Applied Cryptography 2

Elaboration on exam questions

Lukas Prokop

May 8, 2015

Contents

1 Differential Cryptanalysis

Explain the basic idea of a differential attack. What is a difference distribution table? How is the secret key determined in the attack?

This answer is well-researched.

Basic idea Deduce information about the secret key by tracing differences between pairs of plaintexts during the encryption (and decryption). First published by Biham and Shamir to attack DES. Is a chosen-plaintext attack and applicable to many iterated block ciphers.

Difference distribution table Let f be an n to m bit function. The difference distribution table of f is an $2^n \times 2^m$ table whose entries are the number of valid solutions x for each differential $\triangle u \rightarrow \triangle v$; hence $T(\text{input diff}, \text{output diff})$ returns the number of valid solutions.

Secret determination Construct a good differential $\triangle A \rightarrow \triangle D$ for several rounds with high probability. Make a key guess in the last round and compute backwards to get $\triangle D'$. If $\triangle D = \triangle D'$, guess was correct with probability $\Pr(\triangle A \rightarrow \triangle D)$.

Describe the 3-round differential attack on DES.

This answer is well-researched.

- SBox lookups is the only nonlinear operation. Linear expansion, XOR with the round key or permutations on bit level are linear.
- DC is a chosen plaintext attack.
- The structure of the Feistel network is defined as $R_{i+1} = L_i \oplus F_i(R_i, K_i)$ and $L_{i+1} = R_i$.
- We choose $\triangle R_0 = 0$. In the third round we get $\triangle F_2 = \triangle R_3 \oplus \triangle L_0$; so we know the input/output differences of F_2.
- The last round key can be determined by using several pairs of plaintexts with $\triangle R_0 = 0$. The remaining key bits can be determined by brute force.

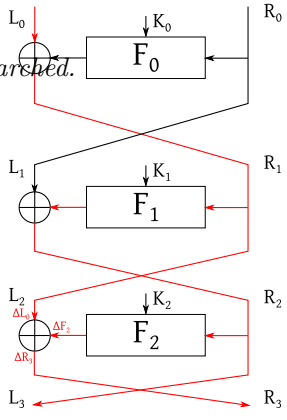

Figure 1: Structure of 3-round differential attack on DES

What is a characteristic and what is a differential? How is the probability of a differential computed (approximated)? Explain the problems associated with this approximation.

This answer is possibly inappropriate. Further discussion required.

---

characteristic A characteristic is a set of differentials. Also the corresponding probability of the characteristic can be part of the characteristic.

differential Difference between two bit values.

probability The probability that $(\triangle L_{i-1}, \triangle R_{i-1})$ is mapped to $(\triangle L_i, \triangle R_i)$ in round i is denoted as p_i. It is computed eg. using a *difference distribution table*. If three input value pairs to S-Boxes return a difference of `0x0A` of 16 possible differences, the probability p_i is given as $\frac{3}{16}$.

problems Depending on the structure (eg. Feistel structure for DES), key size and the use of non-linear elements, it might be difficult to trace differences and guess good key bits. Furthermore DC-mitigating designs achieve uniform distribution for probabilities.

A general problem is that joint probabilities are multiplied yielding impractically small values. Furthermore rounds are considered independently.

What is a Markov cipher? What is the hypothesis of stochastic equivalence?

This answer is well-researched.

A Markov cipher is a cipher satisfying the following equation (over one round):

$$\mathrm{EDP}(\triangle A, \triangle B | X) = \mathrm{EDP}(\triangle A, \triangle B)$$

where X is some input value and EDP is the Expected Differential Probability averaging the probabilities of all independent round keys. Informally this means that the expected differential probability is independent of the knowledge of the input bits. For Markov ciphers the following fundamental theorem holds:

$$\mathrm{EDP}(\triangle A, \triangle B, \triangle C, \triangle D) = \mathrm{EDP}(\triangle A, \triangle B) \cdot \mathrm{EDP}(\triangle B, \triangle C) \cdot \mathrm{EDP}(\triangle C, \triangle D)$$

Stochastic equivalence claims that for almost all keys

$$\Pr[k](\triangle A, \triangle B, \triangle C, \triangle D) \approx \mathrm{EDP}(\triangle A, \triangle B, \triangle C, \triangle D)$$

What is an impossible differential? What is a truncated differential? What is a related key differential? Describe the advantages of each differential.

This answer is well-researched.

Impossible differential Differentials with exceptionally low (zero) probability.

Truncated differential Generalization of differential cryptanalysis where the main idea is to leave parts of the difference unspecified.

Related key Exploit relations between different but related keys. Given K and K^* find a relation f such that $K = f(K^*)$. Only theoretical attack.

Advantages Impossible differentials allow us to determine wrong key guesses (if $(\delta, 0)$ is given). Truncated differentials are powerful against word/byte oriented ciphers and are often combined with extensions of differential techniques (i.e. impossible differentials, boomerang attack). Related key attacks have not been carried out (only theoretical).

Describe the impossible differential attack on 6 rounds of DES.

This answer is possibly inappropriate. Further discussion required.

- Encrypt pairs of plaintexts with difference $(\delta, 0)$
- Guess last round key and decrypt ciphertexts one round
- If we observe the difference $(0, \delta)$, then this guess for the key is wrong

Explain the idea of the Boomerang Attack.

This answer is well-researched.

- Introduced by Wagner
- Motivation: Differential probabilities decrease, the more rounds are added.
- Idea: We split the cipher into two sub-ciphers and use a differential for each sub-cipher: $E_K = E_K^2 \circ E_K^1$. We use two short high probability differentials for $\frac{R}{2}$-rounds instead of one low probability for R-rounds.
- The attack typically succeeds if the probability of a correct quartet (P, P^*, C, C^*) is higher than the probability of a wrong quartet.

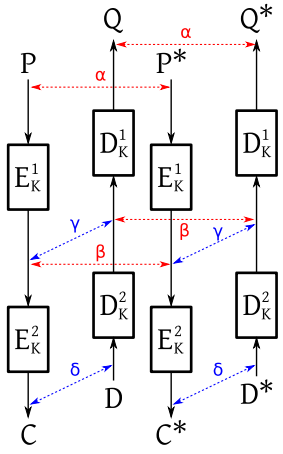

Figure 2: Boomerang attack quartet

- It is a chosen plaintext and adaptive chosen ciphertext attack.

2 Linear Cryptanalysis and Saturation Attack

Describe the basic idea of linear cryptanalysis. Explain the entries in the linear approximation table. How is the secret key determined in the attack?

This answer is well-researched.

Basic idea A linear cipher is trivially weak: each plaintext/ciphertext pair provides linear equations on the key. We use a linear function to approximate the cipher.

Linear approximation table entries Let f be an n to m bit function. The linear approximation table contains the biases $\varepsilon \cdot 2^n$ of the linear approximation $\alpha \cdot x = \beta \cdot y$ of $y = f(x)$ for all input masks α and all output masks β. The Linear Approximation Table is the linear analogous of the Difference Distribution Table.

Secret key determination We use the following algo to retrieve one key bit:

1. Initialize two counters $T_0 = 0$ and $T_1 = 0$.
2. For each plaintext/ciphertext pair (p_i, c_i) do
   (a) if $p_{i,0} \oplus p_{i,3} \oplus c_{i,2} \oplus 1 = 0$: increase counter T_0
   (b) if $p_{i,0} \oplus p_{i,3} \oplus c_{i,2} \oplus 1 = 1$: increase counter T_1
3. We then get the following information about the key:
   (a) if $T_0 > T_1$: $k_{0,0} \oplus k_{0,3} \oplus k_{1,2} = 0$
   (b) if $T_1 > T_0$: $k_{0,0} \oplus k_{0,3} \oplus k_{1,2} = 1$

Describe the Piling-up Lemma. What is it used for?

