CRYPTOGRAPHY AND CODING THEORY

1+2. Algebraic basics

<table>
<tr><td><b>Integer</b></td><td>$\mathbb{Z}=\{0,\pm1,\pm2,\pm3,\dots\}$</td><td>$15\in\mathbb{Z}$</td></tr>
<tr><td><b>Divisibility</b></td><td>Let a and n be integers with $a\neq 0$, <b>a divides n</b> if and only if there is an integer b such that $n=a*b$: $a|n \Leftrightarrow \exists b\in\mathbb{Z}: n=a*b$</td><td>$5|15$</td></tr>
<tr><td rowspan="3">properties</td><td>$a|0, a|a, 1|n$</td><td>$5|0,\quad 5|5,\quad 1|5$</td></tr>
<tr><td>If $a|b$ and $b|c\ then\ a|c$</td><td>$5|15\ and\ 15|60 \rightarrow 5|60$</td></tr>
<tr><td>If $a|b\ and\ a|c \Rightarrow a|(s*b+t*c)$ for all integers s and t</td><td>$3|6\ and\ 3|15$
$\Rightarrow 3|(2*6+4*15)$</td></tr>
<tr><td><b>Division theorem</b></td><td>If a and b are integers with $b>0$ then there are unique integers q (quotient) and r (rest) such that $a=q*b+r\ and\ 0\leq r<b$</td><td>$9=2*4+1$
$0\leq 1\leq 4$</td></tr>
<tr><td>notation
q=intDiv(a,b)
r=mod(a,b)</td><td>$q=\left\lfloor\frac{a}{b}\right\rfloor\ (floor\ function=abrunden)$
$r=a-b*q=a\ mod\ b$</td><td>$q=\left\lfloor\frac{9}{4}\right\rfloor=2$
$r=9-4*2=1$</td></tr>
<tr><td><b>Greatest Common Divisor (gcd)</b></td><td>largest non-negative integer d that divides both a and b: $\gcd(a,b)$
Special case: $\gcd(0,0)=0\rightarrow per\ definition$</td><td>$\gcd(18,30)=6$
$\gcd(-10,20)=10$</td></tr>
<tr><td rowspan="4">properties

gcd(a,b)</td><td>$\gcd(a,0)=|a|$ note: also valid for $a=0$</td><td>$\gcd(-10,0)=10$</td></tr>
<tr><td>$\gcd(a,b)\geq 0$</td><td>$\gcd(-20,-14)=2$</td></tr>
<tr><td>For any integer q: $gcd(a+q*b,b)=\gcd(a,b)$
"adding a multiple of one integer to the other does not change their gcd"</td><td>$\gcd(3+8,4)=\gcd(3,4)$
$\gcd(3+6,2)=\gcd(3,2)$</td></tr>
<tr><td>$if\ b\neq 0, we\ may\ choose\ q=-\left\lfloor\frac{a}{b}\right\rfloor\rightarrow a+q*b=a-\left\lfloor\frac{a}{b}\right\rfloor*b=a\ mod\ b$
$gcd(a\ mod\ b,b)=gcd(a,b)$</td><td>$\gcd(14\ mod\ 6,6)$
$=\gcd(14,6)$</td></tr>
<tr><td><b>Euclidean algorithm</b>

/gcdstep(a,b)</td><td>Based on $\gcd(a,b)=\gcd(b\ mod\ a,a)\ \ and\ \ \gcd(a,0)=|a|$
<code>gcd(a,b):</code>
<code> while (a!=0):</code>
<code>   r=b mod a; b = a; a = r</code>
<code> return b</code></td><td><table><tr><th></th><th>a</th><th>b</th><th>r</th></tr><tr><td rowspan="4">gcd(15,25)
= 5</td><td>15</td><td>25</td><td>10</td></tr><tr><td>10</td><td>15</td><td>5</td></tr><tr><td>5</td><td>10</td><td>0</td></tr><tr><td>0</td><td>5</td><td></td></tr></table></td></tr>
<tr><td><b>Extended Euclidean Algorithm</b></td><td>The set of all integer linear combinations of two integers a and b coincides with the set of all integer multiples of $\gcd(a,b)$
$a*\mathbb{Z}+b*\mathbb{Z}=\gcd(a,b)*\mathbb{Z}$</td><td>$4*2+6*1$
$=\gcd(4,6)*7$</td></tr>
<tr><td>in other words:</td><td>For any given integers a,b,n the equation $ax+by=n$ can be solved by integers x and y if and only if $\gcd(a,b)\,|\,n$ $a*x+b*y=\gcd(a,b)$</td><td>$\mathbf{4}*(-1)+\mathbf{6}*1=2$
$\mathbf{4}*?+\mathbf{6}*?\neq 3$</td></tr>
<tr><td>egcd(a, b):

/egcd(a,b)
/egcdstep(a,b)</td><td><code>x=0, y=1, u=1, v=0</code>
<code>while(a!=0):</code>
<code> q=</code>$\left\lfloor\frac{b}{a}\right\rfloor$<code>, r=b mod a, m=x-u*q, n=y-v*q</code>
<code> b=a, a=r, x=u, y=v, u=m, v=n</code>
<code>return b, x, y</code></td><td><table><tr><th>a</th><th>b</th><th>x</th><th>y</th><th>u</th><th>v</th><th>q</th><th>r</th><th>m</th><th>n</th></tr><tr><td>$\mathbf{4}$</td><td>$\mathbf{5}$</td><td>0</td><td>1</td><td>1</td><td>0</td><td>–</td><td>–</td><td>–</td><td>–</td></tr><tr><td>1</td><td>4</td><td>1</td><td>0</td><td>−1</td><td>1</td><td>1</td><td>1</td><td>−1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>−1</td><td>1</td><td>5</td><td>−4</td><td>4</td><td>0</td><td>5</td><td>−4</td></tr></table>$4*(-1)+5*1=1$</td></tr>
<tr><td><b>modular arithmetic</b></td><td>$19:00+8:00=27:00\rightarrow 03:00$ … represent the same time
We say that 3,27,51, … <b>are congruent module 24</b></td><td></td></tr>
<tr><td>Congruences

mod(a,n)</td><td>Let a,b,n be integers with $n\neq 0$. We say that
$a\equiv b(mod\ n)$ "$a\ is\ congruent\ to\ b\ modulo\ n$"
If $(a-b)$ is a multiple (positive or negative) of n.
$a\equiv b(mod\ n)\Leftrightarrow a=b+n*k, k\in\mathbb{Z}$</td><td>$3\equiv 27(mod\ 24)$

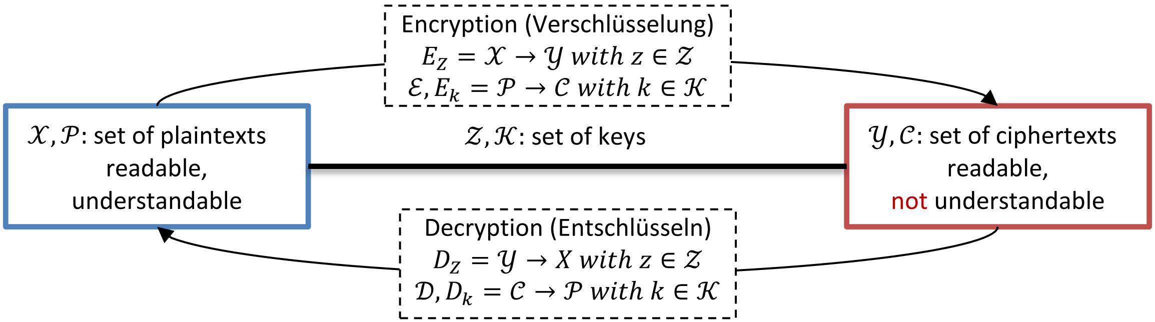
$3=27+(-1)*24$</td></tr>
<tr><td>addition, subtraction and multiplication</td><td>$Suppose\ a\equiv b(mod\ n)\ and\ c\equiv d(mod\ n)\ Then$
$a+c\equiv b+d(mod\ n)$
$a-c\equiv b-d(mod\ n)$
$a*c\equiv b*d(mod\ n)$</td><td>$a=3, c=2, n=7$
$3+2\equiv 10+9(mod\ 7)$
$3-2\equiv 10-9(mod\ 7)$
$3*2\equiv 10*9(mod\ 7)$</td></tr>
<tr><td rowspan="2"><b>Modular Inverses</b>

Be careful with division!

/ecgd(a,b)</td><td>Let a, n be integers with $n\neq 0$
If the congruence $a*x\equiv 1\ (mod\ n)$ has a solution $x\in\mathbb{Z}$,
we say a is invertible modulo n
and x is the multiplicative inverse for $a\ (mod\ n)$</td><td>$3*?\equiv 1(mod\ 5)$
$3*0\ (mod\ 5)\equiv 0$
$3*1\ (mod\ 5)\equiv 3$
$3*\mathbf{2}\ (mod\ 5)\equiv\mathbf{1}$
<b>mod inv of 3 mod 5 is 2</b></td></tr>
<tr><td>The integer a is invertible module n if and only if $\gcd(a,n)=1$
Since $\gcd(a,n)=1$ ther exist integer x and y such that $a*x+n*y=1$</td><td>$\gcd(3,5)=1$
$3*2+5*(-1)=1$</td></tr>
<tr><td>solving</td><td>$a*x\equiv b\ (mod\ n)$
$\gcd(a,n)=1\rightarrow$ use extended Euclidean algorithm to find s and t
$\gcd(a,n)=c>1\rightarrow\left(\frac{a}{c}\right)*x\equiv\frac{b}{c}\left(mod\frac{n}{c}\right)$ →solutions: $x_0, x_0+\frac{n}{c}, x_0+\frac{2n}{c}$</td><td>$5*4\equiv 6\ (mod\ 7)$
$5*3+7*(-2)=1$</td></tr>
<tr><td></td><td>a must be coprime (teilerfremd) with n.
Therefore, we use prime numbers, because they are coprime except 0.</td><td><b>mod inv of 2 mod 6 not exist</b>
$2*1+6*0=2$</td></tr>
</table>

Fermat's Little Theorem	If p is a prime, then for every integer a $a^p \equiv a \ (mod\ p)$ If p is a prime and p does not divide a (coprime), then $a^{p-1} \equiv 1 \ (mod\ p)$ Attention: There may be exponents $e < p - 1$ such that $a^e \equiv 1 \ (mod\ p)$	$2^5 \equiv 32 \equiv 2 \ (mod\ 5)$ $2^{5-1} \equiv 16 \equiv 1 \ (mod\ 5)$
usage	What is the remainder of $2^{10203}\ mod\ 101$? $\rightarrow 2^{100} \equiv 1 \ (mod\ 101)$ $2^{10203} \equiv (2^{100})^{102} * 2^3 \equiv (1)^{102} * 2^3 \equiv 2^3 \equiv 8 \ (mod\ 101)$	
coprime (or relatively prime)	Two integers a and b are coprime (teilerfremd) if $\gcd(a,b) = 1$	$\gcd(4,9) = 1$
Euler's Phi-funct.	$\phi(n) =$ number of integers $1 \leq a \leq n$, such that $gcd(a,n) = 1$	$\phi(6) = 2 \rightarrow \{1,2,3,4,5,6\}$
properties /phi(n) /phi(7)	$\phi(p) = p - 1, p \in \mathbb{P}$	$\phi(7) = 6 \rightarrow \{1,2,3,4,5,6,7\}$
	$\phi(p * q) = p * q \quad \underbrace{-q}_{mit\ p\ teilbar,} \quad \underbrace{-p}_{mit\ q\ teilbar,} \quad \underbrace{+1}_{da\ p*q\ 2mal\ gezählt} \quad , p, q \in \mathbb{P}, p \neq q$	$\phi(2 * 3) = 6 - 2 - 3 + 1$ $= 2$
	$\phi(p^n) = p^n - p^{n-1} = p^{n-1} * (p - 1)$	$\phi(2^3) = 8 - 4 = 4 * 1$ $\{1,2,3,4,5,6,7,8\}$
	$\phi(m * n) = \phi(m) * \phi(n), \quad \gcd(m,n) = 1$	$\phi(2 * 3) = \phi(2) * \phi(3) = 2$
	$n = p_1^{e_1} * p_2^{e_2} * \dots * p_k^{e_k},\ p_i \in \mathbb{P}, p_i \neq p_j\ für\ i \neq j$ $\phi(n) = \phi(p_1^{e_1}) * \phi(p_2^{e_2}) * \dots * \phi(\ p_k^{e_k})$ $\phi(n) = \prod_{i=1}^{k} p_i^{e_{i-1}} * (p_i - 1) = n * \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right)$	$225 = 3^2 * 5^2$ $\phi(225) = \phi(3^2) * \phi(5^2)$ I must know the prime factors.
Euler's Totient Theorem	if a und n are positive integers and relatively prime: $a^{\phi(n)} \equiv 1 \ (mod\ n)$	$\gcd(3,4) = 1$ $3^{\phi(4)} \equiv 3^2 \equiv 9 \equiv 1 \ (mod\ 4)$
properties	if n is prime: $a^{\phi(p)} \equiv a^{p-1} \equiv 1 \ (mod\ p)$ →Fermats little theorem	$3^4 \equiv 81 \equiv 1 \ (mod\ 5)$
usage	What are the "last two digits" of $123^{562} \rightarrow mod\ 100$ which is not prime $Euler's\ theorem: m^{\phi(100)} \equiv 1 \ (mod\ 100)$ and $\gcd(123,100) = 1$ $123^{\phi(100)} \equiv 123^{40} \equiv 1 (mod\ 100)$ $123^{562} \equiv (123^{40})^{14} * 123^2 \equiv 1 * 123^2 = 23^2 = 29 \ (mod\ 100)$	
Multiplicative Order	The multiplicative order of $g\ mod\ n$ is the smallest positive integer e that: $g^e \equiv 1 \ (mod\ n), g \in \mathbb{Z}$	$g = 2, n = 5$ $2^1 \equiv 2 \ (mod\ 5)$ $2^2 \equiv 4 \ (mod\ 5)$ $2^3 \equiv 8 \equiv 3 \ (mod\ 5)$ $2^{\mathbf{4}} \equiv 16 \equiv 1 \ (mod\ 5)$ $ord(2) = 4 \ (mod\ 5)$
properties /multord(g,n) /multord(8,5)	$g^f \equiv 1 \ (mod\ n),\ f \in \mathbb{N}$, if and only if f is divisible by the order e of g	$2^8 \equiv 1 \ (mod\ 5)$
	$g^k \equiv g^l \ (mod\ n)$, if and only if $k \equiv l \ (mod\ e)$	$2^{101} \equiv 2^{301} \ (mod\ 5)$ da $101 \equiv 301 \ (mod\ 4)$
	$g^k = \frac{e}{\gcd(e,k)}, \quad k \in \mathbb{N}$ $ord(2^6) \equiv ord(4), da\ 2^6 \equiv 64 \equiv 4 \ (mod\ 5)$	$ord(2^2) = \frac{4}{\gcd(4,2)} = 2$ $ord(2^3) = \frac{4}{\gcd(4,3)} = 4$
Generators module p generator / primitive element /gen(g,p) /gen(2,7)	$p \in \mathbb{P}, g \in \{1,2,\dots,p-1\}$ g is a generator $mod\ p$ if: $g^i\ mod\ p$ with $1 \leq i \leq p - 1$ generates $1,2,\dots,p-1$ $\rightarrow g$ is a generator if the order of $g\ mod\ p$ is $p - 1$ There are generators for any prime p. The number of generators $mod\ p$ is given by $\phi(p - 1)$	$g = 2, p = 7$ $2^1 \equiv \mathbf{2}$ $2^2 \equiv 4$ $2^3 \equiv 1$ $2^4 \equiv \mathbf{2}$ $\rightarrow no$ $ord(2) = 3$ $g = 3, p = 5$ $3^1 \equiv 3$ $3^2 \equiv 4$ $3^3 \equiv 2$ $3^4 \equiv 1$ $\rightarrow yes$ $ord(3) = 4$
Chinese Remainder Theorem (Chinesischer Restwertsatz) crypt/ $\text{chin}\begin{pmatrix} a_1 & m_1 \\ \dots & \dots \\ a_n & m_n \end{pmatrix}$	$x \equiv a_1 \ (mod\ m_1)$ $x \equiv a_2 \ (mod\ m_2)$ $x \equiv a_n \ (mod\ m_n)$ $\quad \gcd(m_i, m_j) = 1, \quad i \neq j$ $M = \prod_{i=1}^{n} m_i = m_1 * m_2 * \dots * m_n$ $M_i = \frac{M}{m_i} = m_1 * m_2 * \dots * m_{i-1} * m_{i+1} * \dots * m_n \rightarrow \gcd(m_i, M_i) = 1$ $\rightarrow r_i * m_i + s_i * M_i = \gcd(m_i, M_i) = 1, \quad e_i = s_i * M_i \ (mod\ M)$	$x = 5 \ (mod\ 7)$ $x = 3 \ (mod\ 11)$ $x = 10 \ (mod\ 13)$ $M = 7 * 11 * 13 = 1001$ $M_1 = 143 \rightarrow e_1 = 715$ $M_2 = 91 \rightarrow e_2 = 364$ $M_3 = 77 \rightarrow e_3 = 924$
	$x = \left(\sum_{i=1}^{n} a_i * e_i\right) mod\ M$	$x = 894$

