Quantum Computing and Cryptography

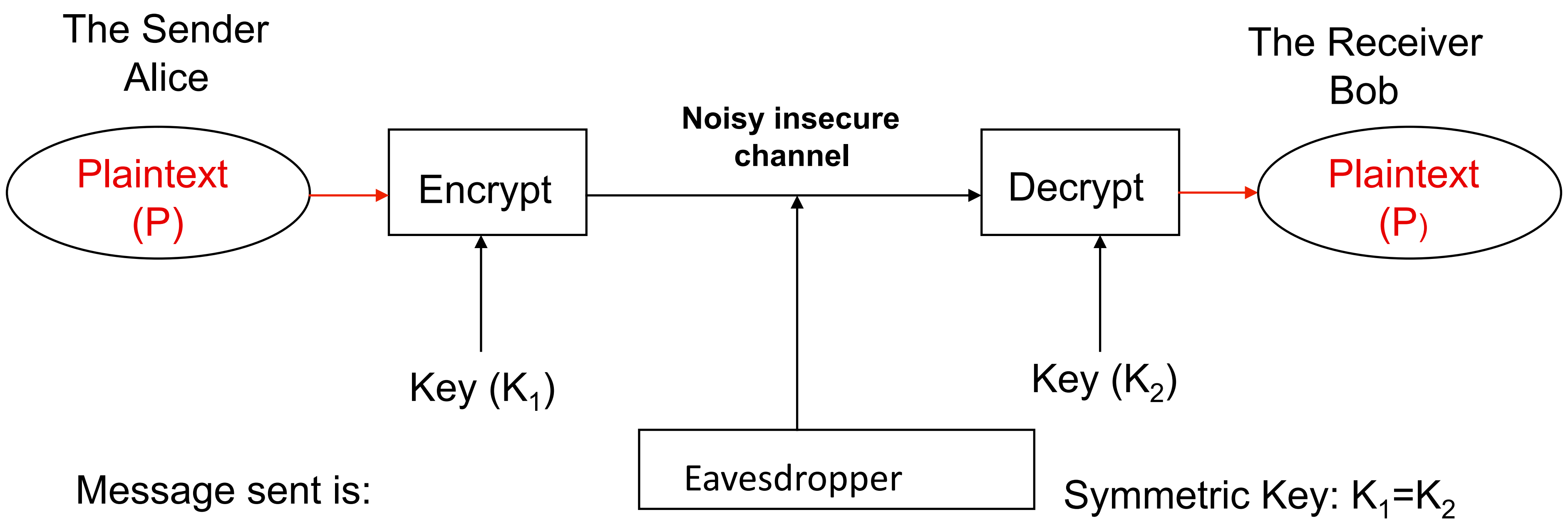
Freedman Symposium

John Manferdelli
JohnManferdelli@hotmail.com
John.L.Manferdelli@intel.com

Outline
Cryptography in the connected world

- The primitives
  - Public Key
  - Symmetric Key
  - Hashes
- Classical attacks
  - Exhaustive search
  - Equation solving
  - Differential and linear cryptanalysis
  - Factoring
  - Discrete Log
  - Information decoding
- Quantum Attacks
  - Shor (Period finding --- factoring and discrete log)
  - Grover (Search --- symmetric key)
- Effects of Quantum Computing (conventional wisdom)
  - Switch from factoring and discrete log based public key systems to McEliece or something
  - Double symmetric key size
- An idea for using a quantum computers
  - On block ciphers

The wiretap channel



Message sent is:

$C= E_{K1}(P)$

Decrypted as:

$P=D_{K2}(C)$

P is called plaintext.

C is called ciphertext.