This answer is well-researched.

The Piling-Up Lemma says:

> Let X_i $(1 \leq i \leq n)$ be independent Boolean expressions with probabilities $\mathcal{P}(X_i = 0) = \frac{1}{2} + \varepsilon_i$. Then
>
> $$\mathcal{P}(X_1 \oplus X_2 \oplus \cdots \oplus X_n = 0) = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \varepsilon_i$$

When chaining linear approximations we want to determine the probability of certain values for the overall linear approximation. The Piling-Up Lemma helps us with that.

Describe the 3-round linear attack on DES.

This answer is well-researched.

We consider the DES' Feistel structure. It holds that

$$R_0[15] \oplus F(R_0, K_1)[7, 18, 24, 29] = K_1[22]$$

and

$$L_3[15] \oplus F(L_3, K_3)[7, 18, 24, 29] = K_3[22]$$

where $F(R_0, K_1) = L_0 \oplus R_1$ and $F(L_3, K_3) = R_1 \oplus R_3$. Those equations hold with probability $\frac{12}{64}$.

$$R_0[15] \oplus L_3[15] \oplus L_0[7, 18, 24, 29] \oplus R_3[7, 18, 24, 29] = K_1[22] \oplus K_3[22]$$

holds with probability 0.7. We use the Secret Key determination algorithm to determine $K_1[22] \oplus K_3[22]$.

How does the "proof of security" against differential cryptanalysis for AES work? Give the definition of the branch number.

This answer is possibly inappropriate. Further discussion required.

- "Bounds" can be computed to estimate probabilities
- For example the bound for 1 S-Box is,
  - $d = \max_{a \neq 0, b} \mathrm{EDP}(a, b)$
  - $l = \max_{a, b \neq 0} \mathrm{ELP}(a, b)$

  $\Rightarrow$ we need S-Boxes with small maximum values in DDT or LDT. Bound is given by
  - $\mathrm{EDP} \leq d^{\#\text{ active S-Boxes}}$
  - $\mathrm{ELP} \leq l^{\#\text{ active S-Boxes}}$
- Important properties:
  - Number of active components (S-boxes) before S-Box
  - Worst-case (max) differential/linear probability in S-box

The branch number B is defined as minimum number of active components in (a, b). Formally, the branch number of a $n \times n$ matrix A is defined by

$$\beta(A) = \min \left\{ w(x) + w(A \cdot x^T) \,|\, x \in (\{0,1\}^m)^n, x \neq 0 \right\}$$

where $w(x)$ is the hamming weight of x.

AES MixColumns has a branch number of 5. ShiftRows is diffusion-optimal. The number of active S-Boxes per 4 rounds is 25.

For 2 rounds of AES, ELP and EDP can be computed with brute-force.

General huge margin (max. 2^{-150} for 4-round characteristic).

Explain the Λ-set used in the Saturation Attack. Illustrate the actions of the AES components on a Λ-set.

This answer is well-researched.

The Λ-set is defined as

> A Λ-set is a set of 256 states a_t (4×4 byte arrays), where we define for byte indices i, j and state indices $t, s = \{0, \ldots, 255\}$:
>
> C: $a_t[i,j] = c \; \forall t$ (constant, passive)
>
> S: $a_t[i,j] \neq a_s[i,j]$ if $t \neq s$ (saturated, active)
>
> B: $\sum_t a_t[i,j] = 0$ (balanced)
>
> Saturated implies balanced. Constant implies balanced.

SubBytes Saturated → saturated, constant → constant, balanced → undetermined

ShiftRows Only changes indices

MixColumns Action depends on all 4 bytes of the column: $[CCCC]^t \rightarrow [CCCC]^t$, $[CCCS]^t \rightarrow [SSSS]^t$, $[BBBB]^t \rightarrow [BBBB]^t$, $[SSSS]^t \rightarrow [BBBB]^t$

AddRoundKey Everything stays the same

3 Hash Functions

Explain the Merkle-Damgard construction. Which property can be proven and what are the requirements for the proof?

This answer is possibly inappropriate. Further discussion required.

Properties:

- single-piped construction where M_i is added to the stream iteratively starting with the IV.
- So, $H(M) = f(f(\dots f(f(\text{IV}, M_1), M_2), \dots, M_{t-1}), M_t)$
- Also a finalization function might be put around $H(M)$
- f is the compression function. h (ie. $H(M)$) is the resulting hash function.
- Used by MD5, SHA-1, SHA-2, ...

requirement of proof f is collision resistant, appropriate padding scheme

can be proven f is collision resistant $\Rightarrow$ h is collision resistant

Describe at least two problems of the Merkle-Damgard construction. How can these problems be avoided?

This answer is well-researched.

The following problems are mentioned in the slides or Wikipedia:

Length extension Compute $H(x \,\|\, y)$ from $H(x)$ without knowing x; hence $f(f(\dots f(H(x), y_1), \dots, y_{t-1}), y_t)$

Multi collisions 2^t collisions can be found in $t \cdot 2^{n/2}$ plaintexts instead of the usual $2^t \cdot 2^{n/2}$.

Birthday paradox many attacks above the "birthday bound"

Second preimage attacks against long messages are always much more efficient than brute force

Herding attacks or Nostradamus attack meaning first committing to an output h, then mapping messages with arbitrary starting values to h can be applied. They are possible for more work than finding a collision, but much less than would be expected to do this for a random oracle.

Mitigations:

- adding non-trivial output transformation
- wide-pipe constructions (e.g. Chop-MD, HAIFA)
- adding counter/salt/tweak inputs to f
- Sponge construction

All in common: larger intermediate state, output transformation.

Compare differential attacks on block ciphers with differential attacks on hash functions. What is the main advantage in a hash function attack?

This answer is nonexistent or lacking details. Further research required.

Comparison TODO

Advantage In hash function attacks we can choose the message to attack one of the security properties.

Describe the 3 basic steps (linear part, nonlinear part, message modification) in the differential attack on SHA-1.

This answer is well-researched.

Linear Linear message expansion

$$W_i = \begin{cases} M_i & 0 \leq i \leq 16 \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-15}) & 16 \leq i \leq 80 \end{cases}$$

Nonlinear Large system of equations difficult to solve. First, select differences! To find non-linear characteristics use the following algorithm:

1. Start with message difference and state word difference after step 16
2. Find propagation of differences
   - using guess-and-determine
   - complexity determined by number of differences
   - primarily try zero difference
3. Determine conditions of differential characteristic

Message modification To improve complexity of attack in first few steps, modify message at the beginning.

Basic message modification (until step 16): Choose A i according to conditions, then compute

$$W_i = A_i - A_{i-5} - (A_{i-1} \lll 5) - f(A_{i-2}, A_{i-3} \lll 2, A_{i-4} \lll 2) - K_i$$

Advanced approaches:

- advanced message modifications

- equation solving
- neutral bits
- boomerang/tunnels
- greedy approach

Describe the Sponge Construction of the SHA-3 winner Keccak. Which capacity size c and hash output size n is needed to get a collision and preimage security of 128 bits? Explain why.

This answer is well-researched.