3a. Symmetric Cryptography

Terms	Cryptos = hidden (from Greek) Desire of confidentiality -> protection from disallowed reading.	
Tasks	**Integrity** (Integrität) = Ensure that nobody has changed the received document. **Authenticity** (Authentifizierung) = Ensure who has sent this document. **Indisputable** (unbestreitbar) = Ensure that, this person has done that.	
Cryptography	 Encryption (Verschlüsselung) $E_Z = \mathcal{X} \rightarrow \mathcal{Y}$ with $z \in \mathcal{Z}$ $\mathcal{E}, E_k = \mathcal{P} \rightarrow \mathcal{C}$ with $k \in \mathcal{K}$ $\mathcal{X}, \mathcal{P}$: set of plaintexts readable, understandable $\mathcal{Z}, \mathcal{K}$: set of keys $\mathcal{Y}, \mathcal{C}$: set of ciphertexts readable, not understandable Decryption (Entschlüsseln) $D_Z = \mathcal{Y} \rightarrow X$ with $z \in \mathcal{Z}$ $\mathcal{D}, D_k = \mathcal{C} \rightarrow \mathcal{P}$ with $k \in \mathcal{K}$	
Symmetric	The key d to decrypt can easily be computed from the key e to encrypt.	
Attacks	The Attacker knows the algorithm.	
Kerckhoffs's principle	The security of an encryption system rests solely on the secrecy of the key. And not on the missing knowledge of the algorithm.	
Scenarios	**Ciphertext-only:** attacker knows only the ciphertext (most difficult) **Known Plaintext**: he also knows some part of the plaintext (realistic) **Chosen Plaintext**: try by myself, with chosen input **Brute force**: Try all combinations -> key space needs to be large	h,a,e,g,s,d,f weather forecast a,a,a,a,a -> x,x,x,x,x a,b,c,d, ... ->
Goal	Determine the key z in use.	
Block Ciphers (Verschlüsselung)	We have an alphabet $\mathcal{A}$ of plain text and cipher text symbols n: fixed block length $\mathcal{X} = \mathcal{A}^n$: set of plaintexts $\mathcal{Y} = \mathcal{A}^n$: set of ciphertexts does not say how long the key is	e.g. $\mathcal{A} = \{0,1\}$ *or* $\{a \dots z\}$ e.g. 64-bit code
requirements	Encryption = Permutation = change bit order **Injective** (one-by-one): $f(x) = f(y) \rightarrow x = y$, otherwise, two equally plaintext would result in the same ciphertext. **Surjective** (onto): $y \in \mathcal{Y} \rightarrow \exists x \in \mathcal{X}: f(x) = y$, otherwise, there would be valid ciphertexts without valid plaintexts. → **Bijective** Self-Mapping (Injective and Surjective)	e.g. shuffle cards each $\mathcal{P}$ has one unique $\mathcal{C}$ each $\mathcal{C}$ has at least one $\mathcal{P}$ $\mathcal{C} \leftrightarrow \mathcal{P}$
Linear functions	$\mathcal{A} = \mathbb{Z}_m = \{0,1,\dots,m-1\}$ all computations are modulo m, to ensure that result is between 0 and $m-1$	e.g. $\mathcal{A} = \{0..25\}, m = 26$
linear	Scalars: $\alpha, \beta \in \mathbb{Z}_m$ Vectors: $\vec{v}, \vec{w} \in (\mathbb{Z}_m)^n$ Function $f: (\mathbb{Z}_m)^n \rightarrow (\mathbb{Z}_m)^k$ $f(\alpha\vec{v} + \beta\vec{w}) = \alpha * f(\vec{v}) + \beta * f(\vec{w})$	
affine = linear + bijective /invmod(m,n) /invmodstep	Map M: $(k \times n)$-matrix with entries $\mathbb{Z}_m$ b: vector in $(\mathbb{Z}_m)^k, b = 0 \rightarrow f$ *is linear* $f(\vec{v}) = \left(M\vec{v} + \vec{b}\right) \bmod m$ an affine map is bijective if: 1. $k = n$ 2. $\gcd(\det(M), m) = 1 \rightarrow (\det(M))^{-1} (mod\ m)$ *exists* determinant of M must be coprime with m	
determinant det()	The factor of area changes when multiplying with a position vector. If negative we flip the area (antisymmetric) $2 \times 2 \rightarrow$ calculate $a_x * b_y - b_x * a_y$ $3 \times 3 \rightarrow$ Hand rule of Sarrus	
	Unity matrix does not change a vector when multiplying. -> $det = 1$	
Confusion (Verwirrung)	$y_i = F_i(\vec{x}, \vec{z}), i \in \{1 \dots n\}$ F_i should be mathematically complex -> linear functions are not enough For a given x and y, it is not feasible to solve for z. -> do this with different rounds (enough big): $E = E_R \circ E_{R-1} \circ \dots \circ E_1$	
Diffusion (Streuung)	Every ciphertext bit should depend on every plaintext and every key bit. -> Changing a single bit in the plaintext (or the key), on the average 50% of the ciphertext bits should change	

<table>
<tr>
<td><b>Alg: Vigenère Cipher</b>
von Julius Caesar

affine encryption</td>
<td colspan="2">Encryption: $E_Z: (\mathbb{Z}_m)^n \to (\mathbb{Z}_m)^n,\ \vec{v} \to \vec{v} + \vec{z}\ (mod\ m)$
Decryption: $D_Z: (\mathbb{Z}_m)^n \to (\mathbb{Z}_m)^n, \vec{v} \to \vec{v} - \vec{z}\ (mod\ m)$
<table>
<tr><th></th><th>Variation 1</th><th>Variation 2</th><th>Variation 3</th><th>One-Time-Pad</th></tr>
<tr><td><b>Plaintext</b></td><td>$a, b, c, \dots, x, y, z$</td><td>$a, b, c, \dots, x, y, z$</td><td>$a, b, c, \dots, x, y, z$</td><td>010001101110</td></tr>
<tr><td><b>Ciphertext</b></td><td>$d, e, f, \dots a, b, c$</td><td>$e, i, x, \dots, a, b, k$</td><td>e.g. Apfel</td><td>101101010110</td></tr>
<tr><td><b>Number of key</b></td><td>26</td><td>$26! = 4 * 10^{26}$</td><td></td><td>long as plaintext</td></tr>
<tr><td><b>Encryption</b></td><td>Shift to right</td><td>Randomly permutate
<table>
<tr><th>a</th><th>b</th><th>c</th><th>…</th><th>x</th><th>y</th><th>z</th></tr>
<tr><td>e</td><td>f</td><td>x</td><td>…</td><td>h</td><td>u</td><td>k</td></tr>
</table></td><td>
<table>
<tr><th>a</th><th>b</th><th>c</th><th>.</th><th>x</th><th>y</th><th>z</th></tr>
<tr><td>a</td><td>b</td><td>c</td><td>.</td><td>x</td><td>y</td><td>z</td></tr>
<tr><td>p</td><td>q</td><td>r</td><td>.</td><td>m</td><td>n</td><td>o</td></tr>
<tr><td>f</td><td>g</td><td>h</td><td>.</td><td>c</td><td>d</td><td>e</td></tr>
<tr><td>e</td><td>f</td><td>g</td><td>.</td><td>b</td><td>c</td><td>d</td></tr>
<tr><td>l</td><td>m</td><td>n</td><td>.</td><td>i</td><td>j</td><td>k</td></tr>
</table></td><td>add a random key
e.g.
111100111000</td></tr>
<tr><td><b>Brute force attack</b></td><td>easy, only #26
too little keys</td><td>difficult, but possible
word structure</td><td>ciphertext too short
word structure</td><td>secure proven
key to long</td></tr>
<tr><td><b>Example</b></td><td>(+3) haus -> kdxv</td><td>zac -> kex</td><td>zac -> zph</td><td></td></tr>
</table>
</td>
</tr>
<tr>
<td><b>Alg: Hill Cipher</b></td>
<td>$\mathcal{Z}$: set of all invertible $n \times n$ matrices with components from $\mathbb{Z}_m$
matrix must be invertible: $\gcd(\det(M), m) = 1$
Key: $M \in (\mathbb{Z}_m)^{n \times n}$
$$E_M: (\mathbb{Z}_m)^n \to (\mathbb{Z}_m)^n, \vec{v} \to M * \vec{v}\ (mod\ m)$$
Linear permutations of vector of length n</td>
<td></td>
</tr>
<tr>
<td><b>Alg: General Affine Cipher</b></td>
<td>Key: (M, b)
M: invertible Matrix in $(\mathbb{Z}_m)^{n \times n}$
b: vector in $(\mathbb{Z}_m)^n$
Encryption: $E_{(M,b)}: (\mathbb{Z}_m)^n \to (\mathbb{Z}_m)^n, v \to Mv + b\ (mod\ m)$
Special Cases:
$M = 1$: Vignère
$b = 0$: Hill
Every affine encryption is solvable.</td>
<td></td>
</tr>
</table>

4. Algebraic basics 2

<table>
<tr><td><b>Algebraic Group</b></td><td>A group is a set G together with a binary operation $\circ$,
which combines two elements of G.</td><td>G = Set of Integer $\mathbb{Z}$
$\circ$ = addition '+'</td></tr>
<tr><td>properties</td><td>Closure (Abgeschlossenheit): $a, b \in G \Rightarrow a \circ b \in G$
Associativity: $(a \circ b) \circ c = a \circ (b \circ c)$
Identity Element e (Einheitselement): $e \circ a = a \circ e = a$
Inverse Element a^{-1}: $a^{-1} \circ a = a \circ a^{-1} = e$</td><td>$a + b \in \mathbb{Z}$
$(a + b) + c = a + (b + c)$
$0 + a = a + 0 = a$
$(-a) + a = a + (-a) = 0$</td></tr>
<tr><td>Abelian Group</td><td>Commutative Group $a \circ b = b \circ a$</td><td>$a + b = b + a$</td></tr>
<tr><td><b>Algebraic Field</b>
(Körper)</td><td>A field is a set F together with two binary operations $\oplus$ and $\otimes$,
satisfying the properties</td><td>$F = Rational\ numbers\ \mathbb{Q}$
$\oplus = ' +', \otimes = ' *'$</td></tr>
<tr><td>properties</td><td>$(F, \oplus)$ is an Abelian Group.
The identity element with respect to $\oplus$ is denoted by 0.</td><td>$(\mathbb{Q}, +)$
$e = 0$</td></tr>
<tr><td></td><td>$(F - \{0\}, \otimes)$ is an Abelian Group.
The identity element with respect to $\otimes$ is denoted by 1.</td><td>$(\mathbb{Q} - \{0\}, \otimes)$
$e = 1$</td></tr>
<tr><td></td><td>Distributive Law holds: $a \otimes (b \oplus c) = a \otimes b \oplus a \otimes c$</td><td>$a * (b + c) = ab + ac$</td></tr>
<tr><td>Remarks</td><td>$\oplus$ is commonly called "addition", $\otimes$ is commonly called "multiplication"</td><td></td></tr>
<tr><td></td><td>We can solve linear equality systems in an algebraic field, because of the 4 basic operations (addition, subtraction, multiplication, division).</td><td></td></tr>
<tr><td></td><td>Modulo is not an algebraic field.</td><td></td></tr>
<tr><td>properties</td><td>$\forall a \in F, a \otimes 0 = 0 \otimes a = 0$
$\forall a, b \in F\ and\ a, b \neq 0 \Rightarrow a \otimes b \neq 0$
$a \otimes b = 0\ and\ b \neq 0 \Rightarrow a = 0$
$a \neq 0\ and\ a \otimes b = a \otimes c \Rightarrow b = c$</td><td>$a * 0 = 0 * a = 0$
$1 * 2 \neq 0$
$a * 5 = 0 \rightarrow a = 0$
$3 * a = 3 * b \rightarrow b = c$</td></tr>
<tr><td><b>Finite Fields /</b>
Galois Fields</td><td>$GF(q)$: Field with a finite number q of elements</td><td>$GF(2) = \{0,1\} \rightarrow q = 2$</td></tr>
<tr><td></td><td>Smallest number λ such that $\sum_{i=1}^{\lambda} 1 = 0$
λ is always a prime</td><td>$1 + 1 = 0\ (mod\ 2)$
$\lambda = 2$</td></tr>
<tr><td></td><td>Finite Fields exist only if $q = \lambda^n$ with $n \in \mathbb{N}$ and $\lambda \in \mathbb{P}$
$n = 1 \rightarrow$ Prime Field
$n > 1 \rightarrow$ Extended Field</td><td>$2 = 2^1$ →Prime
$4 = 2^2$ →Extended</td></tr>
<tr><td><b>Prime Field</b>
(Restklassenkörper)</td><td>$GF(p)$ or $\mathbb{Z}_p$
Number of elements p is prime
$F = \{0,1,2, \ldots, p - 1\}$</td><td><table><tr><td>$GF(2)$
$p = 2$
$F = \{0,1\}$</td><td>$GF(3)$
$p = 3$
$F = \{0,1,2\}$</td></tr></table></td></tr>
<tr><td>Addition

/prifiadd(p)</td><td>$a \oplus b = a + b\ mod\ p$</td><td><table><tr><th>$\oplus$</th><th>0</th><th>1</th><th>2</th><th>additive inv</th></tr><tr><td>0</td><td>0</td><td>1</td><td>2</td><td>$-0 = 0$</td></tr><tr><td>1</td><td>1</td><td>2</td><td>0</td><td>$-1 = 2$</td></tr><tr><td>2</td><td>2</td><td>0</td><td>1</td><td>$-2 = 1$</td></tr></table></td></tr>
<tr><td>Multiplication

/prifimul(p)</td><td>$a \otimes b = a * b\ mod\ p$
since p is prime $\gcd(a, p) = 1$ for all $a \in F - \{0\}$ and thus a^{-1} exists</td><td><table><tr><th>$\otimes$</th><th>0</th><th>1</th><th>2</th><th>multipl inv</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0^{-1} not exist</td></tr><tr><td>1</td><td>0</td><td>1</td><td>2</td><td>$1^{-1} = 1$</td></tr><tr><td>2</td><td>0</td><td>2</td><td>1</td><td>$2^{-1} = 2$</td></tr></table></td></tr>
<tr><td><b>Polynomials</b></td><td>$p(x) = a_m * x^m + a_{m-1} * x^{m-1} + \cdots + a_1 * x + a_0, \quad a_i \in F$</td><td>$p(x) = 3x^2 + x - 1$</td></tr>
<tr><td>properties</td><td>if $a_m \neq 0$ then a_m is called the <b>leading coefficient</b>
and m is the <b>degree</b> of $p(x)$</td><td>leading coefficient: $a_m = 3$
degree: $m = 2$</td></tr>
<tr><td></td><td>if $a_m = 1$ then $p(x)$ is called <b>monic</b> (monisch)</td><td>$p(x) = x^2 + x - 1$</td></tr>
<tr><td></td><td>The set of polynomials over the field F is denoted by $F[x]$</td><td></td></tr>
<tr><td>example</td><td>$p(x) = 1.0 * x^2 + 1.0\ over\ \mathbb{R} \rightarrow$ start in the real numbers
$p(x) = 0$ →no solution in $\mathbb{R}$
We define α such that $p(\alpha) = \alpha^2 + 1 = 0$
1. $\alpha \in \mathbb{R} \rightarrow$ the solution of $p(x)$
2. $\alpha^2 + 1 = 0 \Rightarrow \alpha^2 - 1$
E = {a + b ∗ α|a, b, ∈ ℝ} →define extended field
$p(x) = x^2 + 1$ over $GF(2) = \{0,1\}$
$p(x) = (x + 1)(x + 1) \rightarrow$ Behauptung
$p(x) = x^2 + x + x + 1 = x^2 + x\underbrace{(1 + 1)}_{0} + 1 = x^2 +$
1 →Beweis</td><td></td></tr>
<tr><td><b>Irreducible polynomials</b>
factor()</td><td>A polynomial with coefficients in a field F is said to be irreducible over F if it is non-constant and cannot be factored into the product of two or more non-constant polynomials with coefficients in F.</td><td>$x^2 + 1$ is irreducible over $\mathbb{Q}$,
but reducible over $GF(2)$:
$x + 1 = (x + 1)(x + 1)$</td></tr>
</table>

