**Introduction Encryption & History (8) (mandatory)**

**Design and attacks for Enigma (probably not)**

Q 1: What are the diﬀerences between cryptography and cryptanalysis?

A:

**Cryptology** ... is the wisdom of ciphers

1 **Cryptography** ... describes how a secret code works, e.g. encryption and decryption algorithm

2 **Cryptanalysis** ... deﬁnes how to decrypt or analyze a given cipher, e.g. without knowing the key

Q 2: How to decrypt a ciphertext encrypted by a monoalphabetic substitution using a frequency analysis?

A:

**Procedure**

1 Random pairing of the letters of the alphabet

2 Replace the opposing letters in your message

**Procedure**

1 Choose a keyword (e.g. Julius Caesar)

2 Delete all repeating letters

3 Use the keyword only for the beginning and continue with the alphabet after the last letter

4 Be careful not to use repetitive letters in the encoding alphabet

**Coding Rule**

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

J U L I S C A E R T V W X Y Z B D F G H K M N O P Q

**Attacks**

At that time it was not possible to try out all the variants for (2.) and (3.)

But frequency analysis could help

**Cryptanalysis**

Science of decryption without knowing the key

Technique was ﬁrst described in the 9th century by an Arab philosopher

**Procedure**

1 Determine the occurrence frequency for each letter of the plaintext alphabet

2 Determine the occurrence frequency for each letter of the intercepted ciphertext

3 Decode the ciphertext by comparing the two frequency analyses

Q 3: Do you know an example for a polyalphabetic substitution?

Goal: Development of a stronger system than monoalphabetic encryption (end of the 16th century)

* Preliminary work by Leon Battista Alberti (15th century)

Main idea: Use of several cipher alphabets

Advantage: Representation of identical letters of the plaintext by diﬀerent letters of the ciphertext

Ü Covering the frequency of characters in plaintext

* Vigen´ere Cipher was named after a French diplomat
  + Ü Blaise de Vigen´ere (∗1523)
* Historically, the development of polyalphabetic substitution was an important milestone

Q 4: How to decrypt messages encrypted by a Vigen´ere cipher

**Remarks**

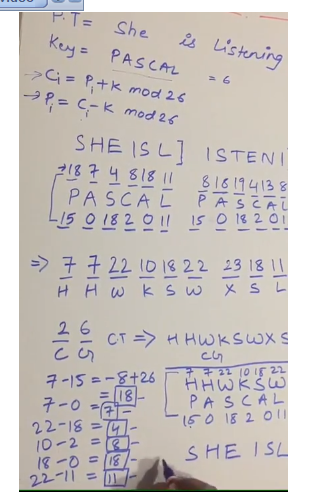
* Vigen´ere square allows the encryption with 26 diﬀerent ciphertext alphabets
* A code word must be negotiated

**Procedure**

1 Write the code word several times over the plaintext

2 Find the row of the table where the ﬁrst entry matches the letter of the code word

3 Determine the encoding letter for the corresponding plaintext letter using this row



Q 5: What are the diﬀerences between Vigen´ere cipher and Homophonic substitution?

**Homophonic Substitution**

* Since only a ﬁxed ciphertext alphabet is used, it is a monoalphabetic substitution
* A plain text letter is represented by several ciphertext letters (usually numbers), but not the other way round

**Vigen´ere Cipher**

* Polyalphabetic substitution
* A ciphertext letter can also represent several plaintext letters

**Conclusions**

* Vigenere cipher better than Homophone substitution method
* From theoretical point of view, both procedures are still insecure!

[ Note:

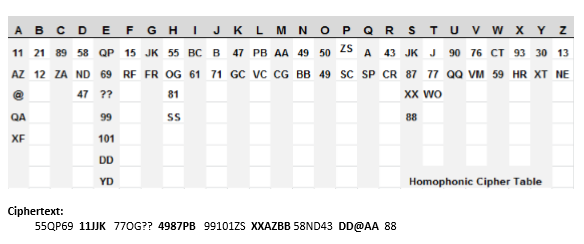
Homophonic Cipher

In this cipher technique, , plaintext letters map to more than one cipher text symbol. Usually, the highest frequency plaintext symbols are given more equivalents than lower frequency letters. In this way, the frequency distribution is flattened, making analysis more difficult.