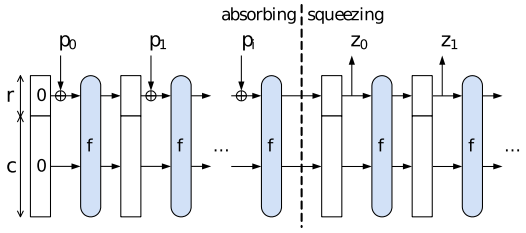

Figure 3: Sponge construction structure (CC-BY 3.0, sponge.noekeon.org)

- Uses the sponge construction which uses two elementary operations sequentially: absorbing and squeezing.
- The large state memory is divided in two sections called R (of size r) and S (of size c).
- R is xor-ed with the next r bytes of the input and S replaced by $f(S)$ repeatedly ("absorbing").
- The first r bits of R are returned as output. If further bytes are needed, apply $f(S)$ to S and repeat ("squeezing").
- c is the capacity of the hash function.
- c must be greater-equal $2n$, n-bit preimage resistance is achieved by $c = 2n$. Hence $n = 128$ and $c \geq 256$.
- Why? Because `http://keccak.noekeon.org/tune.html` ;)
  I guess the answer to provide is "the internal state memory must be large enough" (to mitigate birthday attacks I assume)

4 Authenticated Encryption

What interface, what security services does authenticated encryption (AE) provide? What additional desirable properties could an AE design offer?

This answer is well-researched.

Interface:

Encryption & Authentication
Input: plaintext M, data A, key K, nonce N
Output: ciphertext C, tag T

Decryption & Verification
Input: ciphertext C, tag T, data A, key K, nonce N
Output: plaintext M or rejection if invalid

Security services / properties:

Confidentiality No party without the key can read the message

Authenticity / Integrity Origin can be trusted and modifications of the message get detected

Further desirable properties:

- Nonce misuse resistance
- Release under unverified plaintext
- Parallelizable
- Fewer block cipher calls per block
- Key size = security level
- Online, single-pass
- Inverse-free
- Hardware/software optimization
- Lightweight
- Cheaper security against side-channel/fault attacks
- Simplicity

Describe possible generic compositions for AE. Show that two of them are not generally secure (even when using independent keys).

This answer is possibly inappropriate. Further discussion required.

Generic compositions

- Encrypt-and-MAC (E&M): $C = E^*(M), T = \text{MAC}(M)$

- Encrypt-then-MAC (EtM): $C = E^*(M), T = \text{MAC}(C)$
- MAC-then-Encrypt (MtE): $C \| T = E^*(M \| \text{MAC}(M))$

E&M and MtE are insecure for dependent keys or bad encodings. A bad encoding is explained as follows:

1. Assume super-secure MAC_K and stream cipher E^*_K
2. Build another super-secure cipher $E + +^*_K$ as follows:
   (a)
   $$M' = \begin{cases} 00 & M_i = 0 \\ 01 \text{ or } 10 & M_i = 1 \end{cases} \qquad \forall 0 \leq i \leq |\, M \,|$$
   (b) Apply $E^*_{K'}$ to M'

Attacker can then intercept ciphertext C and for every bit M_i of M:

1. toggle bit $2i$ and $2i + 1$ of C (= of M')
2. send modified $\widetilde{C^i}$ to recipient
3. if rejected: bit i was 0, else 1

Define security notions for confidentiality and integrity of AE algorithms.

This answer is well-researched.

Confidentiality

IND-CPA (indistinguishable under chosen plaintexts) infeasible for an attacker to find out which message M_0 or M_1 was encrypted to C, even if he can query encryptions of any chosen messages.

IND-CCA2 (indist. under adaptive chosen ciphertexts) like IND-CPA, but the attacker can additionally query the decryption of any ciphertext except C.

Integrity

WUF-CMA (weak unforgeability) infeasible for an attacker to find tag T for any new message M, even if he can request tags for any chosen messages $M' \neq M$.

SUF-CMA (strong unforgeability) like WUF-CMA, but attacker cannot create tag for an *existing* message.

INT-PTXT (integrity of plaintext) infeasible for an attacker to construct ciphertext C (with T) for any new message M, even if he can query encryption of chosen messages $M' \neq M$.

INT-CTXT (integrity of ciphertext) like INT-PTXT, but attacker cannot create ciphertext $C' \neq C$ for an *existing* message.

Describe two AE modes (of the six ISO/IEC standards) and their advantages/disadvantages.

This answer is well-researched.

- EtM: no known advantages or disadvantages
- CCM: Secure for ideal cipher E_K, needs no D_K (decryption), two block cipher calls per block, two-pass, not online (need length in advance), CBC-MAC not parallellizable, Not patented
- EAX: like CCM with CMAC instead of CBC-MAC, online
- GCM: fast, one block cipher call per block, hard to implement, AES-GCM is very popular, some weak keys
- OCB 2.0: faster & easier to implement than GCM, patented
- SIV (Key Wrap): nonce-free, not as efficient, data processed twice, offline

What's wrong with the following AE algorithms? Give an attack on each algorithm:

Algorithm A $C = E^*_K(M \| h(M \| N))$ where M is the message, N the nonce, E^*_K is AES-CBC (with nonce N as IV) and h is SHA-3. C is resulting ciphertext (include tag T).

Algorithm B $C = E^*_K(M \| \operatorname{MAC}_K(M))$ where E^*_K is AES-CBC and MAC_K is AES-CBC-MAC (both with nonce N as IV)

Algorithm C $C = E^*_K(M), T = \operatorname{MAC}_{K \oplus N}(C)$ where E^*_K is AES-CBC (with output length = input length and N as IV) and $\operatorname{MAC}_{K \oplus N}$ is AES-CBC-MAC (with fixed output length = security level, and IV 0)

This answer is nonexistent or lacking details. Further research required.

Algorithm B Dependent keys destroy integrity

$$C_{l+1} = E_K(T \oplus C_l) = E_K(0) \qquad \Rightarrow \text{no integrity}$$

Algorithm C The Calico attack works which means nounce N might cancel out key K in $\mathrm{MAC}_{K\oplus N}(C)$. So we apply a birthday attack on $\mathrm{MAC}_{K\oplus N}(C)$. Let n be the length of the key K.

Offline Phase:

1: **for** i in 0 to $2^{n/2}-1$ **do**
2: Compute T for empty M under $N=0$, $K_{\mathrm{MAC}} = i \,\|\, 0$
3: Save pair (T, K_{MAC}) in a list
4: **end for**

Online Phase:

1: **for** i in 0 to $2^{n/2}-1$ **do**
2: Request T for empty M under $N=j$
3: **if** $(T, i \,\|\, 0)$ is in list **then**
4: $K_{\mathrm{MAC}} = i \,\|\, j$ is a key candidate.
5: **end if**
6: **end for**

5 Algebraic Attacks

Describe the ideas and problems of translating the individual AES steps to equations. Over which field are the equation variables selected, in which steps is this problematic?

This answer is well-researched.

To attack a (symmetric) cryptosystem, write it as polynomial equation system and solve for the key.

SubBytes Nonlinear operation, affine mapping $Ax+t$ in $\mathbb{F}_2^8$: complicated!

ShiftRows permutation (easy)

MixColumns many variables

AddRoundKey Add key in $\mathbb{F}_{2^8}$: key schedule

Then derive equations over $\mathbb{F}_{2^8}$ by blowing up state. $\sim 2^{12}$ quadratic equations over $\mathbb{F}_{2^8}$. Solving this equation system is NP-hard.

Describe the idea of Linearization and Relinearization.

This answer is well-researched.

Linearization

Done with XL (Extended Linearization) or XSL (E. Spare L.)
In general $D = 2^n$.

- Let $f_1 = 0, \ldots, f_m = 0$ be the current equations
- Multiply each f_i with monomials up to result degree bound D
- Linearize new polynomials
- Perform Gaussian elimination
- If there is an equation in only one variable, solve it
- Repeat until completely solved

Relinearization
Problem: Not enough linearly independent equations.
Idea: Add more equations. But how?
General way to get more equations: Multiply existing equations with monomials.

What is a Gröbner basis? How is it useful for solving equations? Briefly describe the role of S-polynomials and Buchberger's Algorithm.

