C.b) RSA with Applications

C.31 RSA

The RSA algorithm was introduced by Rivest, Shamir and Adleman in 1977.

- p,q large primes (to be kept secret)
- n := pq modulus (publicly known)
- d *secret key* (*private key;* to be kept secret) with gcd(d, φ(n))=1
- e *public exponent* (publicly known);

  $e \equiv d^{-1} \pmod{\varphi(n)}$

Note: The *public key* is the pair (n,e).

C.31 (continued)

Fact: $(x^d (\text{mod } n))^e \equiv x \ (\text{mod } n)$ and

$(x^e (\text{mod } n))^d \equiv x \ (\text{mod } n)$ for all $x \in Z_n$.

In other words: $x \rightarrow x^d \ (\text{mod } n)$ and $x \rightarrow x^e \ (\text{mod } n)$ define inverse bijections on Z_n.

Proof of the fact: Exercises (Hint: Use the CRT)

Note: For $x \in Z_n^*$ the fact follows immediately from Euler's Theorem (C.10).

C.32 Example

<u>RSA with artificially small parameters:</u>

$$\left.\begin{array}{l} p=11 \\ q=23 \end{array}\right\} \; n=243 \,\} \; \varphi(243)) = \varphi(11*23) = 10*22 = 220$$

$e=3$ } $d=147$ (Note that $ed = 441 \equiv 1 \pmod{220}$.)

C.33 RSA: Fields of Application

- (different types of) digital signatures
- key exchange protocols for symmetric keys
- hybrid protocols
- communication protocols (SSL,TLS etc.)
- Home banking, e-commerce
- Credit cards (chip), GeldKarte (internet usage)
- …

Remark: In this section we will discuss several applications in detail.

Note: The RSA algorithm is by far the mostly widespread public key algorithm.

C.34 Efficiency

If d is in the same order of magnitude as n (usual case, cf. C.38) the s&m algorithm (C.6) requires about

- $\log_2(n)$ modular squarings
- $0.5*\log_2(n)$ modular multiplications

of $\log_2(n)$-integers to compute y^d (mod n).

Note: In general asymmetric algorithms need much more computation time than symmetric ciphers.

C.35 RSA with CRT

Usually RSA implementations use the CRT (C.27) to compute $y^d \pmod n$.

<u>Setup Step (to be carried out once):</u>

Compute

$d_p := d \pmod{(p-1)}$

$d_q := d \pmod{(q-1)}$

Determine integers N_p and N_q with

$N_p \equiv 1 \pmod p$ $\qquad N_q \equiv 0 \pmod p$

$N_p \equiv 0 \pmod q$ $\qquad N_q \equiv 1 \pmod q$

C.35 (continued)

Computation Step:

$x_p := y(\text{mod } p)^{d\_p}$ (mod p)

$x_q := y(\text{mod } q)^{d\_q}$ (mod q)

$y^d \equiv N_p x_p + N_q x_q$ (mod n).

C.36 RSA with CRT: Efficiency

If d is in the same order of magnitude as n (usual case, cf. C.38) the CRT with the s&m algorithm (C.6) requires about

- $\log_2(n)$ modular squarings
- $0.5*\log_2(n)$ modular multiplications

of $0.5*\log_2(n)$-integers to compute y^d (mod n).

Note: For identical hardware the CRT reduces the computation time to about 25 %.

C.37 Recovery Attack on the Secret Key

Goal: Determine d from (n,e).

Fact: If the adversary knows the factorization n=pq he concludes $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$.

Then he computes

$d \equiv e^{-1} \pmod{(p-1)(q-1)}$

with the extended Euclidean algorithm.

$\rightarrow$ RSA is broken

Note: For that reason factorization algorithms have intensively been studied over the last 25 years.

C.38 Selection of the Parameters

- To reduce the computation time the designer clearly preferred small parameters n and d.
- However, to prevent factorization today usually 1024 bit to 2048 bit moduli n are used. The prime factors p and q are of the same order of magnitude (although they should not be too close together!).
- Attention: If $d < n^{0.29}$ the secret key d can be found with lattice-based attacks.