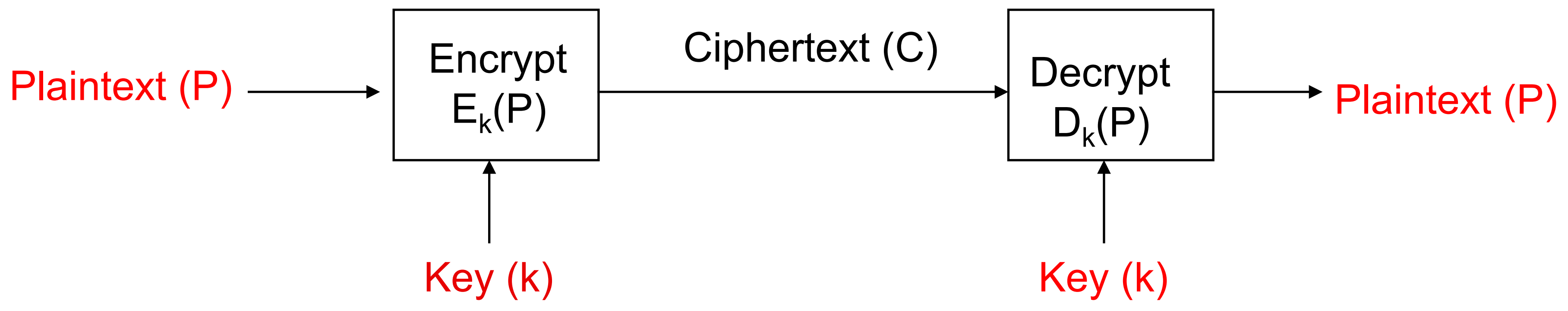
Symmetric Key: $K_1=K_2$

Public Key: $K_1{}^{1}K_2$

K_1 is publicly known

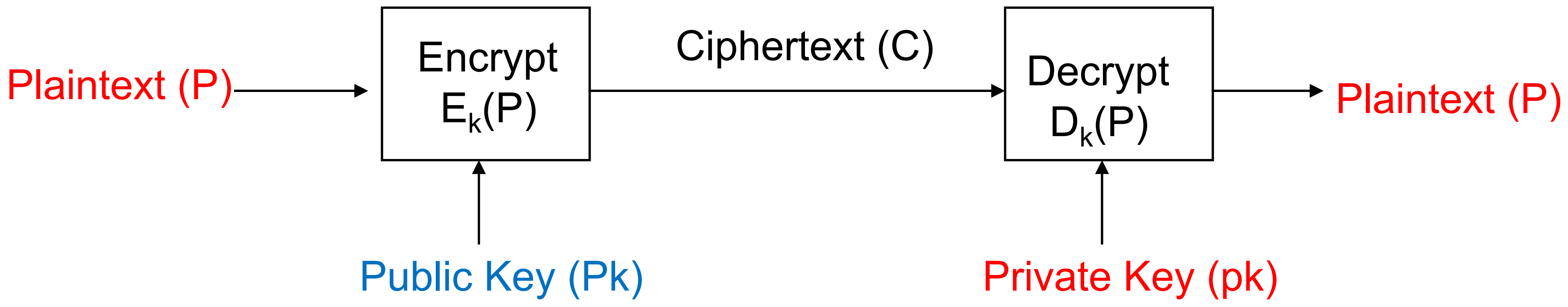
K_2 is Bob's secret

Symmetric ciphers



- Encryption and Decryption use the same key.
  - The transformations are simple and fast enough for practical implementation and use.
  - Two major types: Stream ciphers and block ciphers.
  - Examples: DES, AES, RC4, A5, Enigma, SIGABA, etc.
  - Can't be used for key distribution or authentication.

Asymmetric (public key) ciphers



Encryption and Decryption use different keys.

- Pk is called the public key and pk is the private key. Knowledge of Pk is sufficient to encrypt. Given Pk and C, it is infeasible to compute pk and infeasible to compute P from C.
- Invented in mid 70’s –Hellman, Merkle, Rivest, Shamir, Adleman, Ellis, Cocks, Williamson
- Public Key systems used to distribute keys, sign documents. Used in https:. Much slower than symmetric schemes.