This answer is well-researched.

Gröbner basis Multivariate division of any polynomial in the ideal I by G gives 0.

Usefullness A (reduced) Gröbner Basis G of a polynomial equation system is another system with the same solutions which is easier to solve.

S-polynomial Let lcm be the least common multiple and lt be the leading term. It is sufficient to show that all S-polynomials $S(g_i, g_j)$ reduce to 0:

$$S(p,q) = \frac{\operatorname{lcm}(\operatorname{lt} p, \operatorname{lt} q)}{\operatorname{lt} p} \cdot p - \frac{\operatorname{lcm}(\operatorname{lt} p, \operatorname{lt} q)}{\operatorname{lt} q} \cdot q$$

Buchberger's algorithm Take some polynomials $F = \{f_1, f_2, \ldots, f_{m'}\}$ and the algorithm will provide a Gröbner base $G = \{g_1, \ldots, g_m\}$ of $\langle F \rangle$ with $F \subseteq G$.

1. $G = F$
2. For each pair (p, q) with $p \neq q \in G$: Divide $S(p, q)$ by G and if remainder is not zero, add it to G.
3. Repeat until no changes occur.

What is a Boolean function and its algebraic normal form (ANF) and degree? What does it mean to "derive" a Boolean function? How does it affect the degree?

This answer is well-researched.

A boolean function maps boolean values to a boolean value.

$$f : \mathbb{F}_2^n \to \mathbb{F}_2$$

The algebraic normal form is defined as XOR of ANDs (example: $(a \wedge b) \oplus c \oplus 1 \oplus (a \wedge b \wedge c)$). The degree of a boolean function is defined as polynomial degree of its ANF.

The derivative of $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ with respect to $a \in \mathbb{F}_2^n$ is defined as

$$\frac{\mathrm{d}}{\mathrm{d}a} f(x) = f(x) + f(x + a)$$

For derivatives it holds that

$$\deg\left(\frac{\mathrm{d}}{\mathrm{d}a} f(x)\right) \leq \deg(f(x)) - 1$$

What is a zero-sum, what is a cube? Explain the zero-sum distinguisher for 16 rounds of Keccak-f.

This answer is well-researched.

Zero sum A zero-sum of size K for f: $\mathbb{F}_2^n \to \mathbb{F}_2^n$ is a subset $\{x_1, \ldots, x_n\} \subseteq \mathbb{F}_2^n$ such that

$$\sum_{1 \leq i \leq K} x_i = \sum_{1 \leq i \leq K} f(x_i) = 0$$

Cube k-dimensional affine subspace in $\mathbb{F}_2^n$ with $n - k$ bits fixed, k bits loop through all 2^k values.

Distinguisher A cube is put into the middle of the Keccak-f function f. We combine 6 forward and 10 backward rounds. Zero-sum size $2^{1025} \to 2^{1600-1025} = 2^{575}$ partitions. Description complexity for a partition: 2^{1025}.

6 Factoring

Explain how the integer factorization problem is related to the security of the RSA cryptosystem. Elaborate on this relationship by showing that knowing either $\varphi(n)$ or d allows to factor n.

This answer is well-researched.

$$c^d \equiv m \mod n \tag{1}$$

$$ed \equiv 1 \mod \varphi(n) \tag{2}$$

$$1 < d < \varphi(n) \tag{3}$$
$$\varphi(n) = (p-1)(q-1) \tag{4}$$
$$n = pq \tag{5}$$

In RSA, an attacker wants to retrieve the secret parameter d to decrypt messages using equation 1. Secret d satisfies equation 2 and 3. e and n are public. d can be determined efficiently from equation 2 given $\varphi(n)$ (and e, of course). Hence determining d or $\varphi(n)$ is desirable.

$$\begin{aligned}
\varphi(n) &= pq - p - q + 1 \\
&= n - p - q + 1 \\
q &= n + 1 - p - \varphi(n) \\
n &= pq \\
&= p(n + 1 - p - \varphi(n)) \\
&= -p^2 + pn - p\varphi(n) + p
\end{aligned}$$

Solving this quadratic equation immediately yields both factors p and q.

For e we rewrite the congruence equation with:

$$\begin{aligned}
ed + k\varphi(n) &= 1 \\
ed &= 1 + l\underbrace{(p-1)}_{\text{even}}\underbrace{(q-1)}_{\text{even}} && [l \in \mathbb{Z}] \\
ed - 1 &= 2^s \cdot k && [k \text{ is odd}]
\end{aligned}$$

Given e and d obtaining $\varphi(n)$ is a little bit more difficult, but feasible. A probabilistic Las-Vegas algorithm exists which is based on the following two lemmas:

Lemma. Let $a \in \mathbb{Z}$ and $\gcd(a, n) = 1$.

$$\Rightarrow \operatorname{ord}_n(a^k) = 2^t \text{ (a power of 2)}$$

Proof.

$$\begin{aligned}
1 \equiv a^{ed-1} \equiv a^{2^s \cdot k} \equiv (a^k)^{2^s} \\
\Rightarrow \operatorname{ord}_n(a^k) \mid 2^s \\
\Rightarrow \operatorname{ord}_n(a^k) = 2^t \qquad t \leq s
\end{aligned}$$

Lemma. If $\operatorname{ord}_p(a^k) \neq \operatorname{ord}_q(a^k)$.

$$\Rightarrow \exists t \in \{1, \ldots, s-1\} : 1 < \gcd\left(\left(a^k\right)^{2^t} - 1, n\right) < n$$

Proof. Let p be the factor with the larger order: $\operatorname{ord}_p(a^k) > \operatorname{ord}_q(a^k)$.

$$2^s \geq \operatorname{ord}_p(a^k) > \operatorname{ord}_q(a^k) = 2^t$$

$$\left(a^k\right)^{2^t} - 1 \equiv 0 \bmod q \quad \Rightarrow \quad q \,|\, (a^k)^{2^t} - 1$$

$$\left(a^k\right)^{2^t} - 1 \not\equiv 0 \bmod p \quad \Rightarrow \quad p \nmid (a^k)^{2^t} - 1$$

$$\Rightarrow \gcd\left((a^k)^{2^t} - 1, pq\right) = \gcd\left((a^k)^{2^t} - 1, n\right) = q$$

$$n = pq \Rightarrow q \,|\, n \text{ and } p \,|\, n$$

The algorithm is probabilistic:

Input: n, e, d
Output: p, q with $n = p \cdot q$

1: Determine s and k with $e \cdot d - 1 = 2^s \cdot k$ with k odd
2: Choose arbitrary $a \in \mathbb{Z}$
3: **if** $g = \gcd(a, n) > 1$ **then**
4: **return** $p = g$ and $q = \frac{n}{g}$
5: **end if**
6: Calculate $g = \gcd\left(\left(a^k\right)^{2^t - 1}, n\right)$ for $t = 1, \ldots, s - 1$
7: **if** $g > 1$ **then**
8: **return** $p = g$ and $q = \frac{n}{g}$
9: **end if**
10: Goto Step 2

Joint work with Martina Tscheckl, thanks!

Explain Pollard's $(p-1)$ method and apply it to factor 33.

This answer is well-researched.

Idea:

- Let $k \in \mathbb{N}$ with $i|k \quad \forall i \in \mathbb{N}^+$ with $i \leq B$. B is a smoothness bound. p is a prime divisor of n such that $p - 1$ is a product of small prime powers $\leq B$.
- k is a multiple of all of the prime powers in the factorization of $p - 1$. It follows that k is a multiple of $p - 1$.
- By Fermat's Little Theorem, we have $a^k \equiv 1 \mod p$. Then p divides $\gcd(a^k - 1, n)$.
- If $a^k \equiv 1 \mod n$, we fail to get a nontrivial factor of n.