Extended Fields (Erweiterungs-körper)	Start with a polynomial $m(x)$ of degree $n > 1$ that is irreducible over a given field F. The elements of the extended field E are all **polynomials** in $F[x]$ with degree less than n. $E = \{a_{n-1}x^{n-1} + \cdots + a_1 x + a_0, a_i \in F\}$	$m(x) = x^2 + x + 1$ $F = GF(2) = \{0,1\}$
Addition	Coefficients of the polynomials are added in F	
Multiplication	1. Multiply the polynomials (keep in mind that coefficients are in F) 2. Divide my $m(x)$ 3. Take the remainder (degree is always less than n)	
1. irreducible polynomial of degree $n = 2$	a. $m(x) = x^2 = x * x \rightarrow reducible$ b. $m(x) = x^2 + 1 = (x+1)(x+1) \rightarrow reducible$ c. $m(x) = x^2 + x = x(x+1) \rightarrow reducible$ d. $m(x) = x^2 + x + 1 \rightarrow irreducible\ (there\ has\ to\ be\ one)$	
2. extended field	$E = \{0 * x + 0, \quad 0 * x + 1, \quad 1 * x + 0, \quad 1 * x + 1\}$ $E = \{0, 1, x, x + 1\}$	

3. addition table

/extfiadd(q,m)
/extfiadd(2,x^2+x+1)

$\oplus$	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	$x+1+1=x$
x	x	$x+1$	$2x = 0x = 0$	$2x+1=1$
$x+1$	$x+1$	$x+1+1=x$	$2x+1=1$	$2x+2=0$

4. multiply table

/extfimul(q,m)
/extfimul(2,x^2+x+1)

polyRemainder(f,m)

$\otimes$	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

$x^2 \equiv -x - 1 \equiv x + 1$
$x(x+1) = x^2 + x$
$= x + 1 + x = 1$
$(x+1)(x+1)$
$= x^2 + 2x + 1 = x^2 + 1$
$= x + 1 + 1 = x$

remarks	we calculate with module irreducible polynom. $m(x) \equiv 0\ \big(mod\ m(x)\big)$ no signs in $GF(2)$	
Primitive element	The powers of (e.g. x) generate all non-zero elements of E	$x^0 = 1, \quad x^1 = x$ $x^2 = x + 1$
Primitive polynomial	A polynomial $p(x)$ of degree n over $GF(q)$ is primitive if: $p(x)$ is irreducible $p(x) \mid x^{q-1} - 1, \quad q = p^n$ $p(x) \nmid x^k - 1, \quad 0 < k < p^n - 1$	$GF(2^2) \rightarrow q = 4$ $p = 2$ $x^2 + x + 1 \mid x^3 - 1$ $x^2 + x + 1 \nmid x^2 - 1$
	The root α of a primitive polynomial $p(x)$ of degree n over $GF(p)$ is a primitive element of the field $GF(p^n)$	
	$p(x) \mid x^{q-1} - 1 \Leftrightarrow x^{q-1} - 1 = p(x) * k(x), \quad q = p^n$	
	Let α be a root of $p(x)$: $p(\alpha) = 0$	
	We conclude: α is a $q-1$-root of unity	
	However, $\alpha^k \neq 1$ for $0 < k < q - 1$ Otherwise $p(x)$ would divide $x^k - 1$ for $0 < k < q - 1$	

Example

/polgen(p,n,m)
polgen(2,3,x^3+x+1)
$p^n (mod\ n)$

$p(x) = x^3 + x + 1\ (mod\ x^3 + x + 1) = 0, \qquad n = 3, GF(2)$
$x^3 = -x - 1 = x + 1$

Power	Polynomial in α	binary	int
0	0	$(0 \quad 0 \quad 0)$	0
x^0	1	$(0 \quad 0 \quad 1)$	1
x^1	x	$(0 \quad 1 \quad 0)$	2
x^2	x^2	$(1 \quad 0 \quad 0)$	4
x^3	$x + 1$	$(0 \quad 1 \quad 1)$	3
x^4	$x * x^3 = x * (x+1) = x^2 + x$	$(1 \quad 1 \quad 0)$	6
x^5	$x * x^4 = x^3 * x^2 = x^3 + x^2 = x^2 + x + 1$	$(1 \quad 1 \quad 1)$	7
x^6	$x * x^5 = x^3 + x^2 + x = x^2 + 2x + 1$ $= x^2 + 1$	$(1 \quad 0 \quad 1)$	5
x^7	$x * x^6 = x^3 + x = 2x + 1 = 1$		

$x^2(x^2 + x + 1) = x^2 * x^5$

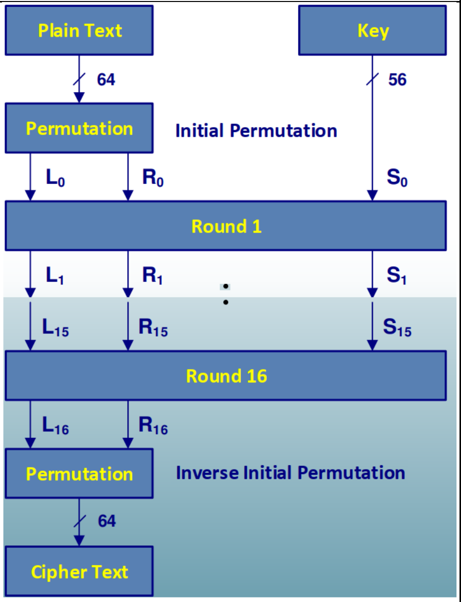
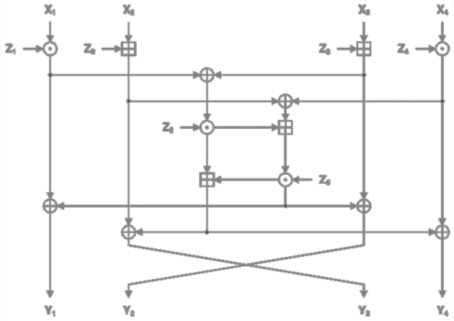
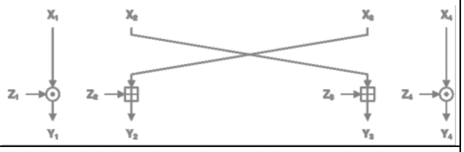
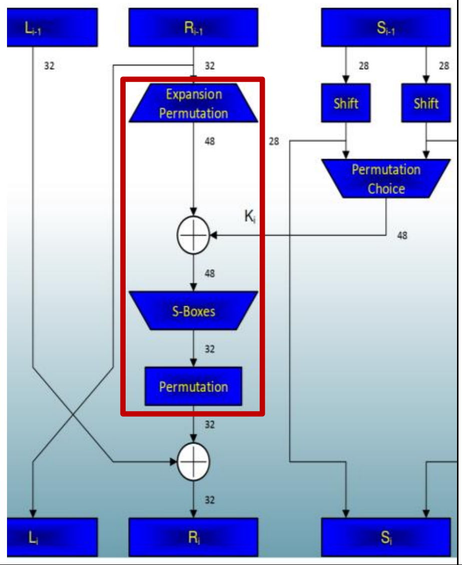
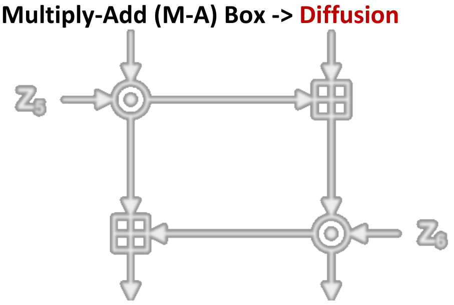
$q = 2^n = 8$
$\alpha = x$

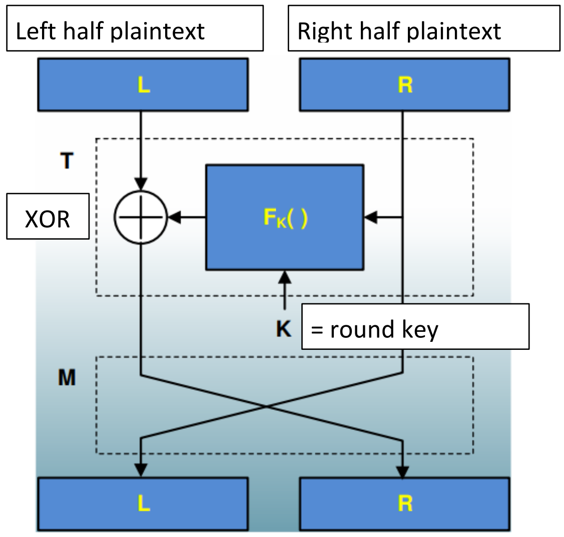
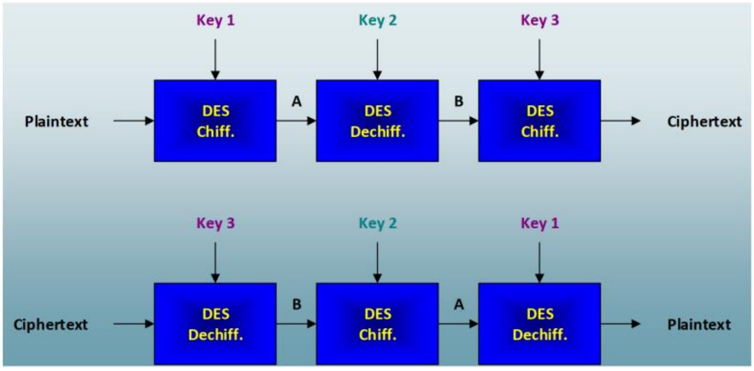
Use the table for multiplying

We use primitive polynomials because x is a generator element.

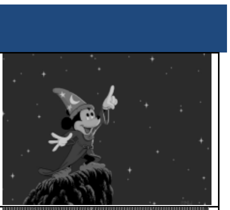
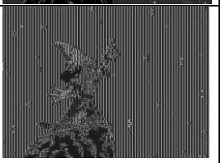
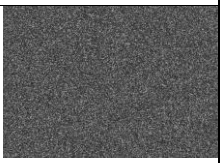
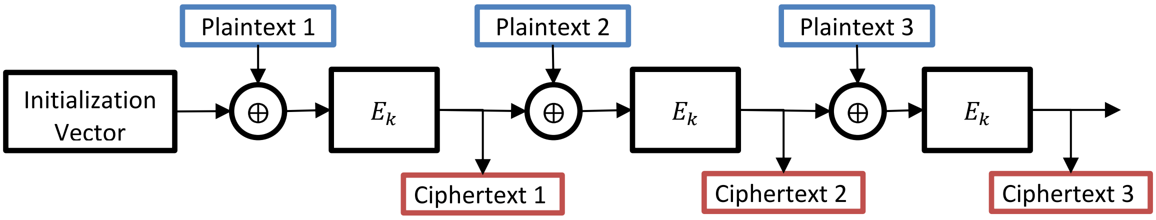
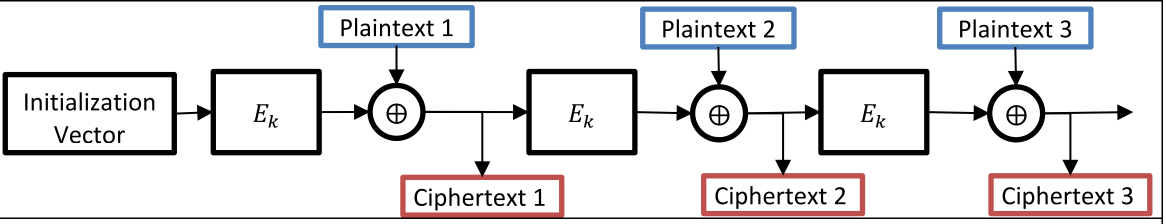
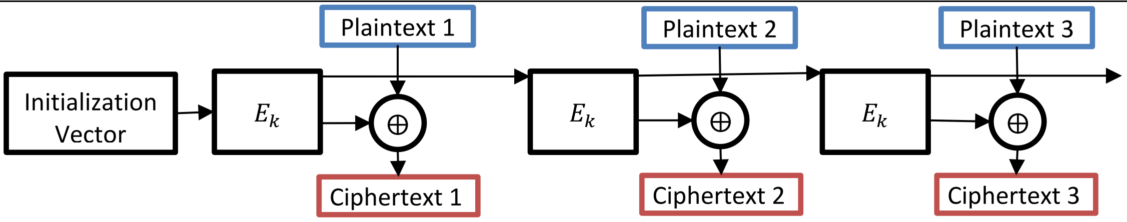
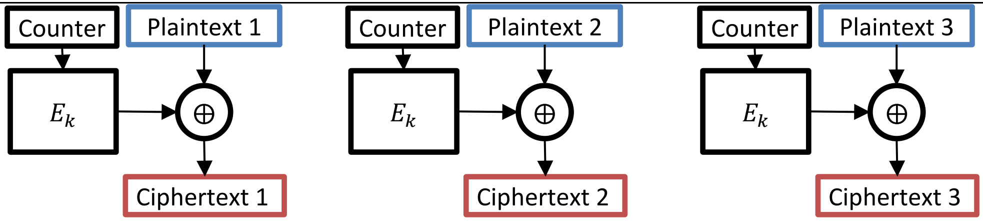
multorder	= first element with value 1 except $x^0 = 1$	

5. Symmetric Encryption Algorithms

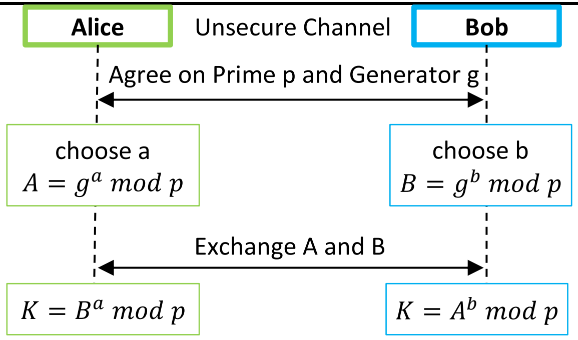
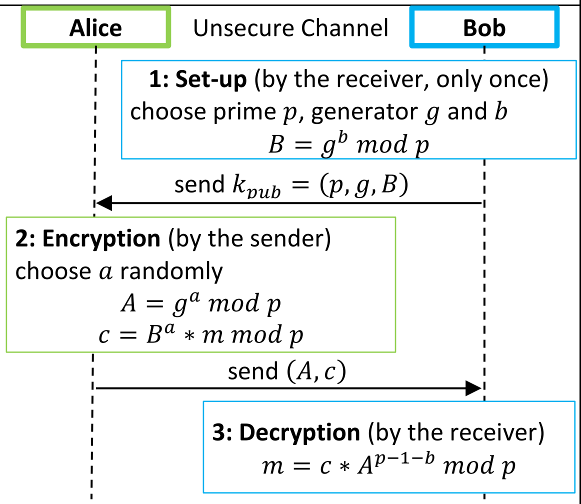
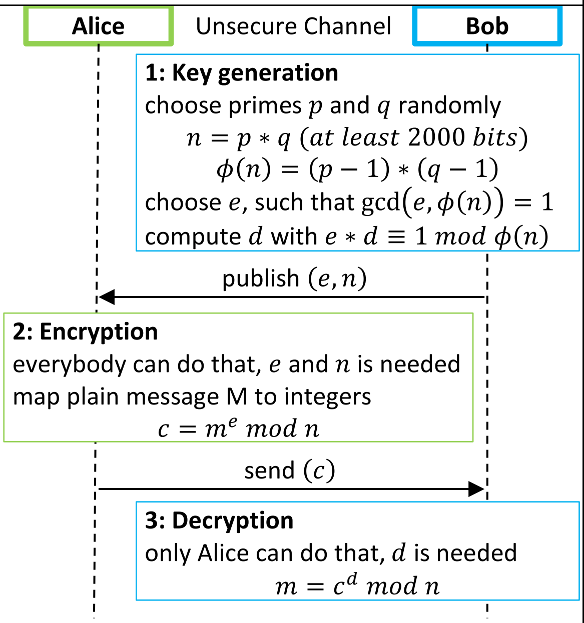
DES - Data Encryption Standard	**AES - Advanced Encryption Standard**	**IDEA - International Data Encryption Algorithm**
Algorithm based on 'Lucifer' Published in 1975 -> IBM und NSA Block cypher, Feistel network		PES - not secure -> Differential Crypto Improved Proposed Encryption Standard 1991
Block size: 64 bits Key size: 56 (+8 parity bits / prüf bits) unsecure, too small -> Brute Force Attack #rounds: 16 (to get a good diffusion)	Block size: 128 bits Key size: 128/192/256 bits secure #rounds = 10/12/14 (key size dependent)	128bit key Key size: 16 bits As much provable security as possible #rounds = 6
 Plain Text Key 64 56 Permutation Initial Permutation L_0 R_0 S_0 Round 1 L_1 R_1 S_1 L_{15} R_{15} S_{15} Round 16 L_{16} R_{16} Permutation Inverse Initial Permutation 64 Cipher Text Permutation -> no crypto significance	Store input bits into state matrix AddRoundKey For each round (except last one) ShiftRows SubBytes AddRoundKey MixColumns ShiftRows SubBytes AddRoundKey Return state matrix	Scalable: Mini-versions with 2/4/8 bit Transparency: no "random-looking" tables or "mysterious" S-Boxes Easy to substitute for DES Fast in Software and Hardware  X_1 X_2 X_3 X_4 Z_1 Z_2 Z_3 Z_4 Z_5 Z_6 Y_1 Y_2 Y_3 Y_4 final round  X_1 X_2 X_3 X_4 Z_1 Z_2 Z_3 Z_4 Y_1 Y_2 Y_3 Y_4
One Round  L_{i-1} R_{i-1} S_{i-1} 32 32 28 28 Expansion Permutation Shift Shift 48 28 Permutation Choice K_i 48 48 S-Boxes 32 Permutation 32 32 L_i R_i S_i	**Store input bits into state matrix** 16*8=128bit input -> insert in state matrix (4x4 with 8bit values) **Add round key (XOR)** early to avoid reversion by the attacker each 8-bit value are interpreted as elements of $GF(2^8)$ with polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$ Key Expansion = Generate a round key from the key **Add Round Key** at the end operations can be inverted -> encryption	**3 incompatible operations -> Confusion** ⊕ Bit-by-bit modulo-two addition (xor) ⊞ Addition modulo 2^{16} ⊙ Multiplication modulo $2^{16} + 1$ of non-zero numbers - $2^{16} + 1$ is prime - 2^{16} is represented by the all-zero string **Multiply-Add (M-A) Box -> Diffusion**  each output depends on every input
1. Expansion Permutation see Expansion Permutation table; 1 -> 2&48; 2 -> 3; 4 -> 5 & 7 Expansion, because several bits of the input will be used twice. **XOR (step 1. and key K)** **S-Boxes (S=Substitution) -> Confusion** 8 Boxes = each 6 input bits, 4 output bits Take first & last bits -> $0 \leq i \leq 3$ -> row Take middle 4 bits $0 \leq j \leq 15$ -> column see S-Boxes table (non-linear) protect against differential analysis **Permutation** (see permutation table) $1 \leftarrow 16, \quad 2 \leftarrow 7, \quad 3 \leftarrow 20$	**SubBytes** = Non-linear byte substitution -> Confusion i) take the multiplicative inverse of $GF(2^8)$, map $\{00\}\ to\ \{00\}$ ii) Affine transformation over $GF(2^8)$ **Shift rows** = copy first row, shift 2nd by 1, 3rd by 2 and last by 3 **Mix Columns** = matrix multiplication of a column (polynomial) with const matrix -> modulo $m(x) = x^4 + 1\ in\ GF(2^8)$, $03 \rightarrow x^2 + 1$ **Add round key** = XOR each column of the state matrix with the corresponding word from the round key	**Encryption/Decryption Similarity** final round causes that the same structure can be used to encrypt and to decrypt. -> Mult-Add-Add-Mult