Cryptographic toolchest

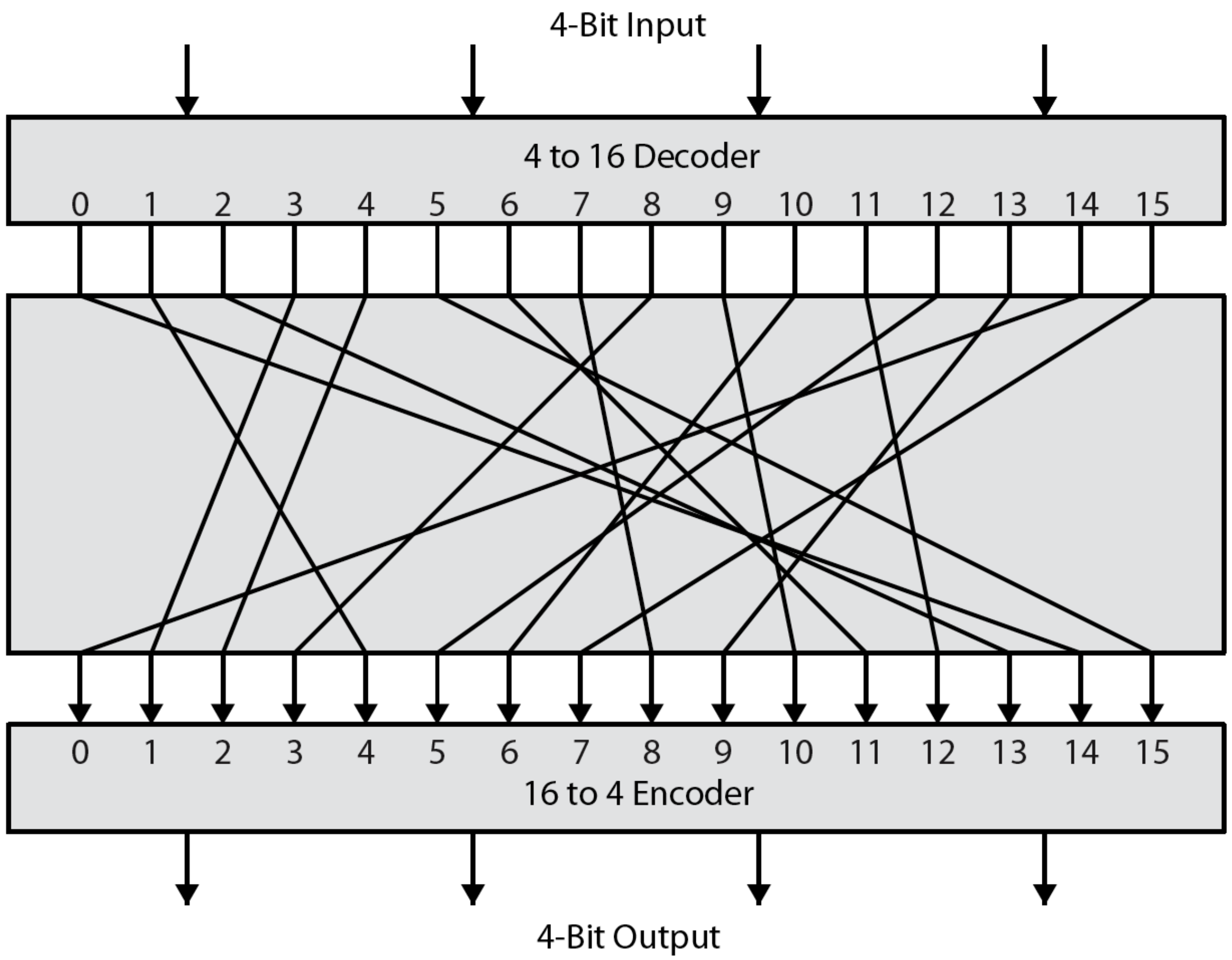
- Symmetric ciphers
  - Block ciphers, stream ciphers
  - Used for bulk encryption
- Asymmetric ciphers
  - Used for key distribution and signatures
- Cryptographic Hashes
  - Used for integrity calculations and identification

