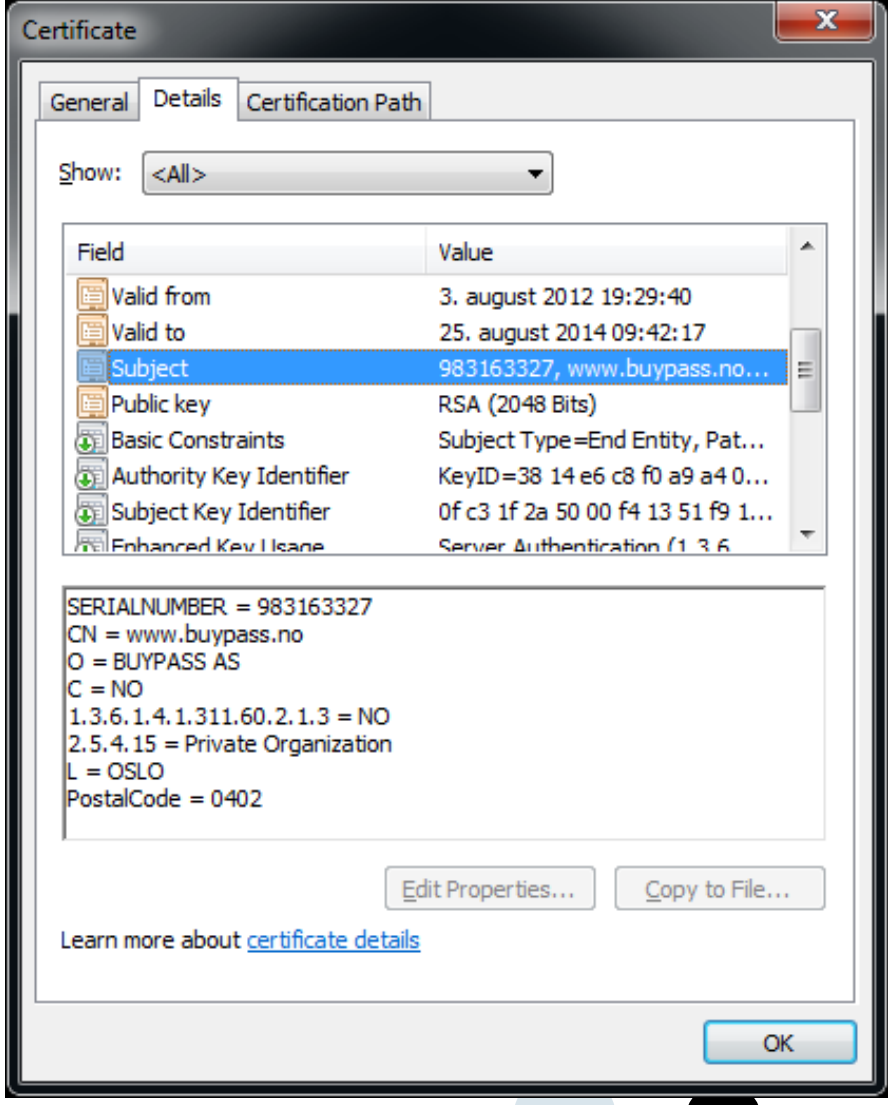


Hochschule Ko

Extended validation

- Extended Validation (verification of existence + authorization)

  Links to **legal entity**, e.g. identified by
  - Company register number; local identifier relative to district court (in Germany)
  - Organisation number (in Norway)

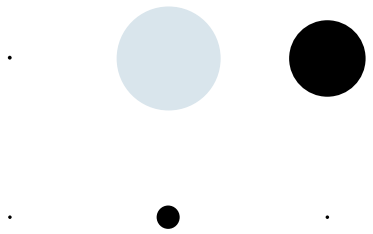


Code signing

- Prove origin (authenticity) + integrity of transfer
- Who signs?
  - Developers? (How to verify authority?)
  - Companies? (Who has access to keys?)
  - Value of signed code – liability?
- Only EV Extended Validation provides appropriate traceability to legal entity

Integration of code signing

- Time span between artefact creation and signature creation
- Access to signing keys
  - Certificate files, hardware tokens, PIN/password
- Inclusion of timestamp
- Automation of distribution
  - App stores

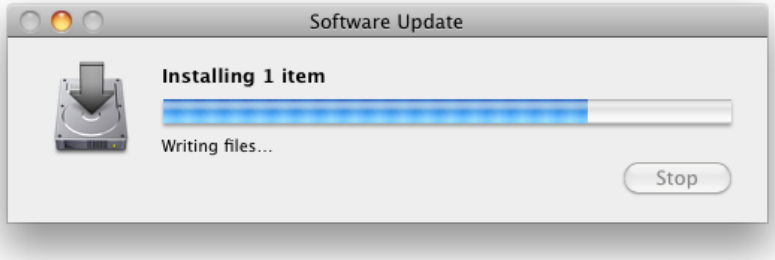


Binary package transfer

- **Delivery** to device
  - Personal computer, server, cloud service, smart phone
  - Heating controller, TV, washing machine, car, missile
- Integrity of **transfer**
- Authenticity of **origin**
- **Acceptability**
  - Trust relationship to origin
  - Evaluation, review, static analysis
  - Proof-carrying code: verifiable promises about binary (hard to specify)

Installation/setup

- Only time to run with administrative privileges
  - Configure target system
  - Use access control mechanisms,
    do not rely on user/administrator
- Configure access control mechanisms
- Verify integrity, i.e., success of installation process



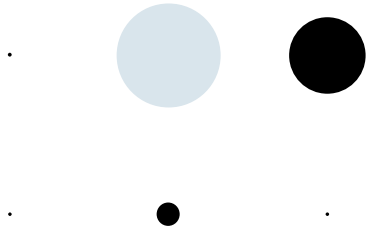
Installation/setup

- Modification of client system might change attack surface
  - File system changes
  - New user accounts, services, open network ports
  - Certificates
  - Configuration files, registry changes

- Tool for monitoring of changes e.g. Attack Surface Analyzer: https://github.com/microsoft/attacksurfaceanalyzer

Integrity verification

- Verification of installed modules
  - Compare with reference
- Verification tool
  must not be compromised
- Time of verification
  - After installation
  - On startup
  - Periodic
- Example: signature creation software



Summary

- BSIMM Building Security In Maturity Model
  - Collection of commonly applied practices
  - Focus mainly on large enterprises
  - Enables comparison with peers
- Microsoft's Security Development Lifecycle (SDL)
  - Useful collection of practices for secure software development
  - Partially supported with tools (from Microsoft)
  - Similar practices appear elsewhere (other frameworks, other companies)
- Traceability, evidence, liability
  - Need to document correct software construction, deployment, execution
  - Integrity and authenticity in business often more important than confidentiality