C.38 (continued)

- After the modulus n usually the public exponent e is selected next. As e is publicly known it may be small.
- The CRT cannot be applied for the public key as p and q revealed d.
- The numbers 3, 17, $2^{16}+1$ are favourite values since they are small and have low Hamming weight ($\rightarrow$ s&m algorithm). Normally, the secret key $d \equiv e^{-1} \pmod{\varphi(n)}$ is of the same order of magnitude as n.

Warning: The value e=3 may be critical (cf. Remark C.63).

C.39 Digital Signatures

Goal: Alice wants to send Bob a message over the internet.

Security Requirements:

- The message need not be kept secret but
- Bob shall be convinced
  - that the message was generated by Alice (*authenticity*).
  - that the message has not been altered on the transmission channel (*data integrity*).

C.39 (continued)

Alice generates a digital signature

d_A Alice’s secret RSA key,

n_A Alice’s modulus

- Alice generates a digital document T (a word file that formulates a contract, an applet etc.)
- Alice (resp., her computer) computes H(T) where H denotes an appropriate hash function. The hash value H(T) is interpreted as an integer $\in Z_n$ (cf. C.43)
- Alice sends $T \,||\, H(T)^{d\_A} \pmod{n_A}$

C.39 (continued)

Bob validates the digital signature

e_A Alice’s public exponent,

n_A Alice’s modulus

- Bob receives T’ || sig and interprets sig as Alice’s signature of T’
- Bob checks whether $(sig)^{e\_A}$ (mod n_A) = H(T’)
- In case of equality sig is Alice’s signature of T’. Bob is convinced Alice has signed the message and that it has not been altered. (Justification: $(H(T)^{d\_A} \pmod{n_A})^{e\_A} \equiv H(T) \pmod{n_A}$.)

C.40 Comparison with Handwritten Signatures

Compliances with handwritten signatures:

- Only the authentic signer is able to generate a valid signature (requires access to his / to her secret key).
- The signature is ‘connected’ with the signed document by the properties of the hash function (handwritten signatures: by the paper).
- Everyone can validate a digital signature with the public key (e,n).

C.40 (continued)

Important differences to handwritten signatures:

- A digital signature depends on the signer and the signed document.
- A digital signature signs the binary representation of a digital document (e.g. a word file) but not its content.
- An expert can (at least in principle) distinguish a forged handwritten signature from an authentic one. A forged digital signature can either be detected very easily (since at least one bit is false), or the forged signature is identical to the correct one.

C.41 Remark

- The signer does not need to know his secret key d. He merely must have access to d, i.e. be able to use it.
- This is even a desirable security feature, especially for sensitive applications. The secret key d is stored in a PSE (personal security environment), typically on the disk (encrypted with a password) or on a smart card. The user enters his password to decrypt d or to activate the smart card signing application.

C.42 Digital Signatures: Applications

- contracts (preventing forgery)
- software (provides trust that it is no malware)
- authentication of web sites
- electronic money, electronic purses (preventing forgery, providing authenticity)
- Trusted Computing (provides trust, *blind signatures* provide anonymity)
- …

Details: later + Exercises

C.43 Padding

C.39 explained the generation and validation of digital signatures. It was loosely said that the hash value H(T) is interpreted as an integer.

More precisely, we exponentiate the integer

$$(I \,||\, P \,||\, H(T))_2$$

with	I	information bytes
	P	padding bytes (fixed (known), random or pseudorandom)
	$_2$	indicates binary representation

C.43 (continued)

- The information bytes provide information on the used algorithms.
- The padding bytes ‘extend’ the bit representation of the hash value to the bit length of n.

<u>Note:</u>

The choice of an appropriate padding scheme helps to prevent various attacks.

Security properties of padding schemes are beyond the scope of this course.

C.44 Attacks on Individual Signatures

C.37 and C.38 considered recovery attacks on the secret key d, which allow (e.g.) the forgery of arbitrarily many digital signatures.

Weak hash functions enable attacks on single signatures even if an adversary cannot find the private key d.

