Cryptography and Network Security, PART IV: Reviews, Patches, and Theory

Timo Karvi

11.2012

Key Lengths I

- The old block cipher standard DES has the key length of 56 (actually 64, but 8 bits are not used). Assuming that DES is an ideal cipher (i.e 2^{56} random invertible functions $\pi_i : \{0,1\}^{64} \longrightarrow \{0,1\}^{64}$), then for all plain texts m and cipher texts c there is at most one key such that $c = DES(k, m)$ with probability $\geq 1 - 1/256 \approx 99.5\%$.

  Proof.

$$Pr\left[\exists k' \neq k : \quad c = DES(k, m) = DES(k', m)\right]$$

$$\leq \sum_{k' \in \{0,1\}^{56}} Pr[DES(k, m) = DES(k', m)] \leq 2^{56} \cdot \frac{1}{2^{64}} = \frac{1}{2^8}$$

- For two pairs (m_1, c_1), (m_2, c_2) the inicity probability is $\approx 1 - 1/2^{71}$.
- For AES-128, given two input/output pairs, unicity probability is $\approx 1 - 1/2^{128}$.

Key Lengths II

- Thus two input/output pairs are enough for exhaustive key search (try all keys until one key gives the cipher texts for known plain texts).
- In 1997 distributed internet key search succeeded to reveal the key in 3 months. 1998 the EFF machine (special purpose machine) broke DES in 3 days. The cost of the machine was 250 000 dollars. Copacobana Rivyera made the current record breaking DES in less than one day (using 128 Spartan 3 5000 FPGAS chips).

Strengthing short key block ciphers I

- Use three keys: $3E((k_1, k_2, k_3), m) = E(k_1, D(k_2, E(K_3, m)))$.
- Double encryption does not help much, because there is meet in the middle attack. Suppose you have plaintext-ciphertext pair. For every key k_i encrypt $E(k_i, m)$. Save the results. Then for every key decrypt $D(k_i, c)$. If the decryption result is found in the table, we have a key candidate (k_i, k_j). Apply the same method to the second pair (m, c), but this time only using the keys found in the first phase. The key is found practically in the first or second phase.
- The time spent: build and sort the table plus search in table or

$$2^{56} \log(2^{56}) + 2^{56} \log(2^{56}) < 2^{63} << 2^{112}.$$

Key lengths and Attacks I

- Minum key length should currently be over 100. AES minimum is 128.
- It is possible to use quantum computers to break block ciphers faster than ordinary computers do. But if the key size is 256, even quantum computers do not help.
- It is still unclear if quantum computers can be built.

Key lengths and public key systems I

- In RSA, key size 1024 is in danger in the near future. 2010 an 768 bit integer was factored.
- Consider trivial factoring testing all the odd numbers from 3 up to $\sqrt{(n)}$. If n is 1024 bit long, there are 2^{1024} different n and $\frac{1}{2} \cdot 2^{512}$ factor candidates.
- If one test takes one microsecond, in one year it is possible to test 3.15×10^{130} cases. Thus in the worst case the test could take about 10^{141} years.
- The running time of this trivial algorithm is comparable to n, $\mathcal{O}(n)$. However, in number theoretic algorithms we should use the size of n, not n itself, when expressing the time complexity. The size of n is $\log_2(n)$ and $n = 2^{\log_2 n}$.
- Thus $\mathcal{O}(n) = \mathcal{O}(2^{\log_2 n})$. So the algorithm is exponential with respect to the size of n.

Key lengths and public key systems II

- There are, however, faster factoring algorithms: general number field sieve (GNFS), Dixon's algorithm, Lenstra's elliptic curve method, etc. But even these are exponential.