Algorithm	**Speed**
RSA-1024 Encrypt	.32 ms/op (128B), 384 KB/sec
RSA-1024 Decrypt	10.32 ms/op (128B), 13 KB/sec
AES-128	.53 ms/op (16B), 30MB/sec
SHA-1	48.46 MB/sec
SHA-256	24.75 MB/sec

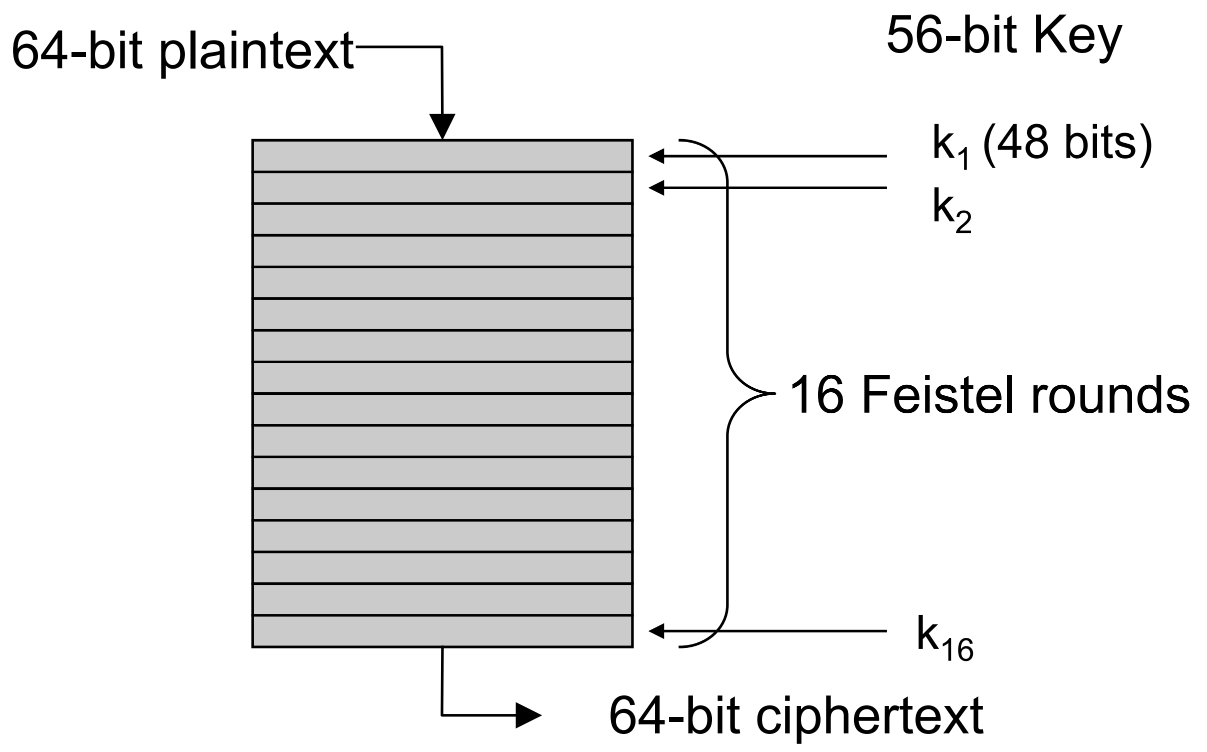
Cryptography and TMITS
The man in the street