 Since more than 26 characters will be required in the cipher text alphabet, various solutions are employed to invent larger alphabets such as a numeric substitution, uppercase, lowercase, upside down, and fanciful symbols etc.

 A Mantua Homophonic Cipher (15th century, Roman Empire) is an example to this type of cipher. Example: Plaintext: HE EATS THEN SLEEPS AND DREAMS Ciphertext: ??

Key: The following table is the key.



Q 6: How works the encryption using a Vernam cipher? Why is this cipher information-theoretically secure?

A:

Procedure

1 Select a key at random and make sure that it is at least as long as the plaintext

2 Write the key over the plaintext

3 Add (for decrypting subtract) modulo 26, for binary representation use the XOR operation

One Time Pad Cipher

known as Vernam Cipher is implemented using a random set of non repeating characters as the Key. The rest of process is same as the Vigenère cipher.

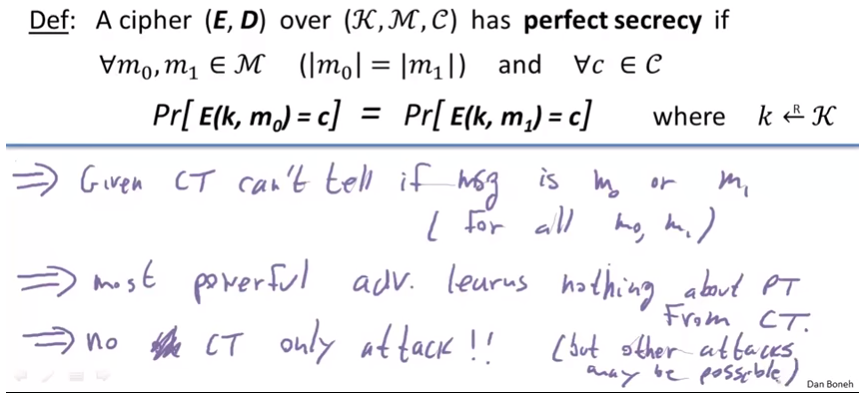
Example1: Plaintext: “ HOW ARE YOU ’’ Key = NCBTZQARX Ciphertext: ?

 Note: If (Pencoding + Kencoding) >= 26 then – 26

|  |  |
| --- | --- |
| F:\First Semester\Software Security_6 Credit\Exam Preparation\One time pad En.PNG | F:\First Semester\Software Security_6 Credit\Exam Preparation\One time pad deco.PNG |

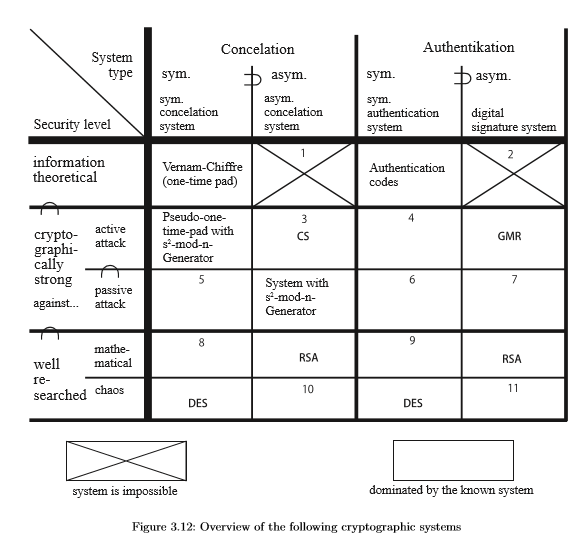
**General Remarks**

* Information-theoretically secure : The attacker is able to infer information about the plaintext, but not enough to determine it exactly
* Security based on randomness of the key
* Developed in 1917 by Major Joseph Mauborgne.
* At the American Army Cryptographic Research Department



Q 6.a: How to categorize modern encryption systems using Pﬁtzmann’s table? What means that a crypto system is cryptographically strong?