Theory: Good Ciphers I

- A cipher should be such that given a ciphertext, there is no clue what is the corresponding plaintext. This can be formalized as follows.
- Let $\mathcal{M}$ be a message space, $\mathcal{K}$ a key space, and $\mathcal{C}$ a ciphertext space. A cipher (E, D) is a pair of functions $E : \mathcal{K} \times \mathcal{M} \longrightarrow \mathcal{C}$ and $D : \mathcal{K} \times \mathcal{C} \longrightarrow \mathcal{M}$ such that $D(k, E(k, m)) = m$ for all $m \in \mathcal{M}$.
- (E, D) has a perfect secrecy, if $\forall m_0, m_1 \in \mathcal{M}$, $|m_0| = |m_1|$, $\forall c \in \mathcal{C}$

$$Pr[E(k, m_0) = c] = Pr[E(k, m_1) = c],$$

where probability is computed over all possible keys k. It is assumed that k is uniform in $\mathcal{K}$, i.e. every k has the same probability.

Theory: Good Ciphers II

- Consider one time pad (OTP): $c = m \oplus k$, where k is as long as m and the encryption is the bitwise xor of m and k. We have:

Lemma

OTP has perfect secrecy.

Proof.

$$Pr_k[E(k,m) = c] = \frac{\#\text{keys } k \in \mathcal{K} \text{ s.t. } E(k,m) = c}{|\mathcal{K}|} = 1/|\mathcal{K}|,$$

because $k \oplus m = c \Longrightarrow k = m \oplus c$. $\square$

- If the number of keys k such that $E(k,m) = c$ is constant, then cipher has perfect secrecy.
- In order a cipher has perfect secrecy, its key space must be at least as large as its message space.

Theory: Good Ciphers III

Theorem

If a cipher has perfect secrecy, then $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof. Assuming $|\mathcal{K}| < |\mathcal{M}|$, we shall derive a contradiction to perfect secrecy. Consider a message $m_0 \in \mathcal{M}$ and key $k_0 \in \mathcal{K}$. Let $c = E(k_0, m_0)$. Consider the set $S = \{m \in \mathcal{M} \mid \exists k \in \mathcal{K} \text{ s.t. } D(k, c) = m\}$. Since D is deterministic, $|S| \leq |\mathcal{K}| < |\mathcal{M}|$. Therefore there exists $m_1 \in \mathcal{M}$ such that $Pr[k \xleftarrow{} \mathcal{K} : E(k, m_1) = c] = 0$, else c is obtainable from m_1 but doesn't decrypt to it, violating the definition of symmetric encryption schemes. Since $Pr[k \xleftarrow{} \mathcal{K} \mid E(k, m_0) = c] > 0$, we have a contradiction. □

- Typically message space consists of arbitrary long messages.
- The theorem means that in order a cipher to have perfect secrecy, an encryption key must be as long as the message. This is not practical.
- That is why in modern encryption systems we use one time pad encryption only as one part of the whole encryption process.

Random Number Generators I

- We have seen that random number generators have an important place in cryptography. They are used to generate keys and nonces.
- We do not really have real random number generators, but we can program pseudo random number generators (PRG). PRG must be unpredictable.
- We define PRG to be a mapping $G : K \longrightarrow \{0,1\}^n$, where K is a seed space and PRG generates n pseudo random bits.

Random Number Generators II

Definition

PRG G is predictable at position i, if there exists a polynomial time algorithm A and an index i, $0 \leq i \leq n-1$, such that

$$Pr_{k \xleftarrow{} \mathcal{K}}\left[A(G(k))|_{1,\ldots,i} = G(k)|_{i+1}\right] > \frac{1}{2} + \varepsilon$$

for non-negligible ε.
PRG is unpredictable, if it is not predictable.

Suppose $G : K \longrightarrow \{0,1\}^n$ is such that for all k, $xor(G(k)) = 1$, i.e. xoring all the bits of the bit string $G(k)$ gives 1. Then G is predictable, because given the first $n-1$ bits we can predict with probability 1 the n'th bit.

Weak Random Number Generators I

- A typical random number generator is Linear Congruence Generator with parameters a, b, p:

$$r[i] = a \cdot r[i-1] + b \mod p.$$

  First $r[0]$ is a seed value and the method outputs bits of $r[i]$ for every i.