C.44 (continued)

Missing second pre-image property:

- Assume that Alice has sent Bob the signed message T || $H(T)^{d\_A}$ (mod n_A) and that Bob is able to find a second message T’≠ T with H(T’)=H(T), which is more favourable for him (e.g.,

  T’ ≅ “I buy Bob’s car for 10000 €. Alice.” instead of

  T ≅ “I buy Bob’s car for 1000 €. Alice.”)
- Then $H(T)^{d\_A}$ (mod n_A) is also a valid signature for T’ in place of T. If Bob replaces T by T’ everyone will believe that Alice had signed this contract.
- Depending on the legal framework (cf. C.57) the contract may be legally binding for Alice.

C.44 (continued)

Missing collision resistance:

- Assume that Bob is able to find any two messages T'$\neq$ T with H(T')=H(T) where T' is more favourable for him (e.g.,

  T $\cong$ "I buy Bob's car for 1000 €. Alice"

  T' $\cong$ "I donate Bob 1000 €. Alice")
- As Bob is a nice guy he prepares the contract T and sends T to Alice. Alice reads the contract, signs it and mails the signed contract to Bob.
- However, if Bob later replaces T by T' everyone will believe that Alice had signed the contract T'.

C.45 RSA: Multiplicity Property

- Note that $y_1^d \, y_2^d \equiv (y_1 \, y_2)^d$ (mod n).
- That is, from signatures / RSA decryption values of y_1 and y_2 one immediately gets the signature / decryption value of their modular product $y_1 y_2$ (mod n).
- The use of hash functions and also of an appropriate padding scheme prevents / counteracts the aimed construction of such messages.

Details: Blackboard + Exercises

C.46 Wherefrom does Bob know Alice's public key (n_A,e_A)?

Proposals:

a) Alice hands Bob a CD with her public key.

Assessment: This solution is surely appropriate for specific scenarios but unacceptable for open networks, for instance, since Alice and Bob may not even know each other.

b) Alice transmits T || $H(T)^{d\_A}$ (mod n_A)|| (n_A,e_A)

Assessment: This solution is absolutely insecure! An active adversary could easily replace the above message by T' || $H(T')^{d\_E}$ (mod n_E)|| (n_E,e_E) where (n_E,e_E) is arbitrarily selected. Bob validates the signature with (n_E,e_E) since he erroneously believes that this was Alice's public key.

c) Certificates

Assessment: Appropriate solution (see C.47 ff.)

C.47 What is a Certificate?

- A certificate contains a *data part* and a *signature part* which contains the issuer’s signature of the data part.

- The data part typically contains
  - certificate owner’s name (alias, ID or similar)
  - public key
  - algorithm IDs (asymmetric algorithm, hash function)
  - permitted use (signing, encryption (key exchange))
  - validity (not before, not after)
  - certificate issuer
  - …

C.47 (continued)

Remark: The most important standard for certificates is X.509

C.48 How to use a Certificates

- Alice sends T || $H(T)^{d\_A}$ (mod n_A)|| $Cert_A$

- Bob first checks the signature of the certificate $Cert_A$ with the public key of the certificate issuer
- If this signature is valid he uses the public key (n_A,e_A) from $Cert_A$ to validate the signature $H(T)^{d\_A}$ (mod n_A) as explained in C.39.

