



Designing Secure Security Applications

Capabilities and Limitations of Cryptography

Danny De Cock

Danny.DeCock@esat.kuleuven.ac.be
Katholieke Universiteit Leuven/Dept. Elektrotechniek (ESAT)
Computer Security and Industrial Cryptography (COSIC)
Kasteelpark Arenberg 10
B-3001 Heverlee
Belgium

Outline

- Typical Crypto Applications
- Typical Security Issues
- Benefits of Secure Software Development?
- Core Security Problems
- Secure vs. Security
- Two Examples
- So what?
- Good practices...

Typical Crypto Applications

- Many applications rely on cryptographic primitives and use security protocols:
 - e-Banking & e-Payments: DigiPass, Proton, Isabel,...
 - Integrity protection: Code signing, Electronic signatures
 - Confidentiality protection: Encrypted file systems, email,...
 - Wireless: GSM, Bluetooth, ZigBee, WiFi,...

Typical Security Issues

- Errors in the cryptographic primitives are rare causes of non-secure security applications
 - Bad use of cryptographic primitives/protocols kills the application
- Causes of Major security flaws
 - Everybody believes he/she is a cryptographer ☹️
 - Very primitive key management
 - User's lack of security awareness

Benefits of Secure Software Development

- Application security
 - Important emerging requirement in software development
 - Controls potential
 - Severe brand damage
 - Financial loss
 - Privacy breaches
- Risk-aware customers (financial institutions, governmental organizations) want to
 - Assess the security posture of products they build or purchase
 - Plan to ultimately hold vendors accountable for security problems in their software
 - Procure reliable and secure software
 - Hold vendors accountable for security problems in software

Core Security Problems

- Software development lifecycle does not deal well with security:
 - Software developers lack structured guidance
 - Books on the topic are
 - Relatively new
 - Collections of good practices
- Security is not a feature that demos well
 - Developers tend to focus on core functionality features
- Security is addressed in an ad hoc manner during development
 - Developers typically provide a minimal set of security services given their limited security expertise
- Applications are too complex to comprehend

Secure vs. Security Software Development

■ Secure software

- Application acts according to its specifications
- Provable features of the application
- Software design is the bottleneck

■ Security software

- Relies on secure software
- Application uses secret and private information
 - Electronic payments, voting, signing,...
 - Protection of privacy, confidentiality, integrity,...
- Critical use of the user/device/... credentials

SSL/TLS Example

- Fact: SSL/TLS is the most popular way to provide data confidentiality and integrity services for data in transit
 - Drop-in for traditional sockets
- But: Most SSL/TLS deployments are susceptible to network-based attacks
 - Technology is widely misunderstood
 - Self-signed server certificates
 - No decent validation of server certificates
 - Insecure client configuration

“Our system is secure: we use the AES”

- What about
 - Key management
 - “Random” keys?
 - Authenticated (?) key agreement
 - Implementation
 - Modes of encryption, initialization vectors,...
 - Attacking the implementation
- Who holds the keys?
 - Who can use the keys?
 - Stored in the clear?
 - Key archives?

What to do about it?

- Large software vendors make lots of effort
 - Ongoing effort to improve security through its development process
 - Involves training and process improvements
 - Good practices:
 - Initial approach: freezing the current status
 - Only allow change that improve the security
- Good system design relies on embedded security
 - Simplifies security issues: no add-on
 - Hides complexity of cryptographic protocols

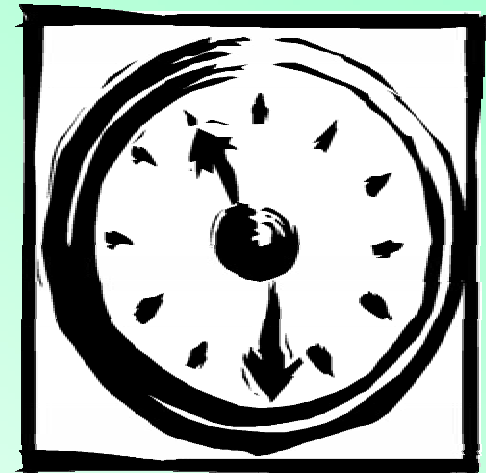
Good Practices

- Centralize security knowledge in software architects and application designers
 - Implementers should not have to make delicate security decisions
 - Cryptographic algorithms and protocols should be considered as modular building blocks
 - Consistent deployment of a security vision saves time and money
 - Security expertise concentrated in a few of the most trusted members of the development organization
 - Allows for better depth of knowledge
 - Results in more effective and secure results
- Good initial security design avoids hard to solve security issues
 - Security patches do not deal with inherent design flaws
 - Simple design is easily understandable/testable/auditable

Questions?

Myself Danny.DeCock@esat.kuleuven.ac.be
<http://www.godot.be>

Yourself <https://www.mijndossier.rrn.fgov.be>
<https://www.mondossier.rrn.fgov.be>
<https://www.meindossier.rrn.fgov.be>



Google™ keywords: “godot eID”