



Internet Security



Prof. Anja Feldmann, Ph.D.

anja@net.t-labs.tu-berlin.de

Prof. Dr. Jean-Pierre Seifert

jpseifert@sec.t-labs.tu-berlin.de

Dipl.-Ing. Benjamin Michéle

ben@sec.t-labs.tu-berlin.de

Topic: RSA Signatures

- ❑ Why?
- ❑ How?
- ❑ Secure or epic failure?
- ❑ Tools you should know ...

- ❑ Focus: Hands on experience aka *dirty details!*

Why Signatures?

- ❑ Example: DSL router at customer's home
 - New feature developed / security hole found
 - Provider wants to deliver the new binary
 - UI on router has option to flash image

- ❑ So where's the problem?

Why Signatures?

- ❑ Provider only wants his software to run on device
 - No virus, no malicious user software, no nada
- ❑ Signature
 - Binary image is signed by provider
 - Signature is checked by router

How Signatures?

- ❑ Symmetric encryption???
 - Bad idea
 - Why?

- ❑ Get read only access to router file system
 - Get the key
 - Sign images for all routers
 - Live happily ever after

- ❑ Not going to happen?
 - You'd be amazed ...

How Signatures?

- ❑ Asymmetric encryption using RSA key pairs
 - Of course ... but how, exactly?

- ❑ Provider signs binary image with **private** key
- ❑ Router checks signature with **public** key

- ❑ Compromise one router
 - Get the public key
 - And ... ahhh ... do nothing ☹ / ☺

How Signatures?

- ❑ So we're done? Goodbye?

- ❑ Well ... nope! It's all about the details!
 - How do we create a signature?
 - How do we check it?

Signatures: Prerequisites

❑ RSA example

- Private exponent d , 2048bit length (only @Provider)
- Public exponent $e=3$ (@Routers)
- Modulus n , 2048bit length (@both)

❑ Considered secure (depends)

Signatures: Trivial #1

- ❑ Provider signs binary b
 - Message $m = \text{"42"}$
 - Signature $s = m^d \pmod{n}$
- ❑ Router checks s
 - Only if $m' = s^e \pmod{n} == 42$ will router accept b
- ❑ Obvious flaw!

Signatures: Try #2

❑ Provider signs binary b

- Message $m = \text{sha1sum}(b)$
- Signature $s = m^d \pmod{n}$

❑ Router checks s

- Only if $m' = s^e \pmod{n} == \text{sha1sum}(b)$ will router accept b

❑ Better?

- If s is small, then $s^3 < n$
 - Let $s' = \text{sha1sum}(b')$, fake signature $s'' = \text{cubic root of } s'$
 - Then $s''^e \pmod{n} = s''^3 = s' == \text{sha1sum}(b')$
 - Router falsely accepts the forged signature s'' and hence b'

Signatures: Try #3

❑ Pad message

- PKCS#1 v1.5
- $m' = 00\ 01\ FF\dots FF\ 00 \mid c_sha \mid sha1(b) \rightarrow 2048\ bit$
- Encrypt m' instead of m

❑ Ok ... now we're done, right?

Signatures: Try #3

- ❑ Pad message
 - PKCS#1 v1.5
 - $m' = 00\ 01\ FF\dots FF\ 00 \mid c_sha \mid sha1(b) \rightarrow 2048\ bit$
 - Encrypt m' instead of m

- ❑ Ok ... now we're done?
 - Not really, what about the implementation?

- ❑ Let the games begin ... 😊

Practical Example: 25c3: Wii [1]

- ❑ Fail #1: Wii ignores padding
- ❑ Fail #2: Wii uses strncmp
- ❑ Why fail?

- ❑ Attack
 - Signature $s' = 0$
 - $s' \wedge e = 000000000000000000000000000000000000...$
 - Manipulate binary b' until $\text{sha1sum}(b') = 0x00*****$
 - $\text{Strncmp}(0, 0x00***** , 20) \rightarrow \text{TRUE}$

- ❑ [1] http://marcansoft.com/uploads/25c3_console_hacking/

Certificate xxxxxx

- ❑ C builds routers and sells to Provider P and Q
- ❑ Provider P and Q give them to customers
- ❑ P wants to sign software for the router
 - C does not want to give out private key
 - How to solve?

Certificate chains

- ❑ Router stores C' s public key
- ❑ C signs a public key from P with its private key
- ❑ P signs new software with its own private key
- ❑ Router checks signature
 - Check signature of P' s public key with C' s public key
 - Check signature of software with P' s public key
- ❑ In other words
 - Validate certificate chain
 - Use certificate to validate software signature

Certificate chains

- ❑ Perfect! That's it, I'm out! Goodbye ...

- ❑ Wait a minute! Implementation details!!!
 - What to sign?
 - How?
 - Certificate structure?
 - Simplification of ASN1?

Certificate chains

- ❑ Private exponent d kept secret
 - Public exponent e = 65537
 - Modulus n = 2048 bit
 - Signature s = 2048 bit
- ❑ $s' = 0001FF..FF00|ASN1(sha1(n))$
- ❑ $s = s'^d \pmod n$
- ❑ Certificate cert = $e|n|s$
- ❑ Verify: $s^e \pmod n == s'$
- ❑ Secure?

Certificate chains

- ❑ Attacker replaces $e=65537$ with $e=1$
 - Remember, only n was signed
- ❑ $s^e \pmod n = s^1 \pmod n = s$!!!
- ❑ Provide fake signature in cleartext!

- ❑ And so on and so on ... More fun to come!

Certificate chains

- ❑ Obvious flaw, but if closed system?
 - Secure, because nobody knows?

- ❑ Never use security by obscurity! Never.
- ❑ Publish the spec of your security system
- ❑ Let others review it

Rollup

- ❑ Think this doesn't happen in reality?
 - We see it all the time!
- ❑ Think this won't happen again?
 - Oh yes it will. Trust us ...
- ❑ Interested in working on these topics as a seminar/project/bachelor/master thesis?
 - Contact anyone from our group or write me an email
 - ben@sec.t-labs.tu-berlin.de

Tools

- ❑ Actually I wanted to show some tools
- ❑ Check them out yourself
 - openssl asn1parse/x509/genrsa/rsa
 - hexedit
 - sha1sum
 - verify signatures by hand using python
 - $s = \text{"\%0512x"} \% \text{pow}(m, d, n)$
 - $m2 = \text{"\%0512x"} \% \text{pow}(s, e, n)$
 - Use these tools with your own certificates and keys
 - Use them with certificates from the Internet
 - Read about RSA key generation

Extended Euclidean Algorithm in Python

```
def extgcd(a, b):
```

```
    u=t=1
```

```
    v=s=0
```

```
    while b>0:
```

```
        q=a//b
```

```
        a, b = b, a-q*b
```

```
        u, s = s, u-q*s
```

```
        v, t = t, v-q*t
```

```
    return a, u, v
```

```
(x1,x2,d)=extgcd(e, phi(n))
```

```
http://www.inf.fh-flensburg.de/lang/krypto/algo/euklid.htm
```

*Thank You for
Your Attention!*