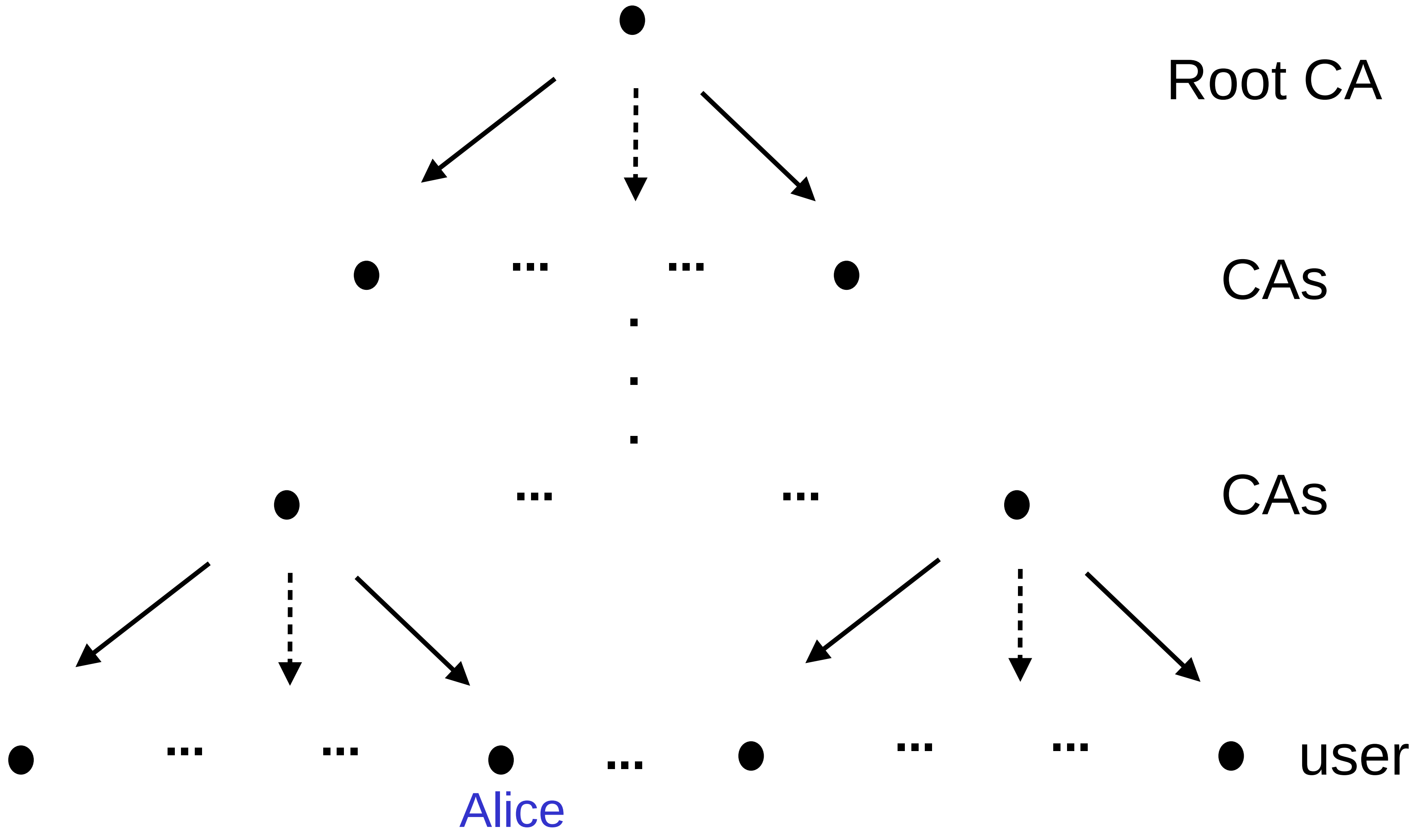
C.49 Wherefrom does Bob know the authentic public key of the certificate issuer?

- Usually there are clearly less certificate issuers than certificate owners. This means that Bob needs to know considerably less public keys than users.
- The public key of the certificate issuer itself may be contained in a certificate, i.e. Alice may transmit a chain of certificates.

C.50 Public Key Infrastructure (PKI)

- Roughly speaking a *PKI* (*public key infrastructure*) is a system that allows the issuing, control and validation of (public key) certificates. In particular, it allows the binding of a public key to a user. The notion of a PKI also comprises organisational measures and technical components.
- The next slide shows a hierarchic structure with a root. Certificates are issued by dedicated *certification authorities* (*CA*s).

C.51 Hierarchic PKI



C.52 Remark

- In a hierarchic PKI the receiver of a message only needs to know the authentic public key of the root. The signer sends a chain of certificates, the first one being the root certificate while the last one contains the signer’s public key.
- The root key might be published in a newspaper or a journal, for instance. Bob stores this public key in his web browser.
- 2 layers (root, user) and 3 layers (root, CAs, user) are typical.

C.52 (continued)

- It was clearly desirable if all users belonged to the same PKI. Unfortunately, in ‘real life’ this is not the case. Instead, there exist many “parallel” PKIs.
- In principal, so-called *cross certificates* enable the secure use of certificates from other PKIs.
- Validated certificates may be stored to improve later signature validations.