Algorithm:

1: $n = 33$ ▷ number to factor
2: $B = 2$ ▷ random, constantly increasing
3: **loop**
4: $k = \text{lcm}(\text{range}(1, B + 1))$
5: $a = \text{random.choice}(\text{range}(2, n - 1))$
6: $d = \gcd(a^k - 1, n)$
7: **if** $d > 1$ and $d \neq n$ **then**

8: **return** d
9: **end if**
10: $B = B + 1$
11: **end loop**

Factoring 33:

1. $n = 33$
2. $B = 8$
3. $k = \text{lcm}(1, \ldots, 15) = 360360$
4. $a = 3$
5. $d = \gcd\left(3^{360360} - 1, 33\right) = 11$

Explain the Fermat factorization. What are the consequences of the Fermat factorization for RSA? Use Fermat factorization to factor 39.

This answer is well-researched.

Idea:

- A 1-by-1 correspondence is given between $n = ab$ and $n = s^2 - t^2$ with

$$t = \frac{a+b}{2} \qquad s = \frac{a-b}{2} \qquad a = t + s \qquad b = t - s$$

Consequences:

- Picking two primes close to each other might provide parameters vulnerable to Fermat factorization.

Algorithm:

1: $n = 39$ ▷ number to factor
2: $s = \lceil \sqrt{2} \rceil$
3: $t^2 = s^2 - n$
4: **while** $\nexists x \in \mathbb{Z} : t^2 = x^2$ **do**
5: $s = s + 1$
6: $t^2 = s^2 - n$
7: **end while**
8: $d = s - \sqrt{t^2}$
9: **return** d

Factorization:

1. $n = 39$
2. $s = 6$

3. $t\_sq = -3$
4. $s = 7$
5. $t\_sq = 10$
6. $s = 8$
7. $t\_sq = 25$
8. $d = 3$

Explain what the difference between the so-called dark age factoring methods and the modern factoring methods are. Give an example how the required congruences can be found.

This answer is well-researched.

- Most modern algorithms use a smoothness bound B and a factor base of all prime numbers smaller B. Relations on the factor base are than searched for (according to Elisabeth Oswald). Dark Age algorithms were more diverse and often exploit exponential behavior.
- Eg. Trial Division, $(\rho - 1)$ method, $(\rho + 1)$ method and Pollard's rho-method are considered Dark Age methods.
- One specific relation on factor base is the congruence of squares. Given prime number n, find x and y satisfying $x^2 - y^2 = n$ with $x \not\equiv \pm y \mod n$. This is interesting because it reveals the factorization $(x + y)(x - y)$.
- The weaker congurence of squares problem tries to determine a $x^2 \equiv y^2 \mod n$ with $x \not\equiv y \mod n$. One example to find such a congruence is to solve the problem $x^2 \equiv y \mod n$. We then multiply several small prime numbers of y together to retrieve a square on the right-hand side.

Explain what is meant by sieving. Sieve the polynomial $f(x) = x^2 + 1$ for $0 < x \leq 10$.

This answer is possibly inappropriate. Further discussion required.

Sieving describes the general technique to count or estimate the number of sifted sets of integers. Important sieves are the Sieve of Eratosthenes/Atkin, Quadratic sieve, General Number Field Sieve and Large Sieve.

I have no clue how this works. The following was written down on the slides of the last year.

For every prime number smaller than our smoothness bound, we check whether $f(x) \equiv 0 \bmod p$. We check the previous observations to identify all numbers $x + kp$ such that $f(x + kp)$ is also divisible by p.

			2,5		2,13		$2{,}5^2$	5,13	2	
	1	1	0	1	0	1	0	0	0	1
x	1	2	3	4	5	6	7	8	9	10
$f(x)$	2	5	10	17	26	37	50	65	82	101

Explain the working principle of the quadratic sieve.

This answer is well-researched.

Find x and y such that

$$y^2 = x^2 \mod n \qquad x \not\equiv y \mod n$$

Then $n \mid x^2 - y^2$ but $n \nmid x + y$ and $n \nmid x - y$ with $x^2 - y^2 = (x + y)(x - y)$. $\gcd(x - y, n)$ yields a non-trivial divisor of n and can be computed by the Euclidean algorithm.

1. Choose smoothness bound B where $\pi(B)$ denotes the number of prime numbers less than B.
2. Using sieving determine $\pi(B) + 1$ numbers a_i such that $b_i = (a_i^2 \bmod n)$ is B-smooth.
3. Factor the b_i and generate exponent vectors mod 2 for each one.
4. Use linear algebra to find a subset of these vectors which add to the zero vector. Multiply corresponding a_i together mod n (giving a) and the b_i together (giving B-smooth square b^2).
5. We are now left with the equality $a^2 = b^2 \pmod n$ from which we get two square roots of $(a^2 \bmod n)$, one by taking the square root in the integers of b^2 namely b, and the other from a computed in step 4.
6. We now have the desired identity: $(a + b)(a - b) = 0 \pmod n$. Compute the GCD of n with the difference (or sum) of a and b. This produces a factor, although it may be a trivial factor (n or 1). If the factor is trivial, try again with a different linear dependency or different a.

Factor 33 using the quadratic sieve.

1. We select some smoothness bound. Let's say $B = \lfloor \sqrt{33} \rfloor = 5$.

2. We create a table with columns a_i and $a_i^2 \mod n$ and -1 and $\pi(B)$ where $\pi(B)$ denotes the prime numbers less than B. We successively create more and more rows until we have enough rows we can use in the next step (too many rows is always fine, but might require a greater smoothness bound). We initialize a_i of each row with $a_i + 1$ from the previous row beginning with $\lceil\sqrt{33}\rceil$.

3. The items of the table are filled according to the usage of the prime numbers. We have 1 in the first row, because $3^1 = 3$. We have 4 in row number 2, because $2^4 = 16$. We have two ones in row three, because $-1^1 \cdot 2^1 = -2$. We have two ones in row four, because $3^1 \cdot 5^1 = 15$. And finally $2^1 \cdot 5^1 = 10$.

a_i	$a_i^2 \mod n$	-1	2	3	5
6	3			1	
7	16		4		
8	-2	1	1		
9	15			1	1
10	-32	1	5		

   Working with negative numbers can be skipped (modulos naturally does not return negative numbers), but it works anyway and can help to be faster when writing down the notes. To be honest, if we don't use negative numbers for $a_i = 10$ here, we have to increase the smoothness bound.

4. We now select rows such that *all* columns provide an even sum. If we select $a_i = 6, 7, 9$ we have sums 4 (for column 2), we have sum 2 (for column 3) and sum 1 (for column 5). So because sum 1 is not even, this selection is inappropriate. But selection of rows 8 and 10 is one working example.

5. We create a relation $x^2 \equiv y^2$. Therefore we take $-1^1 \cdot -1^1 \cdot 2^1 \cdot 2^5 = -1^2 \cdot 2^6 = (-1 \cdot 2^3)^2 = (-8)^2 = 8^2$

$$(8 \cdot 10)^2 \equiv 8^2 \pmod{33}$$

$$80^2 \equiv 8^2 \pmod{33}$$

$$31 \equiv 31 \pmod{33}$$

6. $\gcd(80 - 8, 33) = 3$. 3 is a non-trivial divisor of 33. Success!

7 Discrete Logarithm Problem

Give the definition of the DLP in a finite multiplicative group. Explain how the security of the DLP is related to a generator of the group. Explain how the DLP is related to the representation of the group.

This answer is well-researched.

Discrete Logarithm Problem. Given a prime p, a generator g of $\mathbb{Z}_p^*$, and an element y in $\mathbb{Z}_p^*$, find the integer $x \in \{0, \ldots, p-2\}$ such that $g^x \equiv y \bmod p$.