- For example there is a gnu C library function random:

$$r[i] = (r[i-3] + r[i-31])\%2^{32}.$$

- Never use random() for cryptographic purposes! (Kerberos v4 did this!)

Negligible and Non-Negligible I

- The definition of predictable pseudo random generator used the concept of a negligible number.
- In practice, ε is non-negligible, if $\varepsilon \geq 1/2^{30}$. Then an event is likely to happen over 1GB of data.
- ε is negligible, if $\varepsilon \leq 1/2^{80}$. Then an event will not happen over the lifetime of a key.

Negligible and Non-Negligible II

There are also formal definitions: is a function $\varepsilon : \mathbf{N} \longrightarrow \mathbf{R}^+$ and

- ε is negligible, if for every positive integer d there exists an integer λ_d such that for all $\lambda > \lambda_d$

$$\varepsilon(\lambda) \leq 1/\lambda^d.$$

- ε is non-negligible, if there exists d such that

$$\varepsilon(\lambda) \geq /\lambda^d$$

infinitely often.

Negligible and Non-Negligible III

Examples:

- $\varepsilon(\lambda) = 1/2^{\lambda}$ negligible.
- $\varepsilon(\lambda) = 1/\lambda^{1000}$ non-negligible.
- $$\varepsilon(\lambda) = \begin{cases} 1/2^{\lambda} & \text{for odd } \lambda \\ 1/\lambda^{10000} & \text{for even } \lambda \end{cases}$$

  is non-negligible.

Stream ciphers I

- Another class of ciphers is stream ciphers. Block ciphers encrypt blocks of data, but stream ciphers encrypt only bytes or words of data (even bits).
- PGR's can be used to construct stream ciphers.
- eSTREAM is a project to "identify new stream ciphers suitable for widespread adoption",[1] organised by the EU ECRYPT network. It was set up as a result of the failure of all six stream ciphers submitted to the NESSIE project. The call for primitives was first issued in November 2004. The project was completed in April 2008. The project was divided into separate phases and the project goal was to find algorithms suitable for different application profiles.

Stream ciphers II

- The basic form of eStream stream ciphers is

$$E(k, m; r) = m \oplus \mathrm{PGR}(k; r),$$

where $\mathrm{PGR}: \{0,1\}^s \times R \longrightarrow \{0,1\}^n$, $\{0,1\}^s$ is a seed, R a nonce. The pair (k, r) is never used more than once.

- An old stream cipher is RC4. New eSTREAM ciphers are Salsa and Sosemanuk.
- **Performance** (speed MB/sec)

RC4	126
Salsa20/12	643
Sosemanuk	727
3DES	13
AES-128	109

Stream Cipher Security Definitions I

Definition

A statistical test on $\{0,1\}^n$ *is an algorithm* A *such that for* $x \in \{0,1\}^n$ $A(x)$ *outputs* 0 *or* 1. *The former means "not random", the latter "random".*

Examples:

- $A(x) = 1$ iff $|\#0(x) - \#1(x)| \leq 10 \cdot \sqrt{n}$.
- $A(x) = 1$ iff $\left|\#00(x) - \frac{n}{4}\right| \leq 10 \cdot \sqrt{n}$.
- $A(x) = 1$ iff max-run-of-0$(x) < 10 \cdot \log_2 n$.

Stream Cipher Security Definitions II

Let $G : K \longrightarrow \{0,1\}^n$ be a PRG and A a statistical test on $\{0,1\}^n$.

Definition

Define an advantage as a function Adv such that

$$Adv_{PRG}[A,G] = \left| Pr_{k \leftarrow\!\!- \mathcal{K}}[A(G(k)) = 1] - Pr_{r \leftarrow\!\!- \{0,1\}^n}[A(r) = 1] \right| \in [0,1].$$

If Adv is close to 1, then A can distinguish G from random. If Adv is close to 0, A cannot.