C.53 Certificates in e-commerce Applications

- In typical e-commerce applications the user enters sensitive information (password, credit card number) to websites.
- The website (resp., its owner) usually authenticates itself with a certificate.
- If neither this certificate nor a corresponding root certificate are not contained in the certificate store of the web browser the user is asked whether he wants to stop the process or continue (i.e., whether he accepts the certificate which may be then added to the store).
- What should the user do?

C.54 Tasks of the Certification Issuer (main aspects)

- The certification issuer (a CA, for instance) should check the identity of the applicant of the certificate carefully.
- Verisign Inc., for instance, issues different types of certificates. To obtain a so-called Class 1 certificate the applicant only has to transmit a valid e-mail address. Hence the trust in Verisign Class 1 certificates is low .
- In home banking applications banks usually have Class 3 Verisign certificates. As there are careful identity checks the trust in Class 3 certificates is high.

C.54 (continued)

- The certificate issuer should keep the private key confofential that is used to sign certificates to prevent the forgery of certificates.
- If the certificate issuer generates the key pair for the certificate owner (which is not unusual for CAs) the certificate issuer should be very trustworthy. If he is a crook he might use duplicates of the private keys in the name of the certificate owners.

C.54 (continued)

- A CA usually publish a list of valid certificates and a certificate revocation list on his server.
- To be on the safe side the receiver of a signed message can check whether the signer’s certificate has been revoked before he accepts the signature.

C.55 Self-signed Certificates

- A certificate is said to be *self-signed* if the certification issuer coincides with the certificate owner.
- The trust in self-signed certificates depends essentially on the trustworthiness of the certificate owner (= issuer)

Note: In a hierarchic PKI only the root certificate is self-signed.

C.56 Web of Trust

- The PGP and GnuPG community have no hierarchic PKI but use a web of trust.
- User Alice generates his own key pair (secret key, public key). She generates her own certificate, which is uploaded on a public key server.
- Other users who trust Alice and are convinced that the public key is authentic (e.g. because Alice has transmitted its hash value over the phone) sign this certificate.

C.56 (continued)

Basic idea: Assume that some users have confirmed Alice’s certificate. If Bob trusts at least some of them he also trusts Alice and her certificate.

Security Risk: Certifying users might endorse certificates with too little care.

C.53 (continued) Certificates in e-commerce Applications

- Should the user accept a certificate if its issuer, resp. its authentic public key?
- If sensitive information shall be entered the user should be very careful!
- In particular, for home banking applications he / she should stop the process!
- In other cases the user should at least
  - check details of the certificate (issuer, algorithms, type of certificate etc.)
  - try to check whether he is on the authentic website.

C.57 Legal Framework

- In Germany the Digital Signature Act defines different (security) classes of digital signatures. The highest class, the so-called qualified electronic signatures, demands very restrictive conditions on the cryptographic algorithms, the signing device and on the CA.
- If permitted (for a particular application) qualified electronic signatures are legally binding as handwritten signatures.
- However, in e-commerce applications today only a very small fraction of digital signatures meet these strict requirements.
- For non-qualified electronic signatures the consequences are not defined by the law.

C.58 Key exchange

- When using symmetric ciphers (Chapter B) all involved parties have to share secret key(s).
- How can a secure exchange of symmetric keys be realized? This questions reminds on the secure exchange of public keys.
- Clearly, also symmetric keys may be exchanged on CDs. As for public keys this solution is limited to specific situations (cf. C.46).

C.58 (continued)

- Note that symmetric keys additionally have to be kept secret. This excludes the use of certificates that contain these keys. Moreover, in many application symmetric keys are changed regularly (maybe even for each message).
- RSA also supports secure key exchange.

C.59 Elementary Key Exchange Protocol

Goal: Alice wants to transmit a symmetric key to Bob

Alice knows

(n_B, e_B) Bob's public RSA key

Alice transmits k^{e_B} (mod n_B) where k is interpreted as an integer (cf. C.39 and C.43 (padding)).