A:



**Cryptographically** secure pseudorandom number generator. A **cryptographically** secure pseudo-random number generator (CSPRNG) or **cryptographic** pseudo-random number generator (CPRNG) is a pseudo-random number generator (PRNG) with properties that make it suitable for use in **cryptography**.

The symmetric and the asymmetric encryption systems are based on the same cryptographically strong PRNG. Cryptographically strong PRNG are of interest in other contexts too, because the generation of genuine random numbers is in practice a considerable problem.

Therefore eﬃcient asymmetric encryption systems that are cryptographically strong relative to a pure standard assumption, are still not known. **RSA may not be proven to be cryptographically strong (as secure as factorizing is diﬃcult) but for its use as an indeterministic asymmetric encryption system as in** §3.6.4.1, no successful attacks are known - and this for years.

If asymmetric concealation is required and if you cannot rule out active attacks, you should use RSA (or CS). If you can rule out active attacks then the s2− mod n generator is preferable: **On the one hand it is cryptographically strong (proven), so if you can break it you can factorize and therefore break RSA too.** On the other hand, it is proven for the s2− mod n generator that a passive attacker does not gain any information about the plaintext. Such a proof does not exist for RSA either.

Q 7: How was the Enigma designed and what was the key for this electronic cipher machine? How was it possible to decrypt ciphers encrypted by the Enigma?

**Encryption using s2-mod-n (9) (probably)**

Q 1: What means prime factorization and why is this operation so important for asymmetric encryption systems?

A:

**Deﬁnition**

The prime factorization of a natural number n is the product

n = p1e1·p2e2 · ... ·pkek

where p1,...,pk are diﬀerent prime numbers in pairs and the exponents are positive natural numbers, i.e. e1,...,ek ∈N+

Algorithms Pollard’s rho algorithm Quadratic sieve algorithm Number ﬁeld sieve

Ü No polynomial algorithm for prime factor decomposition has been found yet!

**Factorization is hard**

There is no polynomial algorithm to eﬃciently calculate the prime numbers p and q from a given n, so that p·q = n applies

Q 2: There are two other operations that are based on factorization. Why these operations are important too?

A:

**Implications**

There are two other algorithms that are as hard as factorization

1 Calculating a square root mod n

2 Testing for a square mod n(\*1)

Ü However, if you know p and q, then both tasks can be solved eﬃciently, e.g. root extraction using the CRA (Chinese Remainder Algorithm)!

(\*1) Also called the quadratic residuacity problem

Q 3: What cryptographic assumption is the basis for s2-mod-n?

Q 4: Why is s2-mod-n also called Pseudo One-Time-Pad?

A:

**Encryption and decryption**

Ü Add (or subtract) the pseudo-random bit sequence to the plaintext or ciphertext using the XOR-Operation simular to the One-Time Pad

Q 5: What are the diﬀerences between the symmetric and the asymmetric variant of s2-mod-n?

**A:**

**Problem**

Assuming n people want to communicate in pairs, so you need k diﬀerent keys with k = n·(n−1)/2

**Examples**

- for n = 100 we obtain k = 4.950 keys

- for n = 1000 we obtain k = 499.500 keys

Ü Quadratic increase

**Solution**

- Asymmetric encryption requires only k = 2·n diﬀernet keys

- Keys for symmetric encryption can afterwards be exchanged using asymmetric encryption

Q 6: Which part of the key is public, which part is secret?

A:

**Components Secret Key** - Two prime numbers p and q with p ≡ q ≡ 3 mod 4

**Components Public Key** - Product n with n = p·q

**Encryption**

1 Create a bit sequence with s2-mod-n-method for a randomly selected start value s

2 Add the bit sequence to the plain text to calculate the ciphertext, then send the ciphertext and a last sk+1 that was not used for the encryption

**Decryption**

1 Determine the bit sequence using p und q by successively extracting the square roots(\*2) from sk+1

2 Calculate the plaintext using XOR operation

(\*2)Only use roots that are themselves squares again!

Q 7: How is it possible to compute a square root eﬃciently? Illustrate the procedure using an example.