Structure: Feistel Network	 Left half plaintext Right half plaintext L R T XOR FK( ) K = round key M L R -> Regardless of $F_K(R)$, the same structure can be used to decrypt, by changing left and right at the beginning.	$T(L,R) = (L \oplus F_K(R), R)$ $M(L,R) = (R, L)$
2-DES	2 DES after each other -> with meet-in-the-middle attackable attack from left and right and compare the result $2^{56} + 2^{56} = 2^{57}$ I know what goes in and what comes out	
Triple DES	 Key 1 Key 2 Key 3 Plaintext DES Chiff. A DES Dechiff. B DES Chiff. Ciphertext Key 3 Key 2 Key 1 Ciphertext DES Dechiff. B DES Chiff. A DES Dechiff. Plaintext Key 1,2 and 3 should be independent If all three keys are identic -> single DES	

3b. Block (Cipher) Modes

	What should we do, when we have more than 64/128-bit data to encrypt?	
Electronic Code Book (ECB)	Each plaintext block (of length n) is encrypted individually (with same key) -> not appropriate, except input blocks are random	
drawbacks	Repetitions of plaintext blocks will be perceivable Same plaintext block will always be mapped to same ciphertext block Attacker can change order of ciphertext blocks (or can introduce new blocks)	
Cipher Block Chaining (CBC)	incremental blocks initialization vector (not secret, unpredictable)	
drawbacks	not parallelizable in encryption, parallelizable in decryption Bit errors in a ciphertext block will affect decryption of the actual (50%) and the subsequent block (1bit)	
encryption	 Plaintext 1 Plaintext 2 Plaintext 3 Initialization Vector E_k E_k E_k Ciphertext 1 Ciphertext 2 Ciphertext 3	
Cipher Feedback (CFB)	Feedback of ciphertext blocks into the input of the encryption algorithm Encryption cannot be performed in parallel Bit errors in a ciphertext block will affect decryption of actual and subsequent block	
	 Plaintext 1 Plaintext 2 Plaintext 3 Initialization Vector E_k E_k E_k Ciphertext 1 Ciphertext 2 Ciphertext 3	
Output Feedback (OFB)	Encryption algorithm is used as a pseudo random generator → additive stream cipher IV must be unique for each execution of the mode (but not unpredictable) Needs synchronization between transmitter and receiver No error propagation (1-bit error -> 1-bit in cyphertext)	
CFB and OFB are similar	 Plaintext 1 Plaintext 2 Plaintext 3 Initialization Vector E_k E_k E_k Ciphertext 1 Ciphertext 2 Ciphertext 3	
Counter (CTR)	Encryption/Decryption can be performed in parallel Each counter value should only be used once with the same key → Nonce (Number used only once) No error propagation	
	 Counter Plaintext 1 Counter Plaintext 2 Counter Plaintext 3 E_k E_k E_k Ciphertext 1 Ciphertext 2 Ciphertext 3	
CFB + OFB + CTR	use encryption algorithm for encryption and decryption, but invert order of E_k	

6+7. Asymmetric Cryptography

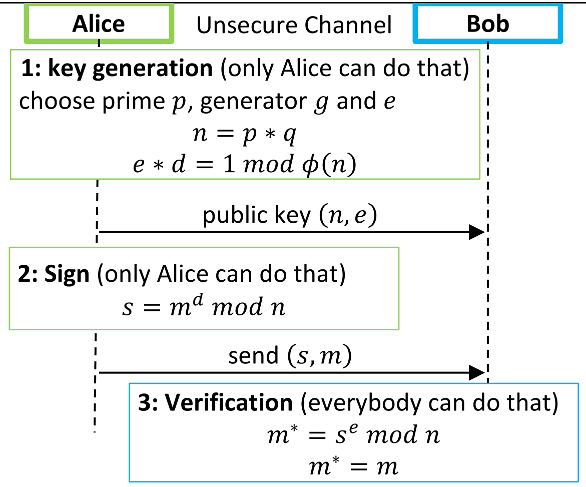
Problem of Symmetric Crypto	Key must be kept secret! Each pair of user's needs another secret key -> number of keys grows → n^2		
Asymmetric Cryptosystem	**Private key**: For every $d \in \mathcal{K}$, it is feasible to compute $e \in \mathcal{K}$. d must be kept secret **Public key**: For (almost) every $e \in \mathcal{K}$, it is computationally infeasible to compute $d \in \mathcal{K}$, such that D_d ist the inverse of E_e. 'e' can be made public.		$D_d\left(E_e(p)\right) = p$
One-Way Function	Easy to compute on every input in polynomial time. Hard to invert, given the image of a random input. Existence of one-way functions is still a conjecture (Vermutung).		
Trapdoor function	One-way function that can be inverted if a secret is known.		
Candidates of one-way functions	**Multiplication and factoring**	Easy: given two primes p and q compute $n = q * p$ Difficult: given $n = p * q$, find the two primes p and q	$7 * 17 = 119$
	Discrete Logarithm Function	Easy: given g, x and p, compute $g^x\ mod\ p$ Difficult: given $g^x\ mod\ p$, g and p, find x	$2^8\ mod\ 5 \equiv 1$
	Elliptic Curves	Easy: given the point P and n, compute $n * P$ Difficult: given $n * P$ and P, compute n	
DHKE **Diffie-Hellman** **Key Exchange**	based on discrete logarithm Not secure against man-in-the-middle attack Works for any cyclic group p has to be prime and big -> avoid brute force g has to be generator for $\mathbb{Z}_p^*$	 **Alice** Unsecure Channel **Bob** Agree on Prime p and Generator g choose a $A = g^a\ mod\ p$ choose b $B = g^b\ mod\ p$ Exchange A and B $K = B^a\ mod\ p$ $K = A^b\ mod\ p$	public: $p = 5, g = 2$ secret: $a = 3, b = 6$ public: $A = 3, B = 4$ secret: $K = 4$
ElGamal Encryption p at least 1024bits $b \in \{2, \dots, p-2\}$ $a \in \{2, \dots, p-2\}$	based on discrete logarithm as difficult to break as DH Works for any cyclic group A: ephemeral/temporary key c: shared key	 **Alice** Unsecure Channel **Bob** **1: Set-up** (by the receiver, only once) choose prime p, generator g and b $B = g^b\ mod\ p$ send $k_{pub} = (p, g, B)$ **2: Encryption** (by the sender) choose a randomly $A = g^a\ mod\ p$ $c = B^a * m\ mod\ p$ send (A, c) **3: Decryption** (by the receiver) $m = c * A^{p-1-b}\ mod\ p$	$p = 11,$ $g = 2, b = 6$ $B = 2^6\ mod\ 11$ $B = 9$ $k_{pub} = (11,2,9)$ $a = 4, m = 7$ $A = 2^4\ mod\ 11 = 5$ $c = 9^4 * 7\ mod\ 11$ $c = 2$ $send(5,2)$ $m = 2 * 5^{11-1-6}$ $mod\ 11 = 7$
RSA (Ronald Rivest, Adi Shamir, Leonard Adleman)	use Euler's Totient Theorem p and q need to be large, independent, large factors n=3072 bits = sym alg 128bits $1 < e < \phi(n)$ $\#e: \phi\left(\phi(n)\right) - 1$ d=private key $0 \leq m < n$ if we knew $\phi(n)$, we could compute d -> egcd factoring n is as hard as computing $\phi(n)$ and the only way to find d (probably). if we now m/4 of first or last digits we can effic. factor n if e is small we use Chinese remainder to compute c	 **Alice** Unsecure Channel **Bob** **1: Key generation** choose primes p and q randomly $n = p * q\ (at\ least\ 2000\ bits)$ $\phi(n) = (p-1) * (q-1)$ choose e, such that $\gcd\left(e, \phi(n)\right) = 1$ compute d with $e * d \equiv 1\ mod\ \phi(n)$ publish (e, n) **2: Encryption** everybody can do that, e and n is needed map plain message M to integers $c = m^e\ mod\ n$ send (c) **3: Decryption** only Alice can do that, d is needed $m = c^d\ mod\ n$	$p = 53, q = 59$ $n = 53 * 59$ $n = 3127$ $\phi(n) = 52 * 58$ $\phi(n) = 3016$ $e = 3$ $d = -1005 = 2011$ $M = "hi"$ $m = 89$ $c = 89^3\ mod\ 3127$ $c = 1394$ $m = 1394^{2011}$ $mod\ 3127$ $m = 89$

Euler's Totient Theorem	$a^{\phi(n)} \equiv 1\ (mod\ n)$ $with\ 1^k = 1: a^{k*\phi(n)} \equiv 1\ (mod\ n)$ $multiply\ a: a^{k*\phi(n)+1} \equiv a\ (mod\ n)$ $e * d = k * \phi(n) + 1$	$3^{\phi(4)} = 3^2 = 9 \equiv 1\ (mod\ 4)$ $3^{3*2} = 729 \equiv 1\ (mod\ 4)$

square and multiply
/sam(a,c,m)
/sam2(a,c)
$a^c\ mod\ m$
 	
/samstep
/sam2step
	compute $a^c\ mod\ m$ for large numbers
	c can be written as binary number $c = b_0 * 2^0 + b_1 * 2^1 + \cdots + b_n * 2^n$
	`re = 1`
	`for i = n..0`
	`   res = res^2 mod m`
	`   if b_i = 1`
	`      res = (res*a) mod m`
	`   end_if`
	`end_for`
		$1234^{5678}\ mod\ 438 = 316$

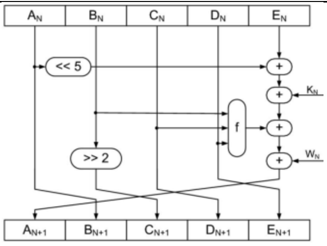
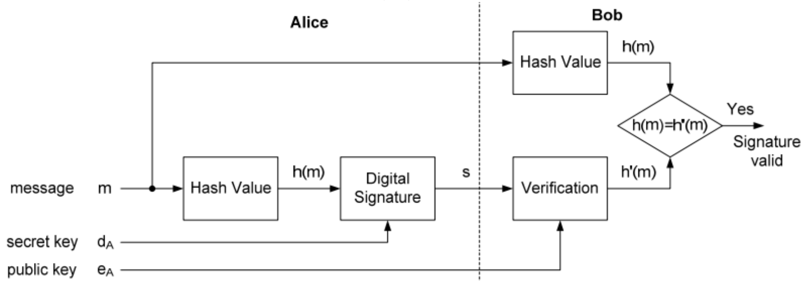
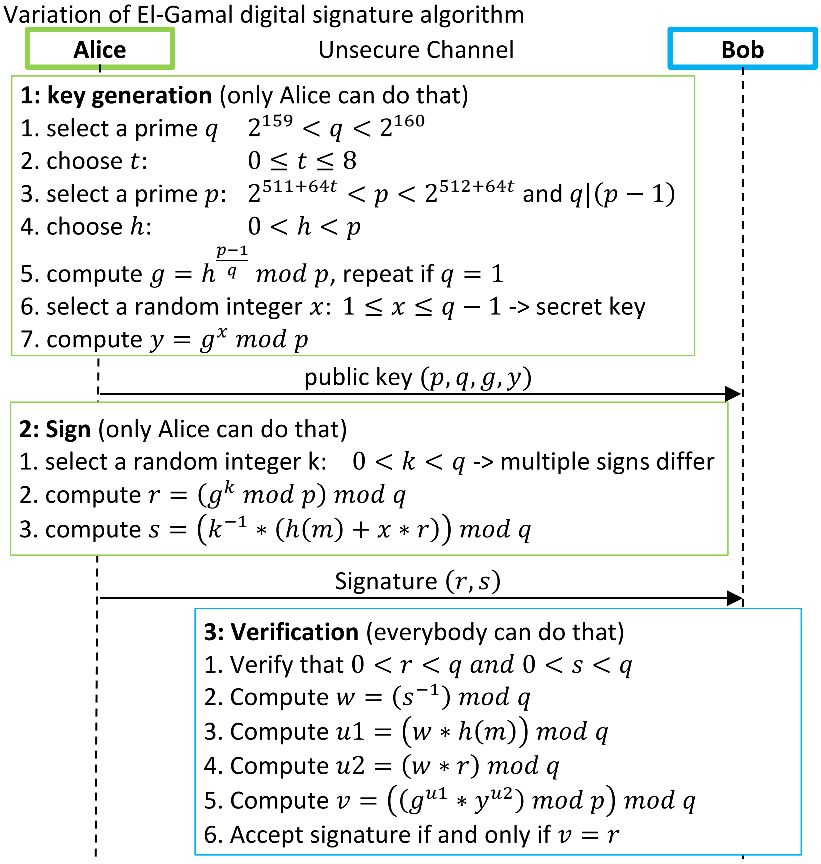
Miller-Rabin Primality Test /isProbPrime(n) /isProbPrimeBase(n,a) composite = not prime	Question: Is n prime or composite? Not the same as factoring! Let n be an integer Suppose there exist integer x and y with $x^2 \equiv y^2 (mod\ n)$, $but\ x \neq \pm y\ (mod\ n)$ Then n is composite and $\gcd(x - y, n)$ gives a nontrivial factor of n.	
	1. Assume that n is odd and write $n - 1 = 2^k * m$	$n = 53$ $\frac{52}{2^1} = 26, \frac{52}{2^2} = \mathbf{13}, \frac{52}{2^3} = 6.5$ $k = 2, m = 13$
	2. Randomly choose a base a with $1 < a < n - 1$	$a = 2$
	3. Compute the starting value $b_0 = a^m\ mod\ n$	$b_0 = 2^{13}\ mod\ 53 = 30$
	4. Compute the sequence $b_0, b_1, \ldots, b_k$ with recursion $b_i = (b_{i-1})^2\ mod\ n$	$b_1 = 30^2\ mod\ 53 = -1$
	5. If n is prime then $b_k \equiv a^{2^k*m} \equiv a^{n-1} \equiv 1\ (mod\ n)$ → Fermat $b_i = 1\ (mod\ n)$ and $b_{i-1} \equiv \pm 1\ (mod\ n)$ otherwise $(b_i)^2 \equiv (b_{i-1})^2 (mod\ n), but\ b_i \neq n_{i-1}$ -> sequence $(b_0, b_1, \ldots, b_k)$ must either start with a 1 or it must somewhere contain a -1	$b_0 = \begin{cases} +1 \rightarrow Prime \\ -1 \rightarrow Prime \\ else \rightarrow continue \end{cases}$ $b_{1..k} = \begin{cases} +1 \rightarrow Composite \\ -1 \rightarrow Prime \\ else \rightarrow continue \end{cases}$
	if n is prime, it will pass the test for any a a composite number passes the test for at most 1/4 or the possible bases a -> it is then called a **strong pseudoprime** for the base a repeating the test M times with randomly chosen values of a, the probability that a composite n passes all the tests is at most $\left(\frac{1}{4}\right)^M$	$M = 50$ $\left(\frac{1}{4}\right)^{50} < 10^{-30}$
Attacks on RSA	In general, if $\gcd(e_A, e_B) = 1$, we can use egcd to find x and y such that: $x * e_A + y * e_B = 1$ and thus: $c_A^x * c_B^x = m^{x*e_A} * m^{y*e_B} = m^{x*e_A} + m^{y*e_B} = m$	
	If $e = 3, m = 128bit, n = 1024$ $m = \sqrt[e]{c} = \sqrt[3]{c}$	