Bob computes $(k^{e\_B} \pmod{n_B})^{d\_B} \equiv k \pmod{n_B}$.

Note: Only Bob has access to his secret RSA key d_B.

C.59 (continued)

Remark: To illustrate the interaction of the different types of keys in C.60 and C.61 private keys, public exponents and symmetric keys are coloured red, green and blue, respectively.

C.60 Hybrid Protocol (simplified protocol fragment)

Goal: Alice wants to transmit a message T to Bob

Security requirements:

Secrecy, data integrity, authenticity

Situation: Alice and Bob do not share a secret key

Alice knows:

(n_B, e_B)	Bob's public RSA key (Bob's certificate ensures its authenticity)

C.60 (continued)

Alice:

- generates randomly an AES session key k_{RND}(valid only for one session)
- computes $C:=AES(T \,||\, H(T)^{d\_A} \pmod{n_A} \,||\, Cert_A, k_{RND})$
- transmits $C \,||\, k_{RND}^{e\_B} \pmod{n_B}$

C.60 (continued)

Bob:

- computes $(k_{RND}^{e\_B} \pmod{n_B}))^{d\_B} \equiv k_{RND} \pmod{n_B}$.
- decrypts C with the session key k_{RND}
- validates Alice’s certificate $Cert_A$
- validates Alice’s signature $H(T)^{d\_A} \pmod{n_A}$ with Alice’s public key (n_A,e_A)

C.61 Remark

- Only Bob has access to his secret key d_B.
- It is strongly recommended to use different RSA keys for signing and key exchange (different keys for different purposes!).
- Key exchange / key agreement mechanisms (cf. Section C.c) are part of many security protocols as SSL, TSL, PGP, for instance.

C.62 Padding

- As for signature applications padding is also applied to key exchange mechanisms, i.e.

  $y:=(I \| P \| k_{RND})_2$

  is exponentiated (see also C.43).

C.63 Remark

- Assume that fixed (known) padding bytes are used. Then
  $y^{e\_B} \equiv (D+k_{RND\_2})^{e\_B} \equiv c \pmod{n_B}$
  with known constant D and c (= transmitted value).
- Finding k_{RND} is then equivalent to finding a small zero of the modular polynomial equation $p(x) := (D+x)^{e\_B} - c \equiv 0 \pmod{n_B}$
- Note that for a large modulus n_B it is usually very difficult to solve non-linear equations if the factorization of n_B is unknown.
- However, with lattice-based methods it is feasible to find all zeroes that are $< n_B^{1/e\_B}$.

C.63 (continued)

Example:

- n_B = 1024 RSA modulus
- $e = 3$

Up to 340 least significant bits can be recovered (e.g. 2 AES keys).

Note: For $e = 17$ a $1024 / 17 \approx 60$ bit key can be recovered with this method. Hence $e = 2^{16}+1$ is a widely used value for key exchange mechanisms.

Note: This attack does not find the secret RSA key d_B but only one encrypted value (here: the session key k_{RND})

C.63 (continued)

Remark:

- This attack is irrelevant for signing applications.
- However, in 2006 Bleichenbacher demonstrated that e=3 can also be dangerous for signing applications in case of careless signature validation, i.e. if the padding format is not checked. The public exponent e =2^{16}+1 also prevents this attack.

C.64 Key Exchange / Key Agreement

- C.59 ff. discuss key exchange mechanism. The (symmetric) key k_{RND} is selected by one party (the sender).
- In key agreement protocols both parties influence the value of the key (see Section C.c), i.e. they “agree” upon a key.

C.65 HMAC

- Let H denote a hash function (cf. C.30). The HMAC of a bitstring m is given by

  HMAC(m,k):=H(k $\oplus$ opad || H(k $\oplus$ ipad || m))

  where opad and ipad are constants (specified in RFC 2104).
- For long messages m on computers the HMAC runs much faster than all the MAC constructions from Section B.c (cf. C.30).

Note: SSL V3.0 (see C.65) uses a similar keyed hash function: MAC(m,k):=H (k || p_1 || H(k || p_2 || m)).

Here p_1 and p_2 denote constants.