Example

Suppose $G : K \longrightarrow \{0,1\}^n$ satisfies $msb(G(k)) = 1$ (msb = most significant bit) for 2/3 of keys in K. Define a statistical test $A(x)$ as

$$\text{if } msb(x) = 1 \text{ output 1 else output 0.}$$

Then

$$\begin{aligned} Adv_{PRG}[A, G] &= \left| Pr_{k \xleftarrow{} \mathcal{K}}[A(G(k)) = 1] - Pr_{r \xleftarrow{} \{0,1\}^n}[A(r) = 1] \right| \\ &= \frac{2}{3} - \frac{1}{2} = \frac{1}{6}. \end{aligned}$$

Secure PRG

Definition

$G : K \longrightarrow \{0,1\}^n$ is a secure PRG, if for all statistical tests A $Adv_{PRG}[A, G]$ is negligible.

We do not know if there are provably secure PRG's, but we have heuristic candidates. We can prove:

- A secure PRG is unpredictable (easy fact).
- If PRG is predictable, then PRG is insecure.
- An unpredictable PRG is secure (Yao 1982).

Semantic Security I

- We had earlier Shannon's concept of perfect secrecy. In Shannon's definition an adversary cannot choose plaintexts and cannot send many chosen plaintexts to be encrypted. Here we develop another concept of secrecy, semantic security, where chosen plaintexts can be used to find out regularities.
- Define experiments $EXP(0)$ and $EXP(1)$ as follows:
  - An adversary chooses two plaintexts m_0 and m_1 and sends them to be encrypted by another entity, whose secret keys are not known by the adversary. It is assumed that $|m_0| = |m_1|$.
  - The other, challenger, chooses a secret key and encrypts one of the messages: $E(k, m_b),\ b = 0, 1$.
  - The adversary uses some algorithm A to decide, if the encrypted message is m_0 or m_1. Let W_b is the event that the algorithm outputs b.
- Now define the advantage of the algorithm over E as follows:

$$Adv_{SS}[A, E] = |PR[W_0] - Pr[W_1]| \in [0, 1].$$

Semantic Security II

Definition

An encryption scheme E is semantically secure, if for all efficient A $Adv_{SS}[A, E]$ is negligible.

If E is semantically secure, then for all explicit m_0, m_1,

$$Prob[E(k, m_0)] \approx Prob[E(k, m_1)].$$

Semantic Security: Examples I

- Suppose an efficient A can always deduce the least significant (lsb) of a plaintext from the corresponding ciphertext. Then E is not semantically secure.
- Suppose an adversary chooses two messages such that $lsb(m_0) = 0$, $lsb(m_1) = 1$. Then A returns always the right answer.
- Thus $Adv_{SS}(A, E) = 1 - 0 = 1$.
- On the other hand, one time pad is semantically secure:

$$\begin{aligned} Adv_{SS}[A, OTP] &= |Pr[A(k \oplus m_0) = 1] - Pr[A(k \oplus m_1) = 1]| \\ &= \frac{1}{2} - \frac{1}{2} = 0. \end{aligned}$$

Using Block Ciphers: ECB I

- The simplest way to use encryption is to divide a message into blocks, encrypt the blocks one after another, and send the encrypted blocks one by one. This mode is called Electronic Code Book.
- However, it is not the best way to do, because the same plaintext blocks will result the same encrypted blocks. This can have drastic consequences, for example when pictures are encrypted.
- We can also prove that ECB is not always semantically secure.

Using Block Ciphers: ECB II

Theorem

ECB is not semantically secure, if messages contain more than one block.

Proof. Let the adversary construct two messages, both containing two blocks.

$$\begin{aligned} m_0 &= \text{Hello World} \\ m_1 &= \text{Hello Hello} \end{aligned}$$

Challenger chooses which one of the messages he will encrypt, encrypts it and sends it back to the adversary. Now the adversary's algorithm A outputs 0, if the encrypted messages are the same, otherwise it outputs 1. Thus

$$Adv_{SS}[A, ECB] = 1,$$

and 1 is not negligible. □

Using Block Ciphers: CBC I

Instead of ECB, it is better to use chaining techniques. The most traditional is Cipher Block Chaining or CBC:

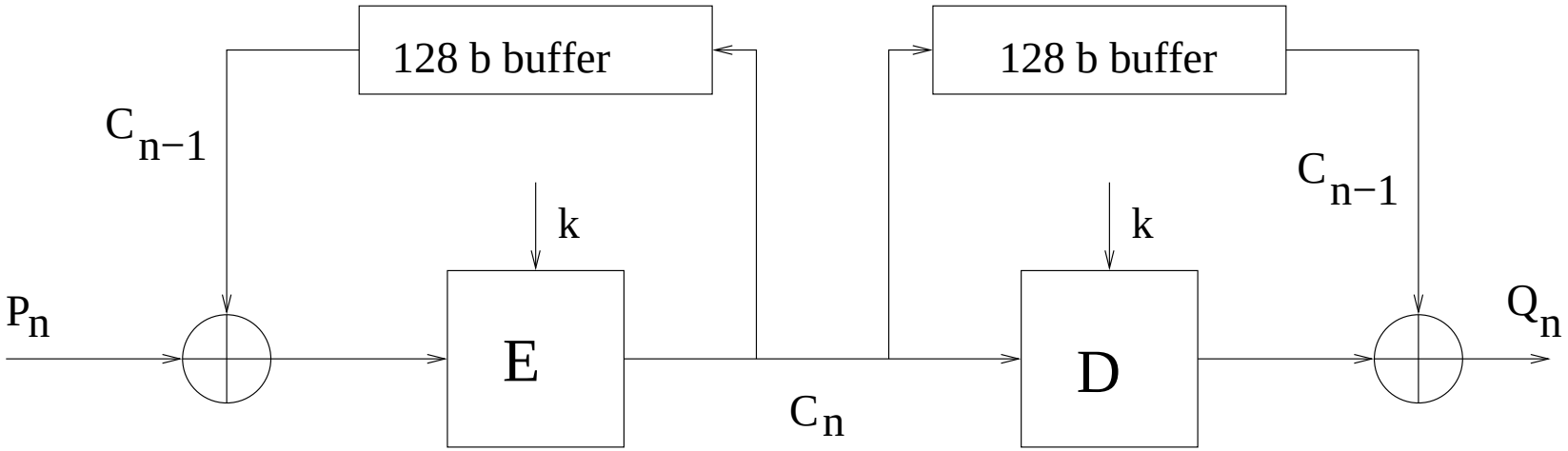

Figure: CBC

When encrypting and decrypting the first plaintext block, it is necessary to give an initial value to the buffers. This IV should be random.

Using Block Ciphers: CBC II

- If IV is known to an adversary, he can use the knowledge to launch an attack. Consider

$$\begin{aligned} C_1 &= E(k, [IV \oplus P_1]), \\ P_1 &= IV \oplus D(k, C_1) \end{aligned}$$

- Use the notation that $X[i]$ denotes the ith bit of the b-bit quantity X. Then

$$P_1[i] = IV[i] \oplus D(K, C_1)[i].$$

- Using the properties of xor, we have

$$P_1[i]' = IV[i]' \oplus D(k, C_1)[i],$$

where the prime notation denotes bit complementation.

Using Block Ciphers: CBC III

- This means that if an opponent can predictably change bits in IV, the corresponding bits of the received value of P_1 can be changed.
- IV can be generated by encrypting a nonce which may be a counter, a timestamp, message number or a random number. The nonce must be changed for every session.

Analysis of CBC I

It is possible to analyse CBC using formal security models. From these analyses, it is possible to derive how often an encryption key must be changed. Consider the following experiment.

- An adversary chooses plaintexts $m_1, m_2, ..., m_q$ and sends them to a challenger to be encrypted.
- The challenger first chooses a bit $b = 0, 1$. If $b = 0$, the challenger chooses a random key and encrypts the messages with this key and encryption scheme E.
- If $b = 1$, he chooses a random permutation f and uses it to "encrypt" the messages.
- The challenger sends the encrypted messages back to the adversary.
- The adversary tries to guess, with the with the help of an efficient algorithm A, to deduce from the encrypted messages if they are of the form $E(k, m)$ or $(f(m)$.