A:

**Computing a square root is hard**

There is no polynomial algorithm to calculate a square root for mod n if you do not know the prime factors p und q

Procedure if you know the prime factors 1 Calculate the square roots for mod p and mod q using the following formulas

- yp = y(p+1)/4 mod p

- yq = y(q+1)/4 mod q

Ü Note: Formulas are only valid for p ≡ q ≡ 3 mod 4

2 Calculate the square root for mod n from yp and yq using the Chinese Remainder Algorithm (CRA)

The following prime numbers are given - p = 3 and q = 7 with n = p·q = 21 - We assume further on that p ≡ q ≡ 3 mod 4 is fulﬁlled

Assumption: The following root is to be calculated

- y = √4 ≡ ?

Task: How can a square root be calculated eﬃciently?

- y = √4 ≡ 2 ≡ 5 ≡ 16 ≡ 19 mod 21

**Formulas**

- yp = y(p+1)/4 mod p

- yq = y(q+1)/4 mod q

Computing the square roots

- y3 = 4(3+1)/4 = 41 ≡ 1 mod 3

- y7 = 4(7+1)/4 = 42 ≡ 2 mod 7

Ü Now we have two intermediate results y3 and y7

Note - The calculation rule can only be used under the condition p ≡ q ≡ 3 mod 4!

Chinese Remainder Algorithm (CRA) CRA(yp,yq,p,q) = u·p·yq + v ·q·yp mod n

Instantiation CRA(1,2,3,7) =u·3·2+v ·7·1 mod 21,

How to calculate the base vectors u and v?

- The integer variables u and v must fulﬁll the condition

gcd(3,7) =u·3+v·7 = 1

- Values for u and v can be calculated using the Extended Euclidean algorithm

Extended Euclidean algorithm

7 = 2·3 + 1 (q = s1 ·p + r1)

3 = 3·1 + 0 (p = s2 ·r1 + r2)

In reverse order, i.e. solve all equations to the rest and then insert them step by step

0 = 3−3·1 (skip this equation, because r2 = 0)

1 = 1·7−2·3 (r1 = q−s1 ·p)

Ü The base vectors are u = −2 and v = 1

Ü Results in the square root CRA(1,2,3,7) = 16

Ü Note: In addition, check whether 16 is a square again

# Reference: Square Roots Mod N Part 2

# https://www.youtube.com/watch?v=dSIkv9U5WO8

Q 8: Why is s2-mod-n cryptographically strong? Note you do not need to provide a formal proof for this property, however you should be able to explain the rough idea behind this proof.

A:

**Question**

Why is the generator s2-mod-n (symmetric variant) an unpredictable (cryptographically stronger) Pseudo-Random Bit Generator (PRBG)?

**Proof obligation**

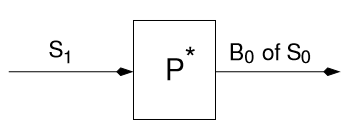
Under the factorizing assumption (resp. in our proof under the quadratic-residuosity-assumption), there is no polynomial algorithm that can distinguish the random sequence generated by the PRBG from a real random sequence

**Assumption**

There is a polynomial algorithm P which predicts the left continuation bit of a given k-bit sequence with a probability greater than 1/2 (assuming that n of the residue class is also known)

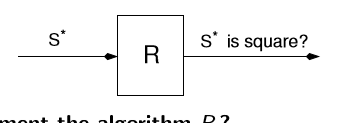
**Proof (First Step)**

Then an algorithm P∗ can be constructed from P, which calculates Bit B0 of the initial value S0 for a given value S1



**Proof (Next Step)**

Then an algorithm R can be constructed from P∗, which checks for a given value S∗ with Jacoby symbol +1, whether S∗ is a square, i.e. whether the condition S∗ ∈ QRn holds



**Conclusion** The algorithm R is able to perform a square test in polynomial time without knowing p and q for any S∗ with Jacobi symbol +1, i.e. R is able to check, whether S∗ ∈ QRn

**Proof (Last Step)** The derived statement is obviously in contradiction to the quadratic residuosity assumption, which is strongly related to the factorization assumption. This means that an attacker who can eﬃciently predict random numbers would also be able to factorize in polynomial time.