The DLP difficulty is independent of the generator, because conversion between logarithms is possible. But having an efficient algorithm to solve the DLP in one group does not imply an efficient algorithm for the other group (dependence on representation)!

The DLP in $(\mathbb{Z}_n, +)$ is easy: you need to solve $a \cdot x \equiv y \mod n$. You can determine x by using the extended euclidean algorithm. However, this does not affect the security of $(\mathbb{Z}_{2^{255}-19}, \cdot)$ as Elliptic curve.

What methods do you know to solve the DLP in a finite multiplicative group? Name two different methods and briefly explain their working principle.

This answer is well-researched.

Methods:

- Exhaustive search
- Babystep-Giantstep method
- Pollard-ρ method
- Pohlig-Hellman method
- Index-Calculus

Exhaustive search:

Goal given $y \equiv g^x$ in group G of order n, find x.

Idea try if $g^1, g^2, g^3, \ldots$ matches y

Optimize precompute index table with the smallest powers of generator g

Explain the working principle of the baby-step giant-step algorithm in detail. What is the time and memory complexity of this method?

This answer is well-researched.

Given the DPL problem $g^x \equiv y \mod p$ we can rewrite it as $y(g^{-m})^i \equiv g^j \mod p$ where $x = mi + j$ with $m = \lceil \sqrt{p} \rceil$ and $0 \leq i \leq m$ and $0 \leq j \leq m$. Precompute some g^j, fix m and trial multiplication is applied to i.

Formally with table size $\mathcal{O}(\sqrt{n})$ and time $\mathcal{O}(\sqrt{n})$,
Input: prime p, a generator g of $\mathbb{Z}_p^*$, and an element y in $\mathbb{Z}_p^*$.
Output: integer $x \in \{0, \ldots, p-2\}$ such that $g^x \equiv y \bmod p$.

1: $s = \lceil (\sqrt{p}) \rceil$
2: Initialize map S (baby steps)
3: Initialize map T (giant steps)
4: **for** all i with $0 \leq i < s$ **do**
5: $\quad S[y \cdot g^i \mod p] = i$
6: $\quad T[g^{(i+1)\cdot s} \mod p] = i$
7: **end for**
8: **for** intersection of keys of S and T: k **do**
9: $\quad$ **return** $(T[k] + 1) \cdot s - S[k]$
10: **end for**

Use the baby-step giant-step algorithm to determine the discrete logarithm of $3 = 5^x \pmod{37}$.

This answer is well-researched.

1. $s = 7$
2. $S : 1 \to 2, 3 \to 0, 5 \to 3, 14 \to 5, 15 \to 1, 25 \to 4, 33 \to 6$
3. $T : 7 \to 3, 11 \to 5, 13 \to 6, 15 \to 4, 18 \to 0, 23 \to 2, 28 \to 1$
4. $\text{keys}(S) \cap \text{keys}(T) = \{15\}$
5. return $(4 + 1) \cdot 7 - 1 = 34$

Explain the working principle of the Pohlig-Hellman algorithm. What is the complexity of this algorithm?

This answer is well-researched.

Goal Solve DLP $y \equiv g^x$ in group G with order n.

Approach Solve DLP in smaller subgroups.

Factorize the group order: $n = \prod_{1 \leq j \leq r} p_j^{e_j}$ with $e_j \geq 0$
for $i = 0$ to r **do** ▷ Find $x = \sum_{0 \leq i \leq e_j - 1} l_i p_j^i$ for each $p_j^{e_j}$
$\quad$ Let $q = p_i$ and $e = e_i$
$\quad$ Let $\gamma = 1$ and $l_{-1} = 0$
$\quad$ Determine $\overline{\alpha} = \alpha^{\frac{n}{q}}$
$\quad$ **for** $j = 0$ to $e - 1$ **do**
$\quad\quad$ $\gamma = \gamma \alpha^{l_{j-1} q^{j-1}}$ and $\overline{\beta} = (\beta \gamma^{-1})^{\frac{n}{q^{j+1}}}$
$\quad\quad$ $l_j = \log_{\overline{\alpha}} \overline{\beta}$
$\quad$ **end for**
$\quad$ $x_i = l_0 + l_1 q + \ldots + l_{e-1} q^{e-1}$
end for
Chinese Remainder Theorem: determine $x \equiv x_i \bmod p_i^{e_i}$ for $1 \leq i \leq r$
return x

Complexity $\mathcal{O}(\sqrt{n})$ for a group of order n

Use the Pohlig-Hellman algorithm to determine the discrete logarithm of $13 = 5^x \pmod{37}$**.**

This answer is well-researched.

1. Prime factorization of $n = 37 - 1 = 2^2 \cdot 3^2$
2. $x_0 = 2^0 l_0 + 2^1 l_1$
   (a) For l_0:
   $$\begin{aligned} \beta^{\frac{p-1}{q_0}} &\equiv \alpha^{\frac{p-1}{q} l_0} \\ 13^{\frac{36}{2}} &\equiv 5^{\frac{36}{2} l_0} \bmod 37 \\ 36 &\equiv 36^{l_0} \\ l_0 &= 1 \end{aligned}$$
   (b) For l_1:
   $$\begin{aligned} \beta_1 \equiv \beta\alpha^{-x_0} &= 13 \cdot 5^{-1} = 13 \cdot 15 = 10 \\ \beta_1^{\frac{p-1}{q_1}} &\equiv \alpha^{\frac{p-1}{q} l_1} \\ 10^{\frac{36}{2^2}} &\equiv 5^{\frac{36}{2} l_1} \bmod 37 \\ 1 &\equiv 36^{l_1} \\ l_1 &= 0 \end{aligned}$$
   (c) $x = 2^0 \cdot 1 + 2^1 \cdot 0 = 1 \bmod 2^2$
3. $x_1 = 3^0 l_0 + 3^1 l_1$
   (a) For l_0:
   $$\begin{aligned} \beta^{\frac{p-1}{q_0}} &\equiv \alpha^{\frac{p-1}{q} l_0} \\ 13^{\frac{36}{3}} &\equiv 5^{\frac{36}{3} l_0} \bmod 37 \\ 10 &\equiv 10^{l_0} \\ l_0 &= 1 \end{aligned}$$
   (b) For l_1:
   $$\begin{aligned} \beta_1 \equiv \beta\alpha^{-x_0} &= 13 \cdot 5^{-1} = 13 \cdot 15 = 10 \\ \beta_1^{\frac{p-1}{q_1}} &\equiv \alpha^{\frac{p-1}{q} l_1} \\ 10^{\frac{36}{3^2}} &\equiv 5^{\frac{36}{3} l_1} \bmod 37 \\ 10 &\equiv 10^{l_1} \\ l_1 &= 1 \end{aligned}$$

(c) $x = 3^0 \cdot 1 + 3^1 \cdot 1 = 4 \bmod 3^2$

4. CRT yields

$$\begin{aligned} x &\equiv 1 \bmod 4 \\ x &\equiv 4 \bmod 9 \\ \hline x &\equiv 13 \bmod 36 \end{aligned}$$

5. $13 \equiv 5^{13} \bmod 37$

Explain the working principle of the index-calculus algorithm. How is it related to modern factoring methods?

This answer is nonexistent or lacking details. Further research required.

Similar approach to modern factoring algorithms (sieving):

1. Choose a factor base $B = \{p_1, p_2, \ldots, p_B\}$
2. Determine discrete logarithm of primes in B
3. Compute discrete logarithm of y based on discrete logarithm of primes in B

Tasks 1, 2 are precomputation, Task 3 is repeated for each query.