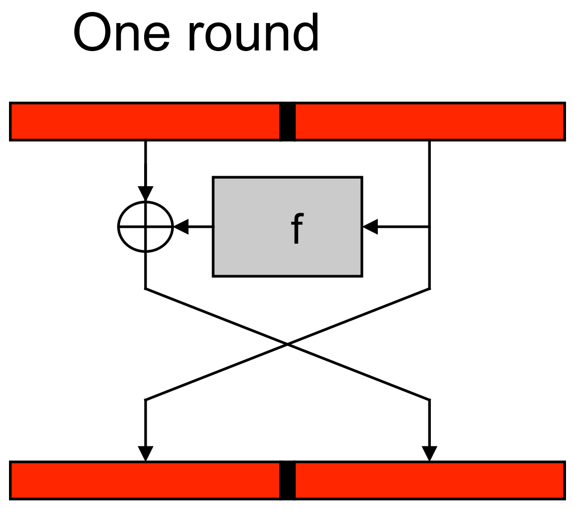
- SSL/TLS (https):
  - Key agreement via Public Key (RSA)
  - Integrity and confidentiality (AES/RC4/SHA1/SHA2)
- Encrypted mail (like S/Mime)
  - Key wrapping via public key (ECC for USG)
  - Integrity and confidentiality via symmetric (AES/ SHA2)
- Encrypted media
- Legal agreements
- Archived data protection

What is a “safe” block cipher



Iterated Feistel cipher --- DES





Note: If s_i(L,R)= (LÅf(E(R)Åk_i), R) and t(L, R)= (R, L), this round is ts_i(L, R).
To invert: swap halves and apply same transform with same key:
$s_i tts_i$(L,R)= (L,R).

DES ---- computing “f”