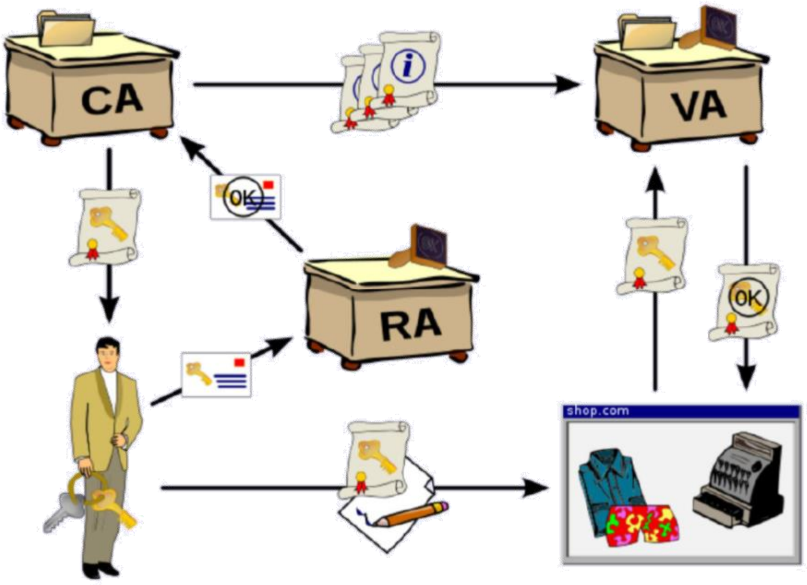
8. Digital Signatures

Definition	The result of a cryptographic transformation of data that, when properly implemented, provides a mechanism for verifying origin authentication, data integrity and signatory non-repudiation.	
origin authentication	Signature can be matched to an entity without a doubt. Nobody can forge (fälschen) the signature.	
data integrity	The signature will no longer be valid if the content of the message is changed after the message has been signed. The signature and the content of the message are unambiguously linked to each other. The signature of a document cannot be used for another document.	
non-repudiation	The signer cannot repudiate (leugnen) his signature.	
Signature generation	The process of using a digital signature algorithm and a private key to generate a digital signature on data. Only **one person** can do that.	private key (d, n)
Signature verification	The process of using a digital signature algorithm and a public key to verify a digital signature on data. **Everybody** can do that.	public key (e, n)
RSA-Signature	 **Alice** Unsecure Channel **Bob** **1: key generation** (only Alice can do that) choose prime p, generator g and e $n = p * q$ $e * d = 1 \; mod \; \phi(n)$ public key (n, e) **2: Sign** (only Alice can do that) $s = m^d \; mod \; n$ send (s, m) **3: Verification** (everybody can do that) $m^* = s^e \; mod \; n$ $m^* = m$	$p = 11, q = 23, e = 3$ $n = 11 * 23 = 253$ $\rightarrow d = 147$ $m = 111$ $s = 111^{47} \; mod \; 253 = 89$ $m^* = 89^3 \; mod \; 253 = 111$ $\rightarrow valid$
Remarks	No encryption, message m can/must be readable and understandable. A long message leads to a long verification.	
Attacks	Authenticity of the public key must be secured (Certificates)	
	No-Message-Attack 1. choose arbitrary number s 2. produce message $m = s^e \; mod \; n$ 3. message m will be accepted as a signed by Alice -> message should contain redundancy. Enforce with redundancy function.	$s = 10$ $m = 10^3 \; mod \; 253 = 241$ $m = m^*$ $m = 123, R(m) = 123'123$
	Multiplicative property of RSA $\begin{matrix} s_1 = m_1^d \; mod \; n \\ s_2 = m_2^d \; mod \; n \end{matrix} \rightarrow s = (m_1 * m_2)^d \; mod \; n$ $m_2 = m * m_1^{-1}$ Alice signs m_1 and m_2, but never m. Attacher can calc $s = s_1 * s_2$ -> message should contain redundancy.	

Hash-Function	A hash function is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length.
	$h: \{0,1\}^* \rightarrow \{0,1\}^n$
	Result = Image
	Input = Preimage	

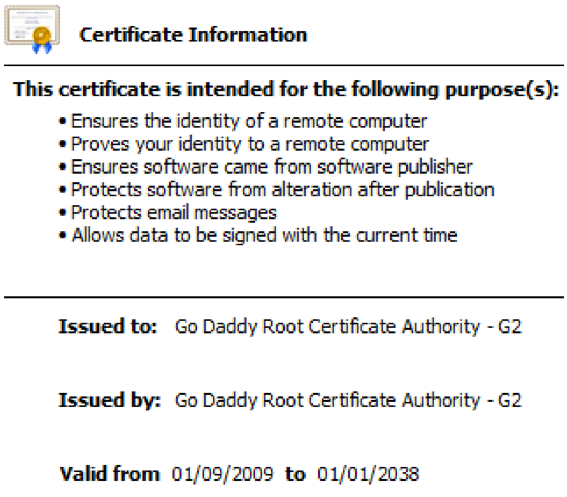
properties	never injective -> set of input value is larger than set of output values collision -> two different input values yield the same output -> very seldom	$000 \rightarrow 100; 101 \rightarrow 100$
properties	preimage resistance -> difficult to find an input string from the output second preimage resistance -> find a second input which results the same collision resistance -> difficult to find two input which results the same	-> weak collision resistance -> strong collision resistance
Examples	SHA (Secure Hash Algorithm) • SHA-1 (160bit) -> no longer considered secure • SHA-2 (224, 256, 384, 512bit) -> secure MD-5 (Message Digest algorithm 5) • MD-5 (128bit) -> no longer considered secure RIPEMD (RACE Integrity Primitives Evaluation Message Digest) • RIPEMD -> no longer considered secure • RIPEMD-160, 320 -> considered secure -> only data integrity	

SHA-1	uses a family of 80 logical functions: $f_0 \dots f_{79}(x, y, z)$ using ∧, ∨, ⨁, ¬ total of 80 32-bit constants K_t $(t = 0..79)$ are defined 1. Fill the message with bits so that the total length is a multiple of 512 2. Split the message into blocks $M^{(i)}(i = 1..N)$ of length 512 3. Use the initial has value $H^{(0)}$ as described in the standard 4. For each message block do the following • Compute W_t • Initialize the five variables $a = H_0^{(i-1)}, b = H_1^{(i-1)}, c = H_2^{(i-1)}, d, e$ • For $t = 0$ *to* 79: Compute T, e, d, c, b, a • Compute $H_0^{(i)}, H_1^{(i)}, H_2^{(i)}, H_3^{(i)}, H_4^{(i)}$	 A_N B_N C_N D_N E_N << 5 K_N f >> 2 W_N A_{N+1} B_{N+1} C_{N+1} D_{N+1} E_{N+1}
Security	Finding collisions is easier than the theoretical limit -> use SHA-2	
Message Authentication Codes "parameterized hash function"	Family of functions with secret parameter k Can be computed efficiently Maps an input x of arbitrary length to a MAC-value $h_k(x)$ of fixed length Authenticity and data integrity	
Digital Signature with Hash function	Signature of an arbitrary long message m. Generation of the signature: $s = h(m)^d \ mod \ n$  Alice Bob Hash Value h(m) h(m)=h'(m) Yes Signature valid message m Hash Value h(m) Digital Signature s Verification h'(m) secret key d_A public key e_A	
properties	Sign only a short hash value instead of a long message m No-message-attack and multiplicative property attack do not work -> because the attacker must generate a message x that gives hash $h(m)$ Exploiting the multiplicative property $(m = m_1 * m_2 \ mod \ n)$ is not possible The signed message m cannot be replaced by another text m^* -> pair m and m^* must be a collision -> rare	
DSA (Digital Signature Algorithm)	Variation of El-Gamal digital signature algorithm  **Alice** Unsecure Channel **Bob** **1: key generation** (only Alice can do that) 1. select a prime q $2^{159} < q < 2^{160}$ 2. choose t: $0 \leq t \leq 8$ 3. select a prime p: $2^{511+64t} < p < 2^{512+64t}$ and $q \mid (p-1)$ 4. choose h: $0 < h < p$ 5. compute $g = h^{\frac{p-1}{q}} \ mod \ p$, repeat if $q = 1$ 6. select a random integer x: $1 \leq x \leq q - 1$ -> secret key 7. compute $y = g^x \ mod \ p$ public key (p, q, g, y) **2: Sign** (only Alice can do that) 1. select a random integer k: $0 < k < q$ -> multiple signs differ 2. compute $r = (g^k \ mod \ p) \ mod \ q$ 3. compute $s = \left(k^{-1} * (h(m) + x * r)\right) mod \ q$ Signature (r, s) **3: Verification** (everybody can do that) 1. Verify that $0 < r < q \ and \ 0 < s < q$ 2. Compute $w = (s^{-1}) \ mod \ q$ 3. Compute $u1 = \left(w * h(m)\right) mod \ q$ 4. Compute $u2 = (w * r) \ mod \ q$ 5. Compute $v = \left((g^{u1} * y^{u2}) \ mod \ p\right) mod \ q$ 6. Accept signature if and only if $v = r$	

Public Key Infrastructure (PKI)	Problem of any asymmetric scheme:
	Authenticity and validity of the public key must be secured
	
		CA = Certification Authority
		RA = Registration Authority
		VA = Validation Authority

Digital Certificate	A set of data that uniquely identifies a key pair and an owner that is authorized to use the key pair. The certificate contains the owner's public key and possibly other information and is digitally signed by a Certification Authority (i.e., a trusted party), thereby binding the public key to the owner. Like a passport.	
Certification Authority	The entity in a Public Key Infrastructure (PKI) that is responsible for issuing certificates and exacting compliance with a PKI policy.	
Digital Signature	The result of a cryptographic transformation of data that, when properly implemented, provides a mechanism for verifying origin authentication, data integrity and signatory non-repudiation.	
Trust Models	• Direct trust (one to another) • Hierarchical trust (root CA -> CA -> people) • Web of trust (Each user can sign a key and define the level of trust that the key's owner can serve as certifier of other keys)	

Example



Certificate Information

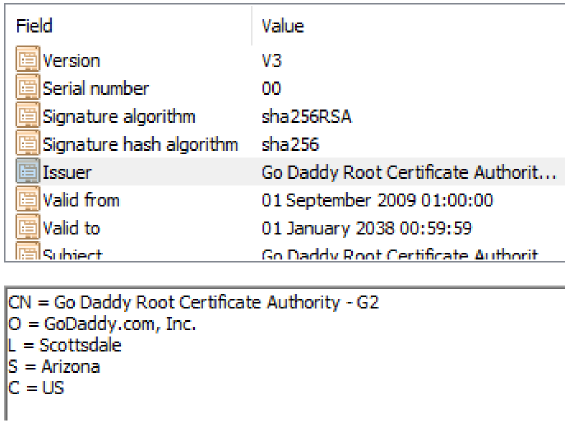
This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer
- Proves your identity to a remote computer
- Ensures software came from software publisher
- Protects software from alteration after publication
- Protects email messages
- Allows data to be signed with the current time

Issued to: Go Daddy Root Certificate Authority - G2

Issued by: Go Daddy Root Certificate Authority - G2

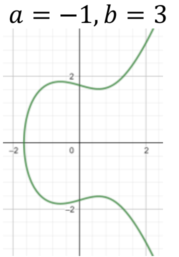
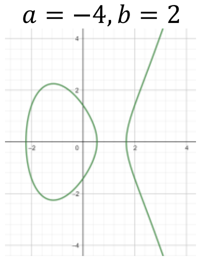
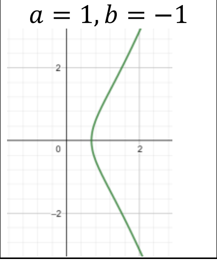
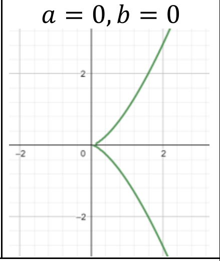
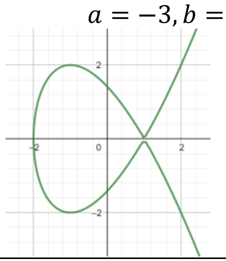
Valid from 01/09/2009 **to** 01/01/2038

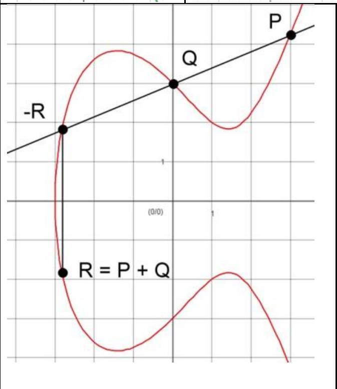


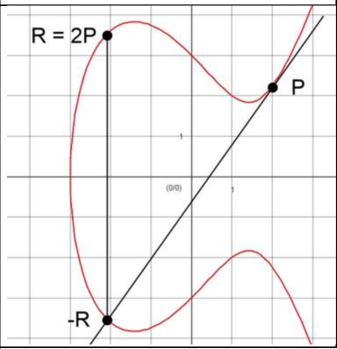
Field	Value
Version	V3
Serial number	00
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	Go Daddy Root Certificate Authorit...
Valid from	01 September 2009 01:00:00
Valid to	01 January 2038 00:59:59
Subject	Go Daddy Root Certificate Authorit

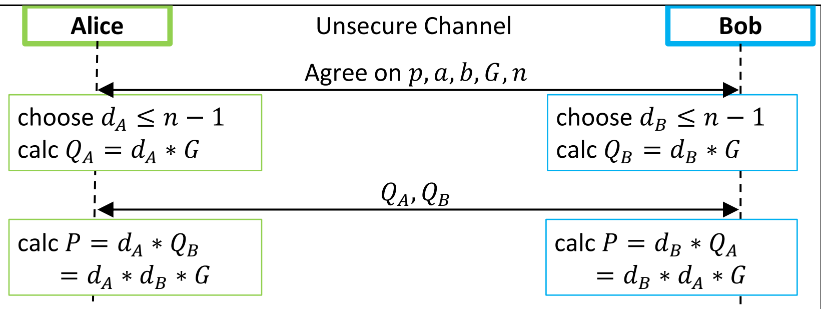
CN = Go Daddy Root Certificate Authority - G2
O = GoDaddy.com, Inc.
L = Scottsdale
S = Arizona
C = US