The complexity is close to the complexity of number field sieves: $\mathcal{O}(e^{(c+o(1))\log n^{\frac{1}{3}}\log\log n^{\frac{2}{3}}})$

Use the index-calculus algorithm to determine the discrete logarithm of the number $32 = 3^x \pmod{43}$.

This answer is well-researched.

1. $32 = 3^x \bmod 43$. Hence $\alpha = 3, \beta = 32, G = \mathbb{Z}/43$. We will use the factor base $\{2, 3, 5\}$.
2. We build the relations:

$k = 1$	$3^1 \equiv 3^1$	$\Rightarrow$	$1 \equiv 1$	$\bmod 42$
$k = 24$	$3^{24} \equiv 16 = 2^4$	$\Rightarrow$	$24 \equiv 4\log_3 2$	$\bmod 42$
$k = 27$	$3^{27} \equiv 2 = 2^1$	$\Rightarrow$	$27 \equiv 1\log_3 2$	$\bmod 42$
$k = 29$	$3^{29} \equiv 18 = 2 \cdot 3^2$	$\Rightarrow$	$29 \equiv \log_3 2 + 2 \cdot \log_3 3$	$\bmod 42$
$k = 37$	$3^{37} \equiv 20 = 2^2 \cdot 5$	$\Rightarrow$	$37 \equiv 2\log_3 2 + \log_3 5$	$\bmod 42$

Given the equation system on the right, a valid solution is:

$$\log_3 2 = 27 \qquad \log_3 3 = 1 \qquad \log_3 5 = 25$$

3. We will try $k = 1$.

$$\begin{aligned} \beta \cdot \alpha^k &= 32 \cdot 3^1 \equiv 10 \bmod 43 \\ 10 &= 2^1 \cdot 5^1 \\ \log_3(32) &= (1 \cdot \log_3 2 + 1 \cdot \log_3 5) - 1 \\ &= 51 \\ &\equiv 9 \bmod 42 \end{aligned}$$

4. $32 = 3^9 \bmod 43$

8 Quantum Cryptography

"A quantum system is in a superposition of states." What does this sentence mean? How is this principle exploited in quantum computers?

This answer is possibly inappropriate. Further discussion required.

Consider light. If you examine light as particle system, you might observe behavior of particles in an experiment. If you examine light as system of waves, you might observe behavior of waves. Light retains a wave-particle duality.

Observables have probabilistic distributions and outcome cannot be predicted by 100%. The system is in a superposition of states meaning that two or several states are used according to some probability distribution. This distribution is complex (consists of amplitude and phase) and non-constant in time.

TODO: exploited in quantum computers?

In general superposition occurs when two separate states add up to a new (stronger) superstate.

What is the basic idea of Shor's algorithm to factor integer numbers? Which operations are performed using a quantum computer? What is the resulting complexity?

This answer is well-researched.

- Choose random number $a < N$
- Compute $g = \gcd(a, N)$
- If $g > 1$ then g is factor of N
- Find $r = \text{ord}(a)$ (i.e. find r such that $a^r = 1 \mod N$) [done with quantum circuit]

- If r is odd, start again
- If $a^{r/2} = -1 \mod N$, start again
- The factors are $\gcd(a^{r/2} \pm 1, N)$

Resulting complexity: $\mathcal{O}((\log_2 N)^3)$ time and $\mathcal{O}(\log_2 N)$ space

Which (cryptographic) problems can be solved using Grover's search algorithm. What is the complexity? What are the disadvantages compared to Shor's algorithm?

This answer is well-researched.

Algorithmic goal: Searching an unsorted database with N entries in $\mathcal{O}(N^{1/2})$ time and using $\mathcal{O}(\log N)$ storage space. Cryptographically it is interesting to compute $f(x)$ for all x in some domain. Then we can use Grover's search algorithm to retrieve x where $f(x) = y$ for some given y.

The complexity is given by $\mathcal{O}(\sqrt{n})$ compared to $\mathcal{O}(1)$ in classical computing.

Disadvantages:

- Not as devastating as the factoring method: still exponential
- Requires a quantum circuit implementation of the function f.

Explain secure communication using quantum bits in form of polarized photons. Describe the eavesdropper's problem.

This answer is well-researched.

We use light/photons to transmit messages. The first axis of the photons is the direction of propagation. The second axis of the photons is the polarization which can be *rectangular* or *circular*. The receiver of the signal has to select the appropriate filter for the used polarization. If the wrong filter is selected, the signal is interpreted in 50% of the cases.

Protocol: Send qubits via channel and receiver tells which filters he used to receive message. Sender confirms which bits were (accordingly) measured correct. Use those bits as session key.

Eavesdropping (or "measurement of the qubit") destroys the qubit. So the eavesdropper must use always the correct filter to receive the full message and forward a copy of the message. Practically impossible for longer key agreements. If wrong message is forwarded, it will be noticed, because sender and receiver cannot communicate.

Explain reconciliation and privacy amplification.

This answer is well-researched.

Fixes the problem that quantum lines are not noise-free.

Reconciliation Sender and receiver split up key into blocks of length k (goal: one block contains at most 1 error). Receiver checks parity bit of one block received via a public channel. If a difference in parity is found then a binary search is performed to find and correct the error.

Privacy amplification Information reconciliation revealed parts of the key. We make it difficult again to know the key.

We compute how much information Eve can have about the key. From a set of universal hash functions we select one and use our transmitted key as input and the hash function returns a shorter key. The size of the output depends on the publicly known information. This new value of the function is considered as new key.

9 Continued Fractions and Lattices

What is a continued fraction of a number? What is the n-th convergent of a number? How can continued fractions be applied to factoring?

The continued fraction expansion of a number $\alpha \in \mathbb{R}^+$ is

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\dots}}}} = [a_0; a_1, a_2, a_3, \dots] \tag{6}$$

It is in some way the "best" approximation of α. The n-th convergent of $\alpha = [a_0; a_1, a_2, \dots] \in \mathbb{R}^+$ is

$$\frac{p_n}{q_n} = [a_0; a_1, a_2, \dots, a_n]$$

The application can be found in the continued fraction factoring method. It considers the square candidates $Q_k \equiv p_k^2 - nq_k^2 \equiv p_k^2 \bmod n$, where $\frac{p_k}{q_k}$ is the k-th convergent of n.

Explain the idea of Wiener's attack on RSA.

This answer is well-researched.

Secret key d appears in convergents of $\frac{p}{q}$ if ...

- primes satisfy $q < p < 2q$,

- public exponent satisfies $e < \varepsilon(N)$.
- small private exponent satisfies $d < \frac{1}{3}\sqrt[4]{N}$

Then, d can be recovered by trying convergent candidates.

Use Wiener's method to recover the private RSA key d, if the public RSA modulus is $n = 391$ and the public exponent is $e = 235$. Assume $m = 2$ and try to decrypt $c = 2^{235} \equiv 246 \bmod 391$ with your possible choices of d.

This answer is nonexistent or lacking details. Further research required.

What is a lattice? What are basic and desirable properties of a lattice basis?

A lattice is a subset $\Lambda \subseteq \mathbb{R}^n$. It satisfies

$$\Lambda = \mathbb{Z}a_1 + \cdots + \mathbb{Z}a_d$$

where $a_1, \ldots, a_d \in \mathbb{R}^n$ are $\mathbb{R}$-linearly independent vectors.

The *lattice dimension/rank* d and the *basis* $a_1, \ldots, a_d$ are basic properties of a lattice. Properties of lattices bases:

- In general, there is an infinite number of bases for a lattice.
- If $a_1, \ldots, a_d$ is a basis, another basis is $a_1, \ldots, a_{i-1}, a_i + ka_j, a_{i+1}, \ldots, a_d$ with $i \neq j$ and $k \in \mathbb{Z}$.
- Transition from one basis to another in general: multiply A with unimodular matrix M over $\mathbb{Z}$ ($\det(M) = \pm 1$)

Short vectors and orthogonality are desirable properties.

$$\langle x, y \rangle = 0 \qquad \text{orthogonality}$$

What is an LLL-reduced lattice basis? Describe the two basic steps of the LLL algorithm.