GMR - Cryptographic Signature System (10) **(probably not)**

How does GMR diﬀer from other signature systems you know? What means collision resistant for two given permutations? How to generate a signature using GMR? Give an example. Why is it necessary to restrict the deﬁnition range for the square functions of GMR? How is this restriction implemented? How to use the Chinese Remainder Algorithm (CRA) to generate GMR signatures? Which part of the key is public, which part is secret? How is it possible to attack the signature system GMR and how can this attack be prevented? Why is GMR cryptographically strong? Note you do not need to provide a formal proof for this property, however you should be able to explain the rough idea behind this proof.

**RSA - Encryption & Signatures (11) (probably)**

Q 1: How does RSA diﬀer from other encryption systems you know? Why is RSA only classiﬁed as well researched?

A:

* Can be used either as an asymmetric encryption system or digital signature system
* RSA is based on the factorization assumption
* Under the assumption of factorization, however, the correctness of RSA is not yet formally proven

Ü Hence RSA is not cryptographically strong, only “well researched”!

**Basis**: Modular exponentiation of messages in the residue class ring

Q 2: Which part of the key is public, which part is secret?

Q 3: How to generate a suitable RSA key pair?

A:

1 Select security parameter l

2 Select prime numbers p and q with |p|≈|q|= l and p 6= q

3 Calculate the product n = p·q

4 Select c with 3 ≤ c < ϕ(n) and gcd(c,ϕ(n)) = 1

Ü Note: ϕ(n) = (p−1)·(q−1)

5 Calculate d as multiplicative inverse of c with c ·d ≡ 1 mod ϕ(n) using the extended Euclidean algorithm

**Secret parameters for key generation**

- p, q und ϕ(n)

**Secret key**

- d, (p and q)

**Public key**

- c and n

Q 4: Why is the naive version of RSA not secure against attacks based on the multiplicative property? How can this attack be prevented?

A:

Note that this is only the naive version of RSA, which means that this setup is not secure against attacks based on the multiplicative property of RSA!

**Assumptions**

1 the public key (t,n) for testing signatures,

2 the messages m1 and m2, and ﬁnally

3 the signatures ms1 and ms2 are known to the attacker

**Passive Attack**

Calculate m3 := m1 ·m2 and

Obtaining the corresponding signature by applying the following calculation rule

ms3 := ms1 ·ms2 = (m1 ·m2)s mod n

Ü This attack is a selective break, where the victim must be willing to sign two messages for the attacker

Active Attack

* Goal The attacker is interested in getting any message signed by the victim

**Procedure**

1 Select the message to be signed arbitrarily, e.g. m3

2 Select a number r randomly with 1 ≤ r < n in such a way, that for r a multiplicative inverse r−1 exists

3 Calculate m2 := m3 ·rt mod n

4 Send message m2 to the victim for signing

5 Calculate ms3 := ms2·r−1 ≡ (m3·rt)s ·r−1 ≡ ms3·r ·r−1 mod n

Ü This is a selective break, where the victim must be willing to sign one message for the attacker

**prevent attacks** based on the multiplicative property:

Ü Both attacks, the passive attack and the active attack of Judy Moore, use the multiplicative structure of RSA

**Countermeasures**

Ü Collision-resistant hash function are used to neutralize the multiplicative structure

Ü For a digital signature system create the signature only from the hashes, not from the plaintext, because for hash function h holds

h(m1)s ·h(m2)s ≠ (h(m1)·h(m2))s

Ü For a concelation system, attach the hash of the message to the plaintext and then encrypt the entire text block

Ü After decryption you need to perform additionally a redundancy check using the received hash value

Q 5: What are the technical challenges involved in implementing an RSA cryptosystem? Which algorithms must be provided?

A:

1 Converting the plain text into a digital representation

2 Calculating large prime numbers eﬃciently

Ü e.g. using the Miller-Rabin primality test

3 Calculating the Multiplicative inverse

Ü e.g using the Extended Euclidean algorithm

4 Eﬃciently exponentiate large numbers

Ü e.g. by repeated squaring and multiplication

5 Strategies to prevent attacks, e.g. neutralizing the multiplicative structure of RSA

Q 6: Why does the double exponentiation of a plaintext with secret and public keys result in the plaintext again? Note you do not need to provide a complete proof of correctness, however you should be able to explain the rough idea behind this proof.