9. Elliptic Curve

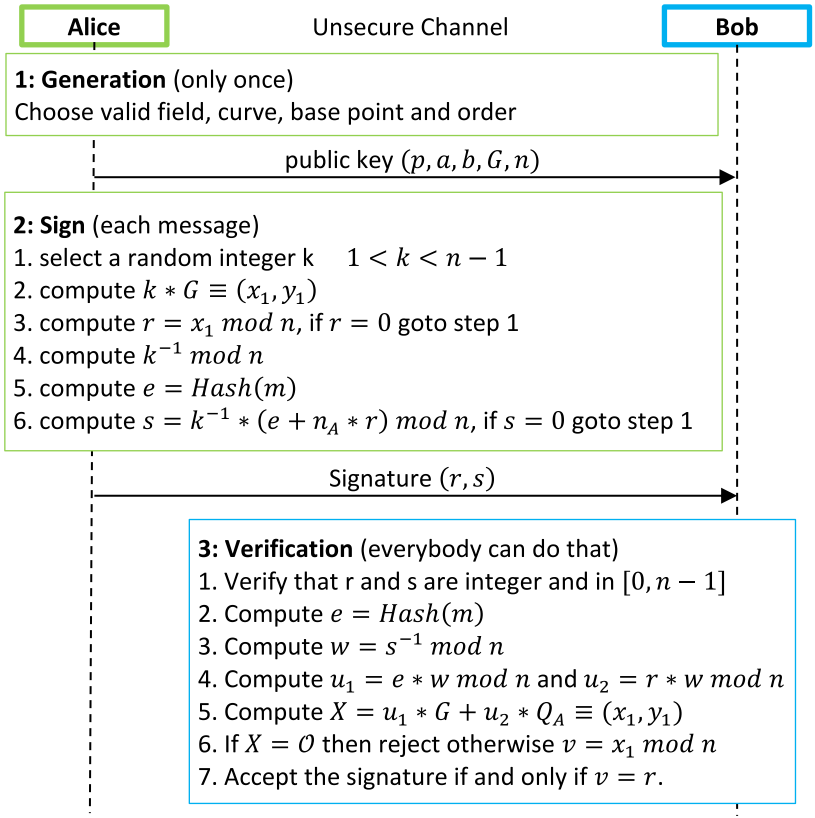
Why	smaller key size -> less space and better performance 128bit AES = 3072bit RSA/DH = 256 ECC	
Definition solve(x^3-ax+b=0,x)	Weierstrass equation combined with a field that has characteristic 2 or 3. $y^2 = x^3 + a * x + b$ The points of the elliptic curve, together with an extra point $\mathcal{O}$, called the point at infinity, can be used to define an additive group. This equation has 3 real or 1 real and 2 complex roots (Nullstellen).	
valid if $4a^3 + 27b^2 \neq 0$ = no multiple roots /validate(ec) /validate(x^3+8x-9) /validmod(ec,p) /validmod(x^3,5)	valid $a = -1, b = 3$  valid $a = -4, b = 2$  valid $a = 1, b = -1$  invalid $a = 0, b = 0$ 	invalid $a = -3, b = 2$ 

Addition
/add(ec,p,q)
/add(x^3-8x+9,{0,3},{2,1})={-1,-4}
/stepadd(ec,p,q)
addmod(ec,p,q,f)
stepaddmod()
	
	Given: $P, Q \in E$
	$P \neq Q, P \neq -Q$
	Construction of $P + Q = R$:
	Draw a line through P and Q.
	Invert intersection $-R$ to yield R
	Special Rules
	point in infinity $\mathcal{O}$ in Y is the neutral elem.
	$\mathcal{O} + \mathcal{O} = \mathcal{O}$
	$P + (-P) = \mathcal{O}$
	$P + \mathcal{O} = \mathcal{O} + P = P$
		Algebraic
		1. Slope
		$m = \begin{cases} \frac{y_P - y_Q}{x_P - x_Q} & P \neq \pm Q \\ \frac{3 * x_P^2 + a}{2 * y_P} & P = Q \\ \infty \to \mathcal{O} & P = -Q \end{cases}$
		2. Interference point
		$x_R = m^2 - x_P - x_Q$
		$y_R = m(x_P - x_R) - y_P$

Multiplication /mult(ec,p,n) /mult(x^3-8x+9,{2,1},3)={-1,4} /stepmult(ec,p,n /multmod(ec,p,n,f) /stepmultmod(ec,p,n,f)	 **Given:** $P \in E$ **Construction of** $2 * \boldsymbol{P}$: Draw the tangent line through P Invert intersection $-R$ to yield $R = 2P$	**Algebraic** see above $P = Q$ btw: $y^2(x) = x^3 + ax + b \quad \frac{\delta}{\delta x}$ $2 * y(x) * y'(x) = 3x^2 + a$
Finite Group ...mod()	because we only used the 4 basic operations, these equations are valid in each field. egcd for division!	
Order of a point /order(ec,p,f)	Smallest non-negative integer, for which $n * G = \mathcal{O}$ Should be as high as possible for cryptography cofactor: $h = \frac{\#E(\mathbb{F}_p)}{n} \in \mathbb{N}$ -> Order of points always divides total number of points	$order(x^3 + x + 1, \{0,1\}, 7) = 5$
Number of Points /numofpoints(p)	→ see Theorem of Hasse $p + 1 - 2 * \sqrt{p} \leq \#E(\mathbb{F}_p) \leq p + 1 + 2 * \sqrt{p}$ for large p $\#E(\mathbb{F}_p) \approx p$	$\#p(7) = 3..13$ $8 - 2\sqrt{7} \leq E(\mathbb{F}_p) \leq 8 + 2 + \sqrt{7}$ $5.17 \leq E(\mathbb{F}_p) \leq 10.82$ $6 \leq E(\mathbb{F}_p) \leq 10$

EC-DHKE (Elliptic curve Diffie-Hellman) $p \in \mathbb{P} \rightarrow field$ $a, b \in \mathbb{F}_p \rightarrow curve$ $G \in E(\mathbb{F}_p) \rightarrow base$ $n \in \mathbb{P} \rightarrow order$	 **Alice** Unsecure Channel **Bob** Agree on p, a, b, G, n choose $d_A \leq n-1$ calc $Q_A = d_A * G$ choose $d_B \leq n-1$ calc $Q_B = d_B * G$ Q_A, Q_B calc $P = d_A * Q_B$ $= d_A * d_B * G$ calc $P = d_B * Q_A$ $= d_B * d_A * G$ If $P \neq \mathcal{O}$ then Alice and Bob take the x-coordinate x_p as the shared key.	$p =$
ECDLP (Elliptic Curve Discrete Logarithm Problem)	Given an elliptic curve $E(\mathbb{F}_p)$ over a finite field $\mathbb{F}_p$, a point G on that curve and another point Q you know to be an integer multiple of G. The problem is to find the integer n such that $n * G = Q$. -> is believed to be hard to solve, even with today's computational power. -> log, because the question is how many times the operation is applied	

Attacks	Attacker knows P, a, b, G, Q_A, Q_B
	Attacker does not know d_A, d_B
	from n, G calculating $n * G$ is easy
	from $n * G$ and G determining n is hard → $ECDLP$	

ECDSA (Elliptic curve Digital Signature Algorithm)	similar as El-Gamal Algorithm, but in a different field.  **Alice** Unsecure Channel **Bob** **1: Generation** (only once) Choose valid field, curve, base point and order public key (p, a, b, G, n) **2: Sign** (each message) 1. select a random integer k $1 < k < n-1$ 2. compute $k * G \equiv (x_1, y_1)$ 3. compute $r = x_1 \bmod n$, if $r = 0$ goto step 1 4. compute $k^{-1} \bmod n$ 5. compute $e = Hash(m)$ 6. compute $s = k^{-1} * (e + n_A * r) \bmod n$, if $s = 0$ goto step 1 Signature (r, s) **3: Verification** (everybody can do that) 1. Verify that r and s are integer and in $[0, n-1]$ 2. Compute $e = Hash(m)$ 3. Compute $w = s^{-1} \bmod n$ 4. Compute $u_1 = e * w \bmod n$ and $u_2 = r * w \bmod n$ 5. Compute $X = u_1 * G + u_2 * Q_A \equiv (x_1, y_1)$ 6. If $X = \mathcal{O}$ then reject otherwise $v = x_1 \bmod n$ 7. Accept the signature if and only if $v = r$.	

10. Quantum Cryptography

Two-hole wall Experiment	Electrons are particles. The probability of arrival behind a two-hole-wall is distributed like the intensity of a wave. We observe interference. -> It is not true that a single electron flies either though hole 1 or 2.	
Observation	If we observe the electron it passes hole 1 or 2.	
Notation	Probability for the transition from a start state Ψ_1 to an end state Ψ_2 $\langle\Psi_2\vert\Psi_1\rangle$	
Photon	Can be polarized vertical ($\updownarrow$) or horizontal ($\leftrightarrow$) or with an arbitrary angle ϕ with respect to x-axis. $\cos(\phi) * \vert\leftrightarrow\rangle + \sin(\phi) * \vert\updownarrow\rangle$	

SKIP

11+12. Linear Block Codes

Error Control Coding

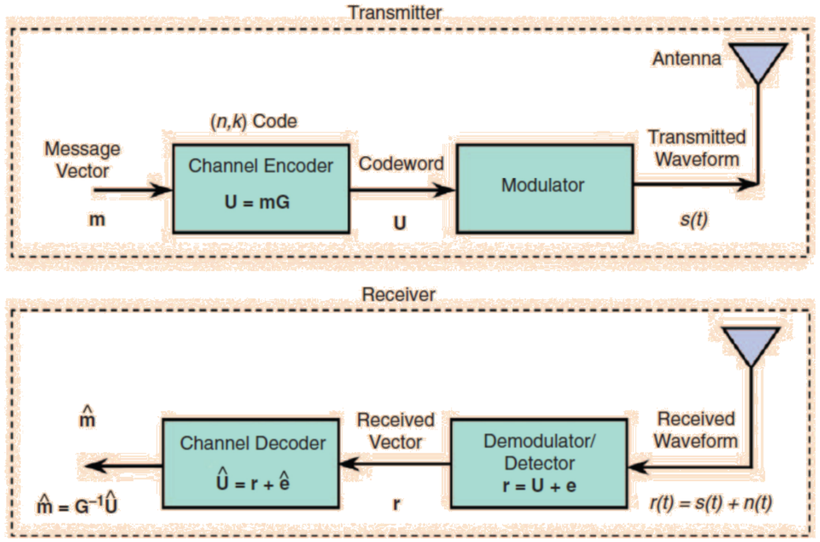
Sizes: capacity C, entropy per second H
Claude Shannon: Error induces by a noisy channel can be reduced to any desired level (if $H \leq C$)

Channel Coding

Data transformations that are used for improving a system's error performance.
Encoder: add redundant information to the transmitted data (code word) - no memory
Decoder: check whether the received data is still exhibit the prearranged structure/regularity
-> Error Detection and Error Correction

(n, k)-Block Code

$\hat{} = approx.$



Message block
$\boldsymbol{m} = (m_1, \dots, m_k)$
k information symbols of a finite field $GF(2^x)$
Code word
$\boldsymbol{u} = (u_1, \dots, u_n)$
$\boldsymbol{u} = (m_1, \dots, m_k, p)$
n code symbols
Demodulator
observe the signal $r(t)$ and produces received vector
$\boldsymbol{r} = (r_1, \dots, r_n) = \boldsymbol{u} \oplus \boldsymbol{e}$
Hard decision: 0 / 1
Soft decision: might be 0 / 1
Error pattern
$e = (e_1, \dots, e_n)$
n(t) = Rauschen

Parity codes

Even parity code	$p = m_1 \oplus \dots \oplus m_k$	$m = 1101 \rightarrow p = 1$
Two-dimensional parity code	$p_1 = m_1 \oplus m_2 \oplus m_3 \oplus m_4$ $p_2 = m_5 \oplus m_6 \oplus m_7 \oplus m_8$ $p_3 = m_1 \oplus m_5, p_4 = m_2 \oplus m_6$ $p_5 = m_3 \oplus m_7, p_6 = m_4 \oplus m_8$	$m = 11010001$ $p_1 = 1, p_2 = 1$ $p_3 = 1, p_4 = 1$ $p_5 = 0, p_6 = 0$

Binary Linear Block Codes

A binary block code with 2^k code words of length n is called linear (n, k) code, if and only if its 2^k code words form a k-dimensional subspace of the vector space of the n-tuples over the field $GF(2)$.

=> the sum of any two code words is a code word. Linear combination!
=> the zero-code word is always a codeword in a linear block, $v + v = 0$

message: $m_i \in GF(2) \rightarrow 2^k$ code words
code word: $u_i \in GF(2) \rightarrow 2^k$ binary vectors of length n

(6,3) block code -> $\{0,1\}^6$

Message 2^3	Codeword
000	000000
100	110100
010	011010
110	101110
001	101001
101	011101
011	110011
111	000111

Vector Space

$\mathbb{F}$: field of Scalars
$\mathbb{V}$: vector space
Two operations:
- Vector addition: $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{V} \Rightarrow \boldsymbol{u} + \boldsymbol{v} \in \mathbb{V}$
- Scalar multiplication $\boldsymbol{u} \in \mathbb{V}, k \in \mathbb{F} \Rightarrow k * \boldsymbol{u} \in \mathbb{V}$
- 10 Axioms

$\mathbb{F} = GF(2)$
$V = \{(v_1 \dots v_n): v_i \in GF(2)\}$

Subspace

$\mathbb{W} \subseteq \mathbb{V}$: Subset. If $\mathbb{W}$ is a vector space itself, it is called a subspace of $\mathbb{V}$.

Linear combination

$$a_1 * \boldsymbol{v_1} + \dots + a_k * \boldsymbol{v_k}$$

Linear independent

$$a_1 * \boldsymbol{v_1} + \dots + a_k * \boldsymbol{v_k} = \boldsymbol{0}$$

Generator Matrix

It is possible to find k linearly independent code words $\boldsymbol{g_1} \dots \boldsymbol{g_k}$ such that every code word $\boldsymbol{u}$ is a linear combination of these k code words.

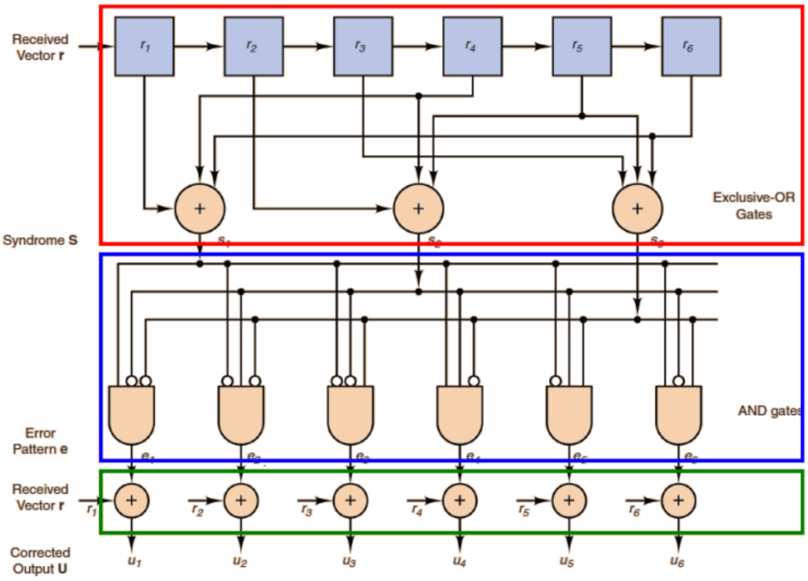
$$\boldsymbol{u} \in C \leftrightarrow \boldsymbol{u} = m_1\boldsymbol{g_1} + m_2\boldsymbol{g_2} + \dots + m_k\boldsymbol{g_k} = \boldsymbol{m} * \boldsymbol{G}$$
$$\boldsymbol{u} = \boldsymbol{m} * \boldsymbol{G}$$

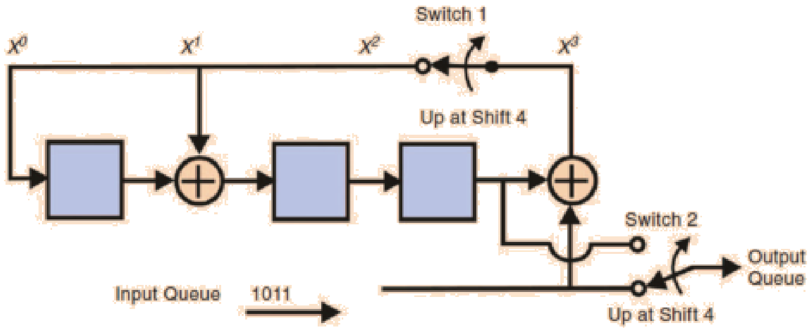
$$vectors\ g_1, g_2, \dots, g_k \in C, are\ linear\ independent$$
$$m_1, m_2, \dots, m_3: Skalare\ (\{0,1\})$$

A generation matrix is **systematic**, if it contains the identity matrix.
Where it is, doesn't matter. $G = [P|I]\ or [I|P]$

$$G = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}$$

$$G = \left[\underbrace{\begin{matrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{matrix}}_{P} \quad \underbrace{\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}}_{I_k} \right]$$