This answer is nonexistent or lacking details. Further research required.

Explain the basic idea of the lattice version of Bleichenbacher's attack on PKCS#1.5 padding.

This answer is nonexistent or lacking details. Further research required.

10 Pairings

What are bilinear pairings in cryptography and which are their basic properties? What assumption does their security rely on? What security applications are they used for?

This answer is well-researched.

Bilinear pairings Bilinear pairings are bilinear maps transforming two elements from two input groups to an element in the target group. A bilinear map is linear with respect to elements in both input groups.

Basic properties Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of order n. Let G_1, G_2 and G_3 be the generators of $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$. Typically $\mathbb{G}_1$ and $\mathbb{G}_2$ have additive notation whereas $\mathbb{G}_T$ has multiplicative notation.

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

So two points from two elliptic curve groups are mapped to the multiplicative group of a finite field.

Security
- Given a multiplicative cyclic group of order n with generator g.
- Computational Diffie-Hellman problem: given g^a, g^b, compute g^{ab}.
- Decisional Diffie-Hellman problem: determine whether $ab = c \mod n$ given g, g^a, g^b and g^c.

Applications
- Privacy-preserving protocols
- Short/blank/proxy/ring/group... signature schemes
- Identity- and attribute-based encryption
- Threshold cryptosystems
- ...

How are bilinear pairings typically constructed? Explain briefly using the pairings by Barreto and Naehrig. How do the parameters have to be chosen to meet a certain security level? (embedding degree, group order, prime) What layers does a typical pairing implementation consist of?

This answer is nonexistent or lacking details. Further research required.

What general four types of pairings exist? How do they differ? Are they all secure? Which are the practically most relevant?

This answer is well-researched.

Type 1
- $\mathbb{G}_1 = \mathbb{G}_2$
- Efficient homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ is available.
- Hashing to $\mathbb{G}_2$ is fast.
- Elements in $\mathbb{G}_1$ tend to be relatively large.
- Relatively difficult to find suitable curves.
- Practically dead: Slow on large prime fields, quasi-polynomial bound for index-calculus attack

Type 2
- $\mathbb{G}_1 \neq \mathbb{G}_2$
- Hashing to $\mathbb{G}_2$ is expensive.
- Efficiently computable group homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ available.

Type 3
- $\mathbb{G}_1 \neq \mathbb{G}_2$
- Hashing to $\mathbb{G}_2$ is efficiently possible.
- There is no efficient group homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$
- Fast type of pairings.

Type 4
- $\mathbb{G}_1 \neq \mathbb{G}_2$
- Not used. Is a special case where $\mathbb{G}_2$ is set to be the whole n-torsion $E[n]$.
- Hashing to $\mathbb{G}_2$ is feasible, but not very efficient.

Explain the three-party key agreement in one round using pairings.

This answer is well-researched.

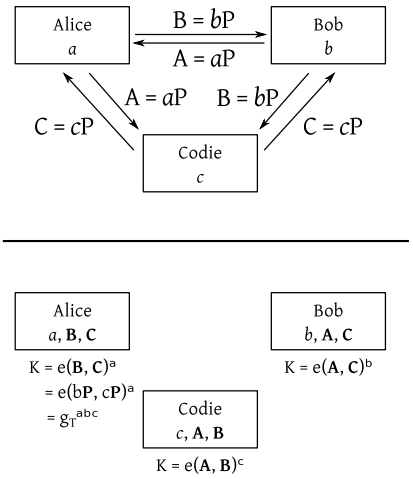

Figure 4: Three-party key agreement scheme in one round using pairings

11 Random Number Generation and Key Derivation

Explain the difference between TRNGs, PRNGs and Hybrid RNGs. List a few construction examples for each type.

This answer is well-researched.

PRNG "Deterministic random bit generator", algorithmic, efficient & deterministic & periodic

TRNG "Hardware random number generator", completely unpredictable physical source of randomness, inefficient & non-deterministic & non-periodic,

Hybrid RNG Combination of TRNG & PRNG

Constructions:

PRNG Mid-Square method, dev/random

TRNG Clock jitter, Metastability, thermal noise

Hybrid RNG parallel & sequential constructions

Give application examples of RNGs in cryptography. What properties should an RNG (PRNG or TRNG) satisfy? (How) Can these properties be tested?

Application examples:

- key generation
- encryption
- masking protocols
- internet gambling

Properties:

- Values are uniformly distributed (k-bit word should have k-bit entropy)
- Impossible to predict future values

Testsuites:

- Tests published by Donald E. Knuth
- Diehard tests
- Tests by National Institute of Standards and Technology
- eg. Frequency Monobit Test

What are the security requirements for a password hashing function (resp. for a key derivation function)? Why are standard cryptographic hash functions not necessarily ideally suited for this purpose? Explain salting.

This answer is well-researched.

Password hashing function requirements:

- Hard to invert
  - (second) preimage resistant
  - collision resistant
- fast to compute hash of a given input
- Variable length input
- Include additional local parameter (salt, ...)
- Configurable computational difficulty

Key Derivation Function requirements:

- Hard to invert
- Variable length input

- Include additional local parameter (salt, ...)
- Configurable computational difficulty
- Variable length output to match key size

Not necessarily ideally suited?

- It is fast and easy to compute the rainbow table for all popular password

Salting:

- We add additional data called "salt" to the user-provided password.
- Salt increases length of the input.
- Salt mainly protects against precomputation (dictionaries, rainbow tables).

What is memory hardness and sequential memory hardness? Explain how scrypt implements this property.

This answer is well-researched.

Memory hardness A memory-hard algorithm on a Random Access Machine is an algorithm which uses $S(n)$ space and $T(n)$ operations, where $S(n) \in \Omega\left(T(n)^{1-\varepsilon}\right)$

Sequential memory hardness A sequential memory-hard function is a function, which

- can be computed by a memory-hard algorithm on a Random Access Machine in $T(n)$ operations
- cannot be computed on a Parallel Random Access Machine with $S^*(n)$ processors and $S^*(n)$ space in expected time $T^*(n)$, where $S^*(n)T^*(n) = \mathcal{O}(T(n)^{2-x})$ for any $x > 0$.
  $\Rightarrow$ for any PRAM $S^*(n)T^*(n) = \Omega(T(n)^2)$

scrypt

1. Uses PBKDF2 to convert a password into a bitstream.
2. Feeds the bitstram to ROMix.
3. Feed the output of ROMix back to PBKDF2 to generate the derived key.

scrypt is conjectured to be sequential memory hard.

Name and explain some possible improvements over scrypt (e.g., features of yescrypt, Lyra2, Makwa).

This answer is well-researched.

Problems in mass-authentication deployment with scrypt:

- Defender: increased memory usage, decreased CPU usage.
  - Memory becomes bottleneck on multiple parallel instances computing scrypt
- Hashing on the defenders side becomes inefficient.
  - Not all CPU cores might be used fully.
  - Must decrease memory usage, to scale → less security

Overview:

1. First generation KDFs (PBKDF2, bcrypt) employ only time based work factor
   - Advantage for attacker with specialized hardware
2. scrypt introduced memory cost and TMTO analysis
   - Defends against specialized hardware
3. Lyra2 is an extension to the concepts introduced by scrypt
4. Makwa employs a radically different concept
   - Ideas from asymmetric cryptography
   - Enables useful new features