Analysis of CBC II

Denote by EXP(0) the case where the challenger uses $b = 0$ and the adversary guesses $b = 1$. Similarly, EXP(1) means that the challenger has used $b = 1$ and the adversary has guessed 1.

Definition

E is a secure pseudo random permutation (PRP), if for all efficient A

$$Adv_{PRP}[A, E] = |Pr[Exp(0) = 1] - Pr[Exp(1) = 1]|$$

is negligible.

This definition differs from the definition of semantic security, because now the adversary can send an arbitrary number of messages. A block cipher is secure, if it satisfies the condition in the definition. If we are going to use chaining of encrypted blocks, we still need a modification. Consider the following experiment.

Analysis of CBC III

- An adversary chooses plaintext pairs $(m_{i,0}, m_{i,1})$, $i = 1, ..., q$. He sends them to a challenger to be encrypted.
- The challenger first chooses a bit $b = 0, 1$. Then he encrypts the messages ($m_{i,b}$ with his secret key and sends the encrypted messages back to the adversary.
- The adversary tries, with the help of an efficient algorithm, to deduce from the encrypted messages if they are of the form $m_{i,0}$ or $m_{i,1}$, i.e. he tries to guess b.
- If the adversary wants to receive $c = E(k, m)$ for some message m, he sends (m, m) to the challenger.

Analysis of CBC IV

Definition

An encryption scheme E is semantically secure under CPA (chosen plaintext attack), if for all efficient A

$$Adv_{CPA}[A, E] = |Pr[Exp(0) = 1] - Pr[Exp(1) = 1]|$$

is negligible.

Analysis of CBC V

Theorem

(CBC Theorem): Let $L > 0$ be the length of messages and q the number of queries an adversary can make. If E is a secure pseudo random permutation, then E_{CBC} is semantically secure under CPA. In particular, for a q-query adversary A attacking E_{CBC} there exists a PRP adversary B such that

$$Adv_{CPA}[A, E_{CBC}] \leq 2 \cdot Adv_{PRP}[B, E] + 2q^2L^2/|X|$$

where X is the space of encrypted blocks.

It follows from the theorem that CBC is only secure as long as $q^2L^2 << |X|$. In practice, q is the number of messages encrypted with the same key. If we want

$$Adv_{CPA}[A, E_{CBC}] \leq 1/2^{32},$$

Analysis of CBC VI

then $q^2L^2/|X| < 1/2^{32}$. For AES $|X| = 2^{128}$, so qL should be less than 2^{48}. This means that after 2^{48} AES blocks the key must be changed.

Integrity and Digital Signatures I

- Usually a mere encryption is not enough, but integrity must be taken into account.
- Integrity methods can be used even without encryption. For example, when delivering free software.
- Integrity tag is calculated of the message and then added into the message. A receiver calculates also the same tag and compares his tag with the received tag. If they are the same, the message has preserved its integrity.
- A tag is usually a digital signature or a hmac value.
- Hmac is calculated using a hash function with a secret value which is known to both sides.
- A digital signature must be verified. To quarantee this, the procedure is as follows:

Integrity and Digital Signatures II

1. A sender prepares a digital signature using his secret key.
2. The signature is appended to the message. A certificate or more is usually also added to the message. It contains the sender's public key which must be used to verify the signature.
3. The receiver verifies the signature using the sender's public key.
4. After this, he must check that the certificate containing the sender's public key is not on the revocation list. If it is, the signature verification fails.
5. If the receiver has not an up-to-date revocation list, it must be loaded. This can be an expensive or slow operation, if the list is large. The other possibility is to use online checking using for example the Online Certificate Status Protocol (OCSP). Then it is not necessary to load the revocation list.
6. The receiver must still verify the authenticity of the certificate. This is done by verifying the signature of the certificate authority in the certificate.