Hamming weight	$w(\boldsymbol{u}) = number\ of\ nonzero\ elements\ in\ \boldsymbol{u}$	$\boldsymbol{u} = (110100) \rightarrow w(\boldsymbol{u}) = 3$
Hamming distance	$d(\boldsymbol{u},\boldsymbol{v}) = number\ of\ bit\ positions\ in\ which\ \boldsymbol{u}\ and\ \boldsymbol{v}\ differ$	$\boldsymbol{v} = (101001)$ $\rightarrow d(\boldsymbol{u},\boldsymbol{v}) = 4$
properties	$d(\boldsymbol{u},\boldsymbol{v}) = w(\boldsymbol{u}+\boldsymbol{v})$ $d(\boldsymbol{u},\boldsymbol{0}) = w(\boldsymbol{u})$ $d(\boldsymbol{u},\boldsymbol{v}) \le d(\boldsymbol{u},\boldsymbol{w}) + d(\boldsymbol{w},\boldsymbol{v})$	$4 = w(011101) = 4$ $3 = 3$ $4 \le 4 + 4$
min properties	Minimum Hamming weight of a code C $w_{\min}\ (C) = \min\{w(\boldsymbol{u}): \boldsymbol{u} \in C, \boldsymbol{u} \ne \boldsymbol{0}\}$ Minimum Hamming distance of a code C $d_{\min}\ (C) = \min\{d(\boldsymbol{u},\boldsymbol{v}): \boldsymbol{u},\boldsymbol{v} \in C, \boldsymbol{u} \ne \boldsymbol{v}\}$	
Theorem	The minimum distance of a linear code block code is equal to the minimum weight of its nonzero code words.	
Decoding	We assume that no bits got lost. $r = (r_1 \dots r_n) = c + e$ Find the code word that differs the least from the received vector. Error detection $\epsilon = d_{\min} - 1$ | Error correction $t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$	
Parity Check Matrix H	A linear (n, k) block code is defined by n − k parity check equations. These equations can be written in matrix form: $\mathbf{u} * \mathbf{H}^{\mathrm{T}} = \mathbf{0}$ Dimensions of H: $(n-k) \times n$ Any vector u that satisfies this equation is a valid code word. $\boldsymbol{u} \in C \Leftrightarrow \mathbf{u} * \mathbf{H}^{\mathrm{T}} = \mathbf{0}$ $u * h_1^T = 0, u * h_2^T = 0, \dots (= dot\ products)$	$\boldsymbol{H} = \begin{bmatrix} \underbrace{\begin{matrix}1&0&0\\0&1&0\\0&0&1\end{matrix}}_{I_k} & \underbrace{\begin{matrix}1&0&1\\1&1&0\\0&1&1\end{matrix}}_{P^T}\end{bmatrix}$ $u_1 + u_4 + u_6 = 0$ $u_2 + u_4 + u_5 = 0$ $u_3 + u_5 + u_6 = 0$
G and H	The rows of G are orthogonal to the rows of H. $\boldsymbol{G} * \boldsymbol{H}^T = \mathbf{0}$ $\boldsymbol{G} = [\boldsymbol{P} \| \boldsymbol{I}_{\boldsymbol{k}\times\boldsymbol{k}}] \Rightarrow \boldsymbol{H} = [\boldsymbol{I}_{(\boldsymbol{n}-\boldsymbol{k})\times(\boldsymbol{n}-\boldsymbol{k})} \| \boldsymbol{P}^T]$	
Syndrome Testing	Is the received vector **r** a valid code word? $\boldsymbol{s} = \boldsymbol{r} * \boldsymbol{H}^T$ 1. case: $\boldsymbol{s} = 0 \Rightarrow \boldsymbol{r} \in C\ (r\ ist\ a\ valid\ code\ word), but\ is\ it\ correct?$ a) $\boldsymbol{r} = u$, error free b) $\boldsymbol{r} \ne u$, not the sent one, not recognizable error -> $e \in C \setminus \{0\}$ 2. case: $\boldsymbol{s} \ne 0 \Rightarrow Fehler!$ The syndrome only depends on the error pattern **e** $\boldsymbol{s} = \boldsymbol{r} * H^T = (\boldsymbol{u} + \boldsymbol{e}) * \boldsymbol{H}^T = \underbrace{c * H^T}_{=0} + e * H^T$ $\underbrace{\boldsymbol{s}}_{\substack{n-k\ check\ equations \\ 2^{n-k}\ \text{syndrome vectors}}} = \boldsymbol{e} * \underbrace{\boldsymbol{H}^T}_{\substack{n\ code\ bits \\ 2^n\ \text{error patterns}}}$ For any value of the syndrome vector, there is more than one possible error pattern -> We just pick the most likely.	
Compute Syndrome **Determine most likely error pattern** **Correct received vector**	 Received Vector r r1 r2 r3 r4 r5 r6 Exclusive-OR Gates Syndrome S s1 s2 s3 AND gates Error Pattern e e1 e2 e3 e4 e5 e6 Received Vector r Corrected Output U u1 u2 u3 u4 u5 u6	$\boldsymbol{s} = \boldsymbol{r} * \boldsymbol{H}^{\boldsymbol{T}}$ search s in H -> results in error e calc all s of each error - if unique -> correctable - if not -> not correctable $\boldsymbol{u} = \boldsymbol{r} + \boldsymbol{e}$

<table>
<tr><td><b>Cyclic Codes</b></td><td>Linear block code. Every cyclic shift of a code word is a code word.
Using polynomials to represent a binary vector.
$$\boldsymbol{u} = u_0, u_1, \dots, u_{n-1}$$
$$\boldsymbol{u}(x) = u_0 + u_1 * x + \cdots + u_{n-1} * x^{n-1}$$</td><td rowspan="4">$$g(x) = x^3 + x + 1, n = 7$$
$$n - k = 3, k = 4$$
$$m = (1010)$$
$$m(x) = 1 + 0x + 1x^2 + 0x^3$$
$$= 1 + x^2$$
$$u(x) = \underbrace{(1 + x^2)}_{Grad<k} * \underbrace{(x^3 + x + 1)}_{Grad=n-k}$$
$$= \underbrace{x^3 + x + 1 + x^5 + x^3 + x^2}_{Grad<n}$$
$$= 1 + x + x^2 + x^5$$
$$u = (1110010)$$
-> not systematic</td></tr>
<tr><td>generator polynomial</td><td>In a (n, k)-cyclic code exists exactly one code polynomial of degree $n - k$:
$$\boldsymbol{g}(x) = 1 + g_1 * x + \cdots + g_{n-k-1} * k^{n-k-1} + x^{n-k}$$
$$\boldsymbol{u} \in C \Leftrightarrow \boldsymbol{u}(x) = \boldsymbol{m}(x) * \boldsymbol{g}(x)$$</td></tr>
<tr><td>must be <b>systematic</b></td><td>$$\boldsymbol{u} = (p_0, \dots, p_{n-k-1}, m_0, \dots, m_{k-1})$$
$$\boldsymbol{u}(x) = \boldsymbol{p}(x) + x^{n-k} * m(x)$$</td></tr>
<tr><td>must be a <b>code word</b></td><td>$$\boldsymbol{u}(x) = \boldsymbol{p}(x) + x^{n-k} * m(x) = \boldsymbol{q}(x)\boldsymbol{g}(x)$$
$$\rightarrow \boldsymbol{p}(x) = x^{n-k} * m(x)\ mod\ \boldsymbol{g}(x)$$</td></tr>
<tr><td>Encoding with linear shift register → division</td><td>

<table>
<tr><td>Cycle 0 to k-1</td><td>switch 2 is down -> message directly fed to the output
at the end, cells contain $\boldsymbol{p}(x) = x^{n-k} * \boldsymbol{m}(x)\ mod\ \boldsymbol{g}(x)$</td></tr>
<tr><td>Cycle k to n-1</td><td>switch 2 is up
the content of the cells will be shifted to the output</td></tr>
</table></td><td>$$g(x) = 1 + x + x^3$$
$$m(x) = (1011)$$
<table>
<tr><th>cycle</th><th>cell</th><th>back</th><th>in</th><th>out</th></tr>
<tr><td>0</td><td>000</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>110</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>2</td><td>101</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>3</td><td>100</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>4</td><td>100</td><td>-</td><td>-</td><td>0</td></tr>
<tr><td>5</td><td>010</td><td>-</td><td>-</td><td>0</td></tr>
<tr><td>6</td><td>001</td><td>-</td><td>-</td><td>1</td></tr>
<tr><td>7</td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td>Error correction</td><td>We compute the syndrom polynomial $\boldsymbol{s}(x) = \boldsymbol{r}(x)\ mod\ \boldsymbol{g}(x)$
$$\boldsymbol{s}(x) = 0 \rightarrow valid$$
$$\boldsymbol{s}(x) \neq 0 \rightarrow error$$
$$\boldsymbol{s}(x) = \boldsymbol{e}(x)\ mod\ \boldsymbol{g}(x)$$</td><td></td></tr>
</table>

13. Hamming, BCH and RS Codes

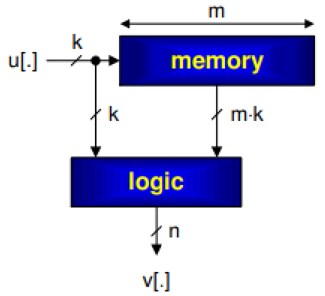
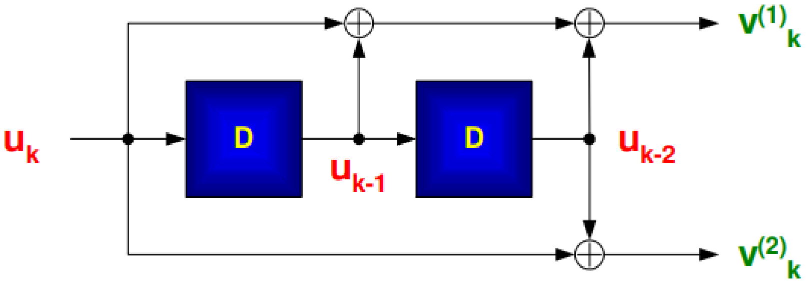
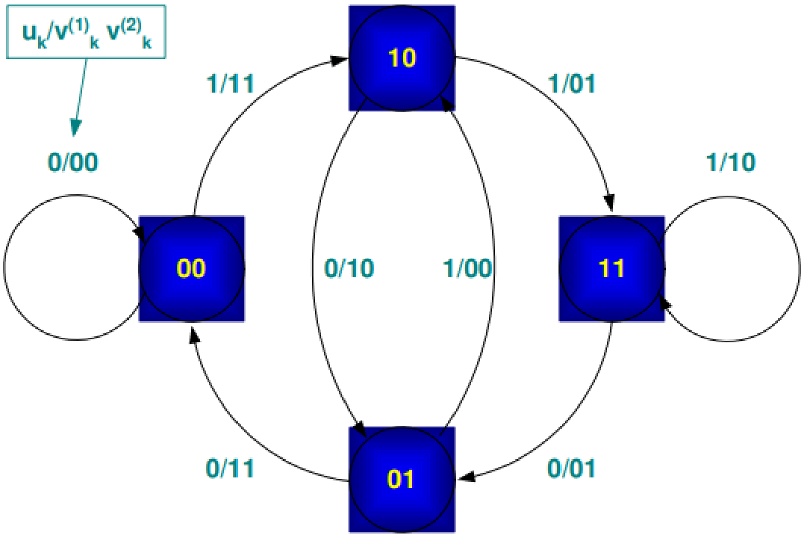
<table>
<tr><td rowspan="3"><b>Hamming Code</b></td><td>A Hamming Code is a linear block code
there elements are binary vectors of b without the zero-vector.
all single bit errors are correctable, nothing else. $q = 1$</td><td></td></tr>
<tr><td colspan="2">$p(x)$:primitive $\{0, \alpha^0, \alpha^1, \dots, a^{2^m-2}\}$, not factorizable polynomial in $GF(2)$
α: primitive element $p(x) \rightarrow p(\alpha) = 0$ -> checksum condition of hamming code -> to generate elements
u: Codeword $= (u_0, \dots, u_{n-1}), u_i \in GF(2)$</td></tr>
<tr><td>Every code word consists of $n = 2^m - 1$ binary digits
Checksum$\sum_{j=0}^{n-1} u_j * a_j = (u_0 \quad \cdots \quad u_{n-1}) \begin{bmatrix} \alpha^0 \\ \dots \\ \alpha^{n-1} \end{bmatrix} = \boldsymbol{u} * \boldsymbol{H}^T = 0$
In <b>H</b> are all elements of $GF(2^m)$ except 0. $\boldsymbol{H} = [a^0\ a^1 \dots a^{n-1}]$</td><td>$m = 2 \rightarrow n = 3$
$m(x) = x^2 + x + 1$
$GF(2) = \{0, 1, \alpha, \alpha + 1\}$
$u_0(0\ 1) + u_1(1\ 0) + u_2(1\ 1) = (0\ 0)$
$(u_0 \quad u_1 \quad u_2) \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$</td></tr>
<tr><td><b>Cyclic Hamming Code</b></td><td>The primitive polynomial $m(x)$ of degree m is the generator polynomial $g(x)$ of the cyclic $(2^m - 1, 2^m - m - 1)$-Hamming code
$u(a^2) = 0$ for every code polynomial since in $GF(2)$: $\sum_i a_i^2 = (\sum_i a_i)^2$</td><td></td></tr>
<tr><td><b>BCH Codes</b>
Bose-Chaudhuri-Hocquenghem</td><td>Choose a field $GF(2^m)$ for some positive integer m.
Let α be a primitive element of this field.
A code word consists of $n = 2^m - 1$ binary digits
$u = (u_0 \dots u_{n-1}), u_i \in \{0,1\} \rightarrow binary$
This code can correct t errors if $r \geq 2t - 1$
<table><tr><td>Checksum
$\sum_{i=0}^{n-1} u_i * a^{i*q} = 0$</td><td>$u_0 a^0 + u_1 a^1 + \cdots + u_{n-1} * a^{n-1} = 0$
$u_0 a^{2*0} + u_1 a^{2*1} + \cdots + u_{n-1} * a^{2*(n-1)} = 0$
$u_0 a^{3*0} + u_1 a^{3*1} + \cdots + u_{n-1} * a^{3*(n-1)} = 0$
...</td></tr></table>Each binary vector which fulfils the check equation for $q = 1,3,5,7$ is valid. 2,4,6, ... are redundant.
$H = \begin{bmatrix} 1 & a^1 & a^2 & \cdots & a^{(n-1)} \\ 1 & a^3 & a^{3^2} & \cdots & a^{3^{(n-1)}} \\ 1 & a^5 & a^{5^2} & \cdots & a^{5^{(n-1)}} \\ \vdots & & & & \vdots \\ 1 & a^r & a^{r^2} & \cdots & a^{r^{(n-1)}} \end{bmatrix}$
$\#rows = 2 * m$</td><td>$m = 4$
$a^{15} = 1$
$n = 2^m - 1 = 15$