C.66 SSL (Secure Socket Layer)

- SSL is a communication protocol that is widely used for home banking and e-commerce.
- Currently, SSL V3.0 and its successor TLS (Transport Layer Protocol) are used. TSL V1.0 is able to communicate with SSL V3.0.
- Security Goal: Secure exchange of messages, ensuring secrecy, authenticity, data integrity.

C.66 (continued)

- SSL is a two-party protocol. The party that starts the protocol is assigned the role of the *client,* the other the role of the *server*.
- SSL falls into two phases
  - Handshake protocol
  - Encryption of application data

C.67 The Handshake Protocol

Within the handshake protocol both parties

- agree upon a cipher suite (= set of cryptographic algorithms with parameters), see C.69
- exchange resp., agree on four keys and two IVs:
    - w IV_C, IV_S
    - w Session key_C, Session key_S
    - w MAC key_C, MAC key_S

  where the indices C and S stand for “client” and “server”, respectively.
- The client uses IV_C, Session key_C, MAC key_C, to encrypt messages, the server IV_S, Session key_S, MAC key_S.

Note: Keys and IVs may be reused in later sessions with the same client-server pair to save negotiation time.

C.68 Mutual Authentication

Within the handshake protocol client and server should authenticate themselves. The following combinations are typical:

- Client and server have a trustworthy certficate (ideal case, rare)
- The client authenticates himself / herself with a PIN (and possibly with a TAN), the server has a (trustworthy) certificate (typical for home banking)
- The client does not authenticate, the server has a certificate (normal for e-commerce applications).

C.69 Encryption of Application Data

- The application data are encrypted with a symmetric cipher.
- In SSL V 3.0 authenticity and data integrity are ensured by a keyed hash function ($\rightarrow$ Note in C.63). TLS uses the HMAC.

C.70 Cipher Suites

SSL requires three types of cryptographic primitives:

- asymmetric algorithms (purpose: authentication of certificates, key exchange / key agreement)
- Symmetric ciphers (purpose: encryption of application data)
- Hash function (purpose: ensure authenticity and data integrity of the application data)

C.70 (continued)

Example (cipher suite):

SSL_RSA_WITH_3DES_EDE_CBC_SHA1

This means

- RSA: used for key exchange
- 3DES_EDE_CBC: Triple-DES ($DES \circ DES^{-1} \circ DES$, cf. B.88) in CBC mode
- SHA1: hash algorithm

C.70 (continued)

Note:

(i) Client and server usually support many cipher suites.

(ii) Within the handshake protocol client and server agree upon the strongest cipher suite that is supported by both parties.

(iii) ‘no encryption’ is possible.

(iv) Any party may abort the handshake protocol if the other party offers only too weak cipher suites.

C.71 PGP (Pretty Good Privacy)

- widespread computer program to encrypt data; the first version was published by Phil Zimmermann in 1991.
- Security goals: secrecy, authenticity, data integrity; uses hybrid mechanisms
- No hierarchic PKI ($\rightarrow$ web of trust, C.56)

C.72 GeldKarte: Internet Usage

- The GeldKarte is a German electronic purse system (cf. Chap. A, Sect. B.b).
- Usually, the customer inserts his GeldKarte into the merchant’s terminal. The GeldKarte chip communicates with the terminal, or more precisely, with the merchant’s smart card.
- To transfer payment records symmetric algorithms are applied (cf. B.29ff., B.58).

C.72 (continued)

- The GeldKarte can also be used for payments over the internet.
- Principal security risk: The cardholder does not enter the merchant’s shop physically. With a faked website a crook could pretend to be another (reliable) merchant.
- If the card reader is integrated in the PC different price might be displayed on the screen than finally requested from the card.

C.72 (continued)

- Solution: The GeldKarte is inserted into an external smart card reader (more precisely: into a Class 3 card reader with own display and own keyboard).
- The cryptographic protocol runs between the GeldKarte chip and the merchant’s smart card.
- The relevant information (price, merchant’s name) is displayed on the card reader.
- The cardholder thus can detect if the price or the merchant’s name on the PC screen are not correct. He must explicitly agree to the payment.
- The merchant’s card authenticates itself with a certificate.