**Proof**

obligation ∀m : Zn •(mc)d = (md)c = mc·d ≡ m mod n Proof

according to the assumption applies

c ·d ≡ 1 mod ϕ(n)

with

ϕ(n) = (p−1)·(q−1) and

a ≡ b mod (c ·d) ⇒ a ≡ b mod c

we can deduce

c ·d ≡ 1 mod (p−1)

⇔ ∃k : Z•c ·d = k ·(p−1) + 1 i.e. the following condition holds

mc·d ≡ mk·(p−1)+1 ≡ m·(mp−1)k mod p

according to Fermat’s little theorem we know if

gcd(m,p) = 1, than mp−1 ≡ 1 mod p

if m is not a multiple of p, we deduce

m·(mp−1)k ≡ m·1k ≡ m mod p

if m is a multiple of p, we deduce m ≡ 0 mod p and m·(mp−1)k ≡ m ≡ 0 mod p

Since p is a prime number, there can be no other cases, i.e. it applies

mc·d ≡ m mod p

The proof is identical for the prime number q

mc·d ≡ m mod q

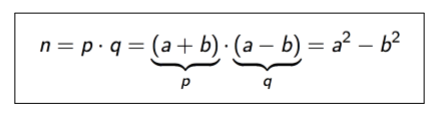
Using the Chinese Remainder Algorithm follows for n = p·q

mc·d ≡ m mod n

Q 7: How works a total break of RSA by Fermat’s factorization method? What is a suitable countermeasure?

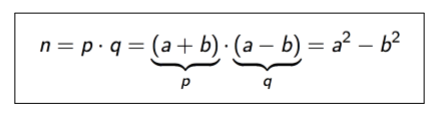
Ü Fermat’s Factorization Method

Algorithm for the prime factorization of a natural number Method is only eﬃcient when p and q diﬀer only a little from √n



* Idea: Search for numbers that fulﬁll the equation
* Start the search at a = b√n + 1c [b floor c]
* Increase a stepwise by 1, until (a2 −n) is a square

Ü Let n = 143; We are looking for the prime factors p and q



* Select a with a = b√n + 1c = b√143 + 1c = 12
* Find a suitable b, that fulﬁlls the equation n = a2 + b2 for a
* b2 = a2 −n = 122 −143 = 1
  + Ü 1 is a square!
* If a = 12 and b = 1 than we are able to calculate
* Ü p = a + b = 12 + 1 = 13
* Ü q = a−b = 12−1 = 11

prevent the attack of Fermat:

Ü Note that the method is only eﬃcient when p and q diﬀer only a little from √n

**Countermeasures**

Ü For the key generation we have to select a module n, where n cannot be factorized with two prime numbers of approximately the same size

Ü The conditions |p|≈|q|= l and p≠q address this problem, i.e. the lengths of p and q must not be identical

DES/AES - Symmetric Encryption (12) (probably)

What is the meaning of confusion and diﬀusion and how these concepts are usually implemented? What are the advantages of a Feistel network and which encryption system uses this technology? Why is DES considered insecure today and should not be used anymore? How works a brute-force attack? What is an important assumption of this attack? Why is the complementarity property of DES useful for an attacker? How much better is Triple-DES compared to DES? How works a Meet-in-the-Middle attack? Which four operations need to be implemented for AES? Which of these operations are based on polynomial arithmetic? How to implement polynomial arithmetic eﬃciently for AES? How many rounds are required for the AES algorithm?

Operation Modes: Block Cipher vs. Stream Cipher (13) (probably)

**Operation modes for full disc encryption (probably not)**

What are the problems with the practical use of encryption methods? How can operation modes help to solve these problems? What is the diﬀerence between a synchronous and a selfsynchronizing mode? Why is the Electronic Codebook Mode (ECB) considered insecure and should not be used? Why could be Cipher Block Chaining (CBC) a better alternative? What are the disadvantages of this operation mode? What is the problem with error propagation? Which operation mode is not sensitive to this problem? How can random access implemented? Which operation modes are suitable for hard disk encryption? Why could be a tweaked-codebook mode (e.g. XTS) a good option? How many keys are required for this mode?