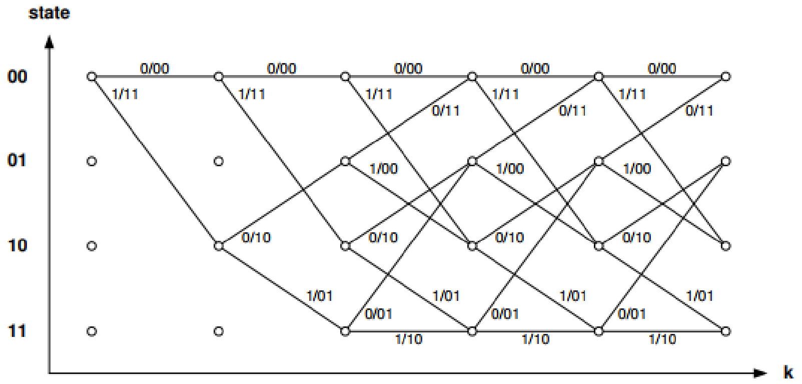
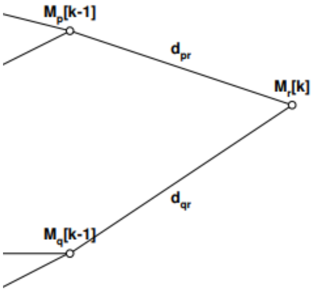
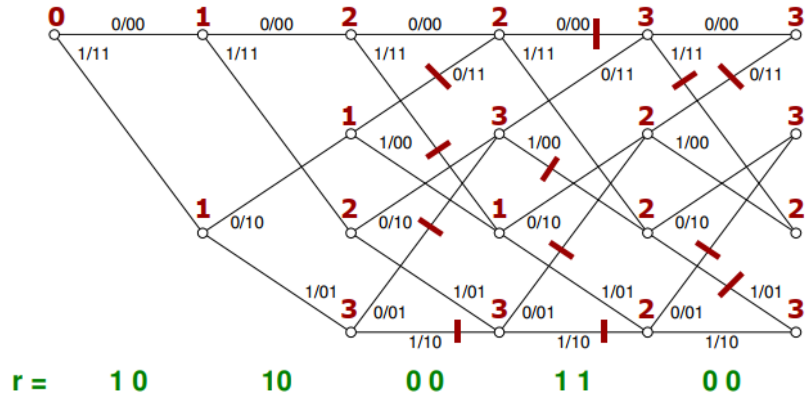
$r = 3 \rightarrow 2\ errors$
$m(x) = x^4 + x + 1$
$\rightarrow from\ table$
$\rightarrow primitive\ over\ GF(2)$
$a^6 = a^3 + a^2$
$H = \begin{bmatrix} 1 & 0 & 0 & & 1 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ 0 & 0 & 0 & & 1 \\ 1 & 0 & 0 & & 1 \\ 0 & 0 & 0 & & 1 \\ 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & & 1 \end{bmatrix}$
$\#rows = 2 * 4 = 8$
$\rightarrow (15,7)\ code$</td></tr>
<tr><td>property</td><td>$u(a^q) = u_0 + u_1 * \alpha^q + \cdots + u_{n-1} a^{q*(n-1)} = 0, q = 1,2, \dots, 2t$
A binary n-tuple $u = (u_0, u_1, \dots, u_{n-1})$ is a code word of a t-error-correcting BCH code of length $n = 2^m - 1$ iff the polynomial $u(x) = u_0 + u_1 * x + \cdots + u_{n-1} * x^{n-1}$ has $a, a^2, \dots, a^{2t}$ as roots</td><td></td></tr>
<tr><td>Generator Polynomial

lcm = least common divisor</td><td>Naive approach $g(x) = (x - a)(x - a^2) \dots (x - a^{2t})$
-> does not work because will not be binary
We need <b>minimal polynomials</b> -> binary coefficients that have a, a^2, a^{2t} as roots
Let $\Phi_i(x)$ be the minimal polynomial of a^i. Then $g(x)$ must be the least comon multiple of $\Phi_1(x), \Phi_2(x), \dots, \Phi_{2t}(x)$
$g(x) = lcm(\Phi_1(x), \Phi_2(x), \dots, \Phi_{2t}(x))$</td><td></td></tr>
</table>

RS-Codes Reed-Solomon used for CD/DVD/ Satellite/ADSL/ xDSL/DVB	Non-binary BCH codes -> $GF(q) \rightarrow usually\ q = 2^m$ A code word consists of $n = q - 1$ code symbols Attention! The code symbols u_i are not binary digits but elements of $GF(q)$. However, if $q = 2^m$, then every code symbol can be represented by a binary vector of length m.	
DFT	Discrete Fourier Transformation of a real vector $\boldsymbol{v} \in \mathbb{R}^n$ $v_k = \sum_{i=0}^{n-1} v_i * e^{-j*\frac{2\pi}{n}*i*k} = \sum_{i=0}^{n-1} v_i * a^{-i*k}, a = e^{j*\frac{2\pi}{n}}$ $a^i \neq 1, \quad 0 < i < n$ $a^n = 1$	
matrix representation	DFT $V = v * A$ $V = v * \begin{bmatrix} a^{-0*0} & a^{-0*1} & \cdots & a^{-0*(n-1)} \\ a^{-1*0} & a^{-1*1} & \cdots & a^{-1*(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ a^{-(n-1)*0} & a^{-(n-1)*1} & \cdots & a^{-(n-1)(n-1)} \end{bmatrix}$ Inverse Transformation $v = V * A^{-1}$ $v = \frac{1}{n} * \begin{bmatrix} a^{0*0} & a^{0*1} & \cdots & a^{0*(n-1)} \\ a^{1*0} & a^{1*1} & \cdots & a^{1*(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ a^{(n-1)*0} & a^{(n-1)*1} & \cdots & a^{(n-1)(n-1)} \end{bmatrix}$ vector $\boldsymbol{v} = \left(v_0, v_1, \dots, v_{(n-1)}\right)$ can be represented by polynomial $\boldsymbol{v}(x) = v_0 x^0 + v_1 x^1 + \cdots + v_{n-1} x^{n-1}$ the DFT of v can be computed by evaluating the polynomial $\boldsymbol{v}(x)$ at $x = a^{-k}$ $v_k = \sum_{i=0}^{n-1} v_i * a^{-i*k} = \boldsymbol{v}(a^{-k})$ the inverse DFT of v can be evaluated with: $v_i = \frac{1}{n} \sum_{k=0}^{n-1} v_k * a^{i*k} = \frac{1}{n} \boldsymbol{v}(a^i)$	
in $GF(2^m)$	Let a be a primitive element of $GF(2^m)$ $a^j \neq 1, j = 1 \dots 2^m - 2$ $a^j = 1, j = 2^m - 1$	
Validation	A vector u is a code word iff its Fourier transform U contains $2 * t$ zeros. $u = (u_0 .. u_{n-1}) \in C \Leftrightarrow U = \left(U_0 \dots U_{n-2t-1} \underbrace{0 \dots 0}_{2t}\right)$ $U_{n-1} = u\left(\alpha^{-(n-1)}\right) = u(\alpha^1) = 0$ $U_{n-2} = u\left(\alpha^{-(n-2)}\right) = u(\alpha^2) = 0$ … $U_{n-2t} = u(\alpha^{n-2t}) = u(\alpha^{2t}) = 0$ The polynomial representation $u(x)$ of a code word has $a^1, a^2, \dots a^{2t}$ as roots $\Rightarrow if\ u(x) = m(x) * g(x) \Rightarrow u(\alpha) = u(\alpha^2) = u(\alpha^{2t}) = 0 \Rightarrow u \in C$ $\Rightarrow u(x)$ must be a multiple of $g(x) = (x - \alpha) * (x - \alpha^2) * \dots * (x - \alpha^{2t})$ $\Rightarrow$ any multiple of $g(x)$ is a valid code polynomial	
Decoding	received vector $r = u + e$ discrete Fourier transform $R = U + E$ $U_i = 0, \quad i = n - 2t, \dots, n - 1$ $E_i = R_i = \boldsymbol{r}(a^{-i}), \quad i = n - 2t, \dots, n - 1$ If the error pattern **e** contains t or less errors, we can generate the whole vector **E** from 2*t known values. -> Berlekamp-Massey Algorithm: Finds the shortest linear feedback shift register (LFSR) that generates the given values of **E**. If the number of symbol errors is t or less, the LFSR will generate the whole vector **E**. Inverse DFT of **E** will give the error pattern **e**.	

14. Convolutional Coding (Faltungscodes) & Turbo Codes

Convolutional Coding	Encoder contains memory n encoder outputs at any given time depend on the k inputs and on m previous input blocks important special case: $k = 1$ encoder is a state machine Rate des codes = k/n (Eingangsbit durch Ausgangsbit) häufig ist k=1	
Encoder example	 $v^{(1)}_k$ u_k D D u_{k-1} u_{k-2} $v^{(2)}_k$ Every input bit u_k yields two output bits $v_k^{(1)}$ and $v_k^{(2)}$ The output bits depend on the actual input bit u_k and two stored bits u_{k-1} and u_{k-2} Number of states = $2^{length\ of\ shift\ register} = 4$	Generator Sequence to describe a state machine. visible in grafic. $g^{(1)} = (g_0\ g_1\ g_2) = (1\ 1\ 1) \rightarrow Encoder$ $g^{(1)}(D) = g_0 D^0 + g_1 D^1 + g_2 D^2 = 1 + D + D^2$ $g^{(2)} = (1\ 0\ 1)$
polynomial representation	▪ Binary sequences $\mathbf{u} = (u_1 \quad u_2 \quad u_3 \quad \cdots)$ $\mathbf{g}^{(1)} = (g_1^{(1)} \quad g_2^{(1)} \quad g_3^{(1)} \quad \cdots)$ $\mathbf{g}^{(2)} = (g_1^{(2)} \quad g_2^{(2)} \quad g_3^{(2)} \quad \cdots)$ $\mathbf{v} = (v_1 \quad v_2 \quad v_3 \quad \cdots)$ ▪ Polynomial representation $\mathbf{u}(D) = u_1 \oplus u_2 \cdot D \oplus u_3 \cdot D^2 \oplus \cdots$ $\mathbf{g}^{(1)}(D) = g_1^{(1)} \oplus g_2^{(1)} \cdot D \oplus g_3^{(1)} \cdot D^2 \oplus \cdots$ $\mathbf{g}^{(2)}(D) = g_1^{(2)} \oplus g_2^{(2)} \cdot D \oplus g_3^{(2)} \cdot D^2 \oplus \cdots$ $\mathbf{v}(D) = v_1 \oplus v_2 \cdot D \oplus v_3 \cdot D^2 \oplus \cdots$ ▪ Discrete convolution becomes multiplication $\mathbf{v}^{(1)}(D) = \mathbf{u}(D) \cdot \mathbf{g}^{(1)}(D)$ $\mathbf{v}^{(2)}(D) = \mathbf{u}(D) \cdot \mathbf{g}^{(2)}(D)$ Code Word: $\mathbf{v}(D) = \mathbf{v}^{(1)}(D^2) \oplus D \cdot \mathbf{v}^{(2)}(D^2)$ ▪ D: delay operator (place holder) ▪ Similar to z-transform	Transformation in digital technic *Faltung im Zeitbereich*
encoder state diagram	 $u_k/v^{(1)}_k\ v^{(2)}_k$ 10 1/11 1/01 0/00 1/10 00 0/10 1/00 11 0/11 01 0/01	

Trellis Diagram	 state 00 01 10 11 k 0/00 1/11 0/11 1/00 0/10 1/01 0/01 1/10 common assumption: encoder starts in the state $(0,0)$ sometiems: a number of zeros is added at the end of the message so that the encoder returns to the state $(0,0)$	
Decoding	Find the path through the trellis that best fits the received data. - Hard decoding: receiver delivers a binary symbol (hamming distance) - Soft decoding: receiver delivers a floating point value (confidence level) square euclidean distance $(r_k - v_k)^2$ about 2dB better than hard decoding	
Viterbi Algorithm	Finds the path through the trellis with the largest (or smallest) metric MLSE – maximum likelihood sequence estimation Principle - At each step, compare the metrics of all path entering each state and store the path with the largest metric (survivor) together with its metric. Eliminate all other paths. - At the end (or after a certain amount of time) the survivor with the best metric is selected and the (first few) bits on this path are chosen as the decoded bits	
	 $M_p[k-1]$ d_{pr} $M_r[k]$ d_{qr} $M_q[k-1]$ - $M_r[k]$: metric of the state r at time k - p, q: predecessor states of the state r - d_{pr}, d_{qr}: branch metrics (e.g. Hamming distance) $M_r[0]=0$ $M_r[k]=\max\limits_{\text{predecessor states } j}\left(M_j[k-1]+d_{jr}\right)$	
	 0 1 2 2 3 3 1 3 2 3 1 2 1 2 2 3 3 2 3 0/00 1/11 0/11 1/00 0/10 1/01 0/01 1/10 r = 1 0 10 0 0 1 1 0 0	

Turbo codes

kein Bestandteil der Prüfung.

BCJR: Formel: 3 Terme: etwas aus der Vergangenheit, etwas vom hier und jetzt und von der Zukunft
Jacobi Symbol

Representations in $GF(2^4)$

- In $GF(2^x) \rightarrow 2 = 0$

				Int	Hex	Bin n_3, n_2, n_1, n_0	Polynomic
$GF(2^4)$	$GF(2^3)$	$GF(2^2)$	$GF(2^1)$	0	0	0000	0
				1	1	0001	1
				2	2	0010	x
				3	3	0011	$x+1$
				4	4	0100	x^2
				5	5	0101	x^2+1
				6	6	0110	x^2+x
				7	7	0111	x^2+x+1
				8	8	1000	x^3
				9	9	1001	x^3+1
				10	A	1010	x^3+x
				11	B	1011	x^3+x+1
				12	C	1100	x^3+x^2
				13	D	1101	x^3+x^2+1
				14	E	1110	x^3+x^2+x
				15	F	1111	x^3+x^2+x+1

Roots of a polynomial

	degree	$in\ \mathbb{Q}$	$GF(2) = [0,1]$	
x	1	$[0]$		
$x+1$	1	$[-1]$		
x^2	2	$[0,0]$		
x^2+1	2	irreducible	$(x+1)(x+1) \rightarrow [-1,-1]$	
x^2+x	2	$[0,-1]$		
x^2+x+1	2	$irreducible$	$irreducible\ \&\ primitive$	
x^3	3	$[0,0,0]$		
x^3+1	3	$[-1]$		
x^3+x	3	$[0]$		
x^3+x+1	3	$irreducible$	$irreducible\ \&\ primitive$	
x^3+x^2	3	$[0,0,-1]$		
x^3+x^2+1	3	$irreducible$	$irreducible\ \&\ primitive$	
x^3+x^2+x	3	$[0]$		
x^3+x^2+x+1	3	$[-1]$		

Functions

q=intDiv(a,b)
r=mod(a,b)
	int: a,b	integer division a/b
		modulo (a mod b)
				$intDiv(9,4) = 2$
				$mod(9,4) = 1$

gcd(a,b) \gcdstep(a,b)	int: a,b	greatest common divisor		$\gcd(10,16) = 2$
\egcd(a,b) \egcdstep(a,b)	int: a,b	extended gcd		$gcde(10,16) = 2$ $10 * (-3) + 16 * 2 = 2$
\phi(n)	int: n	Eulers phi function	$phi(n)$	$phi(10) = 4$
e=\multord(g,n)	int: g,n	multiplicative order	$g^e \equiv 1\ (mod\ n)$	$multord(8,5) = 4$
\gen(g,p)	int: g prim: p	generator / primitive element multiplicative order		$gen(2,7) \rightarrow no, ord = 3$
\chin(m)	matrix: m $(n \times 2)$	Chinese remainder theorem	$x \equiv a_i\ mod\ m_i$	$chin\begin{pmatrix} 5 & 7 \\ 3 & 11 \\ 10 & 13 \end{pmatrix}$ $= M_i \begin{pmatrix} 143 \\ 91 \\ 77 \end{pmatrix}, e_i \begin{pmatrix} 715 \\ 364 \\ 924 \end{pmatrix}$ $x = 894$
\invmod(m,n) \invmodstep(m,n)	matrix: m int: n	inverse of a matrix		$invmodstep\left(\begin{bmatrix} 3 & 2 \\ 1 & 1 \end{bmatrix}, 4\right) = \begin{bmatrix} 1 & 2 \\ 3 & 3 \end{bmatrix}$
\prifiadd(p)	prim: p	addition of prime field		$prifiadd(7)$
\prifimul(p)	prim: p	multiplication of prime field		$prifimul(7)$
\extfiadd(q,m)	int: q poly: m	addition of extended field	$GF(q)$ $m(x) = \cdots$	$extfiadd(2, x^2 + x + 1)$
\extfimul(q,m)	int: q poly: m	multiplication of extended field	$GF(q)$ $m(x) = \cdots$	$extfimul(2, x^2 + x + 1)$
polyQuotient(f,m)	poly: f,m	quationt of a polynom division	f/m	$polyQuotient(x^3 + 1, x^2 + 1)$ $= x$
polyRemainder(f,m)	poly: f,m	remainder of a polynom division	f/m	$polyRemainder(x^3 + 1, x^2 + 1)$ $= 1 - x$
\polgen(p,n,m)	int: p,m poly: m	generate primitive polynoms modulo m	$GF(p^n)$ $m(x) = \cdots$	$polgen(2,3, x^3 + x + 1)$
\sam(a,c,m) \samstep(a,c,m)	int: a,c,m	square an multiply with modulo	$a^c\ mod\ m$	$sam(1234,5678,438) = 316$
\sam2(a,c) \sam2step(a,c)	int: a,c	square an multiply	a^c	$sam(3,4) = 81$
\isprobprime(n)	int n	miller-rabin primality test	$isPrime(n)$	$isprobprime(317) = true$
\isprobprimebase (n,a)	int n,a	with a given base		$isprobprimebase(317,2) = true$
\multmod(ec,p,n,f) \multmodstep	poly: ec point: p int n,f	multiplication on an eliptic curve with modulo	$n * p\ (mod\ f)$	
\addmod(ec,p,q,f) \addmodstep	poly: ec point: p,q int f	addition on an eliptic curve with modulo	$p + q\ (mod\ f)$	
\add(ec,p,q)	poly: ec point: p,q		$p + q$	
\mult(ec,p,n)	poly: ec point: p int: n		$n * p$	
\validate(ec)	poly: ec			
\validatemod(ec,p)	poly: ec			
\numberofpoints(p)	p			
\order(ec,p,f) \orderstep	poly: ec point: p int: f			
\negmod(p,f)	point: p int: f			
\listpoints(ec,f)				