Integrity and Digital Signatures III

7. If the receiver does not have the public key of the certificate authority or he does not know the authority at all, he must search for the certificate authority. In order to make this task easier for the receiver, the sender can send several certificates in the hope that the receiver knows at least one of them.
8. If the receiver knows one of them, then he tries to follow the certificate chain in order to verify the authenticity of the sender's public key.
9. If all these phase succeed, only then has the signature been verified.

Summary of the Requirements for the exam

The advices are valid for two years.

Modular Arithmetics

You should

- be able to prove the basic properties of modular arithmetics,
- understand and be able to construct inverse elements with respect to addition and multiplication.
- understand the concept of the primitive root and be able to find small primitive roots.

It is not necessary to remember the extended Euclidean algorithm. Pocket calculators are not needed.

Finite Fields

You should

- be able to construct finite fields of the form $GF(p^n)$,
- and to find irreducible polynomials of small degree.

Basic Problems

You should

- be able to explain the factorization problem and it relation to cryptography
- as well as the discrete logarithm problem.

AES

You should

- know the basic logic of AES and
- the functioning of SubBytes, ShiftRows and MixColumns operations.

However, it is not necessary to remember the S-box or the multiplication matrix of MixColumns.

RSA I

- You should be able to explain generally, how public and private keys are defined.
- Moreover, it is expected that you can construct public and private keys concretely when n is small.
- You should be able to explain the encryption and decryption procedures.
- It is not necessary to remember the theorems which show how RSA can be broken if some bits of the secret keys are known or if private keys are too small.
- You should be able to explain the problem of short plaintexts and how this problem is solved in practice (including OAEP).
- Remember the side channel attack against RSA!
- It is not necessary to remember the algorithm to calculate powers or requirements for the parameters.

RSA II

- Remember how digital signatures are done using RSA. Remember also the useless attack to forge signatures.

DH

- You should be able to explain the basic Diffie-Hellman method to generate keys.
- Also the man-in-the-middle attack against the basic DH.

Needham-Schroeder with a Server

- You should be able to analyze the key generating protocol of Needham and Schoeder. This means that you can find its weak points. You can explain why various parameters are needed and what extra parameters should be added so that the weakness disappears.
- You should know the concepts of forward and partial forward secrecy and resistance to key compromise impersonation. Also you can analyse some simple cases if they satisfy these concepts.

Attacks

- You should be able to explain the various attack possibilities against key agreement or security protocols.
- For reflection and typing attacks, also examples.
- You should know the possibility of certificate manipulation, but it is not necessary to remember the example.

Design Principles

- It may be possible that you have to write an essay about the design principles for cryptographic protocols.
- The same with the robust principles of public key cryptography.

Protocols I

- Remember Bellare-Rogaway and the attack against it.
- Remember the ISO/IEC 9798 protocol 4. A simple security analysis is expected.
- It is not necessary to remember by heart Andrew's Secure RPC Protocol, but you should be able to analyse it and find its flaw, if ther protocol is shown to you.
- Similarly with Burrow's modification.
- You can explain and analyse Boyd's protocol.
- You should know the Denning-Sacco improvement of Needham-Schroeder.
- You should know the ISO/IEC 11770-3 protocols and be able to explain their differences.

Protocols II

- Not necessary to remember the public key version of Needham-Schroeder, but you should be able to analyse it if it is shown to you.
- The same with Station-to Station Protocol.
- Not necessary to remember IKE, but you should have some ideas what must be taken into account in practical protocols.

Group Protocols

- Practical requirements.
- GDH.2 should be known in such a way that you can even simulate it. Not necessary to remember GDH.1 or 3.
- Remember BD with broadcasts. Not necessary to remember BD without broadcasts by heart, but you should be able to simulate it, if it is shown to you.
- The concept of the key tree, calculation principles for the blinded and private keys, join and leave operations should be known.
- Authenticated GDH.2 should be known. Not necessary to know SA-GDH.2 by heart, but you should be able to simulate it, if it is shown to you.
- Some ideas about the performance of the various protocols.

Part IV

- ECB and CBC, good and bad properties.
- Various definitions of Advantage. You should be able to calculate Adv in simple cases.
- Understanding how to very signatures.