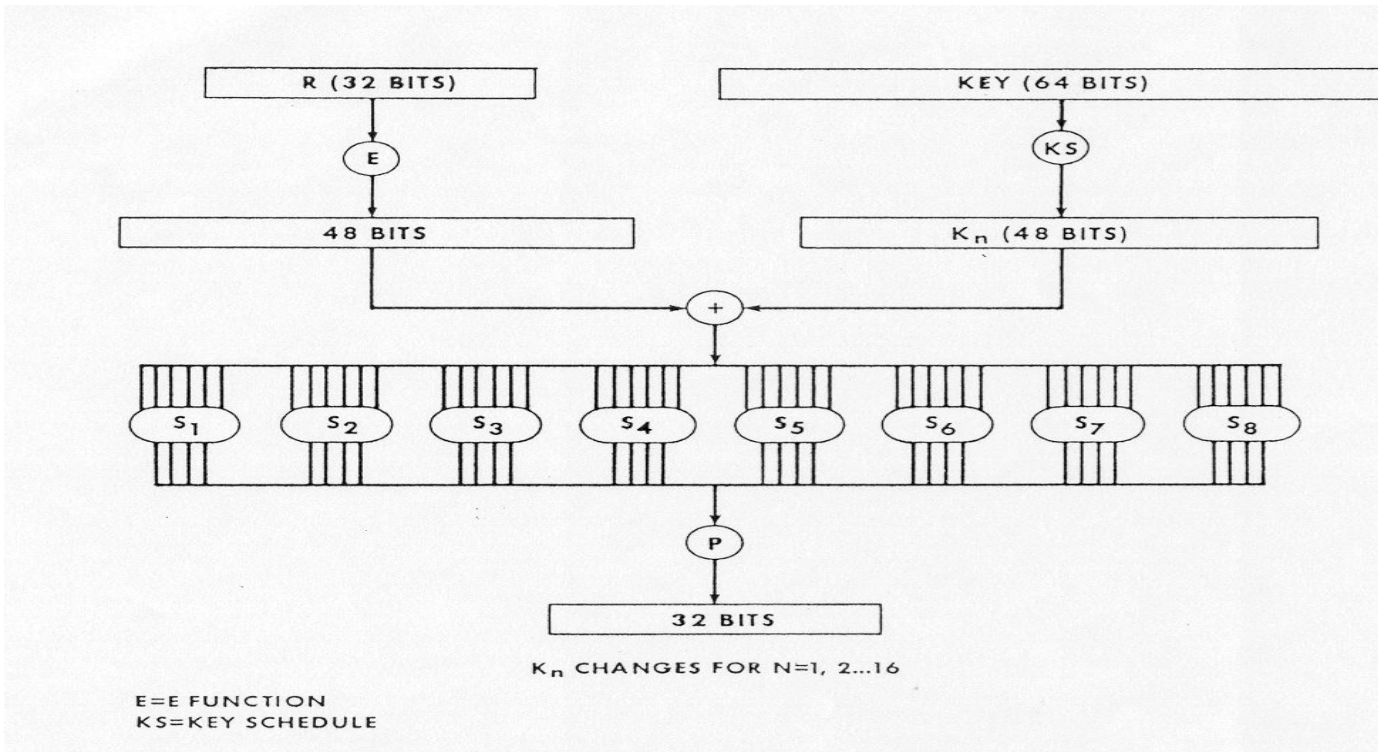


Figure 5.2. Electronic Codebook (ECB) Mode—Calculation of f(R,K).

Sbox as polynomial over GF(2)

```
1,1: 56+4+35+2+26+25+246+245+236+2356+16+15+156+14+146+145+13+135+134+
        1346+1345+13456+125+1256+1245+123+12356+1234+12346
1,2: C+6+5+4+45+456+36+35+34+346+26+25+24+246+2456+23+236+235+234+2346+
     1+15+156+134+13456+12+126+1256+124+1246+1245+12456+123+1236+
     1235+12356+1234+12346
```

Cryptographic Hashes

- A cryptographic hash is a "one way function," h, from binary strings of arbitrary length into a fixed block of size n (called the size of the hash) with the following properties:
    1. Computing h is relatively cheap.
    2. Given y=h(x) it is infeasible to calculate x. ("One way," "non-invertibility" or "pre-image" resistance). Functions satisfying this condition are called One Way Hash Functions (OWHF)
    3. Given u, it is infeasible to find w such that h(u)=h(w). (weak collision resistance, 2nd pre-image resistance).
    4. It is infeasible to find u, w such that h(u)=h(w). (strong collision resistance). Note 4$\rightarrow$3. Functions satisfying this condition are called Collision Resistant Functions (CRFs).

- Turning a good block cipher into a cryptographic hash: Let input be $x = x_1 || x_2 || \dots || x_t$. Let g be a function taking an n bit input to an m bit output. Let E(k, x) be a block cipher with m bit keyspace and n bit block. Let H_0= IV, $H_i = E(g(H_{i-1}), x_i) \oplus H_{i-1}$.

Symmetric key attacks

- Exhaustive search (maybe with help)
- Differential Cryptanalysis
- Linear Cryptanalysis
- Algebraic Cryptanalysis

- Best attacks on DES
  - Exhaustive search: 2^{55}.
  - Linear Cryptanalysis: requires 2^{43} known plaintexts - get 26 bits, brute force the remaining 30 bits.
  - Differential Cryptanalysis: requires 2^{47} blocks.

Differential Cryptanalysis

- Let E and E* be inputs to a cipher and C and C* be corresponding outputs with EÅE*=E' and CÅC*=C'.
- The notation E' → C', p means the "input xor", E' produces the "output xor" C' with probability p. Not all input/output xors and possible and the distribution is uneven. This can be used to find keys. E' → C', p is called a *characteristic*.
- Notation: D_j(x',y')= {u: S_j(u)ÅS_j(uÅx')= y'}. k_jÎxÅD_j(x',y')= t_j (x,x',y'). test(E_j, E_j*,C_j')= t_j(E_j,E_jÅE_j*', C_j')
- For the characteristic 0x34→d in S-box 1 from inputs1Å35=34, D_1(34,d)= {06, 10, 16, 1c, 22, 24, 28, 32} and k_jÎ{7, 10, 17, 1d, 23, 25, 29, 33}= 1ÅD_1(34,d)

Differential Cryptanalysis 4 rounds

Pick

```
    L0', R0': 011010 001100.
```

Then

```
    E(R0'):   0011 1100.
    0011 → 011 with p=3/4
    1100 → 010 with p=1/2
```

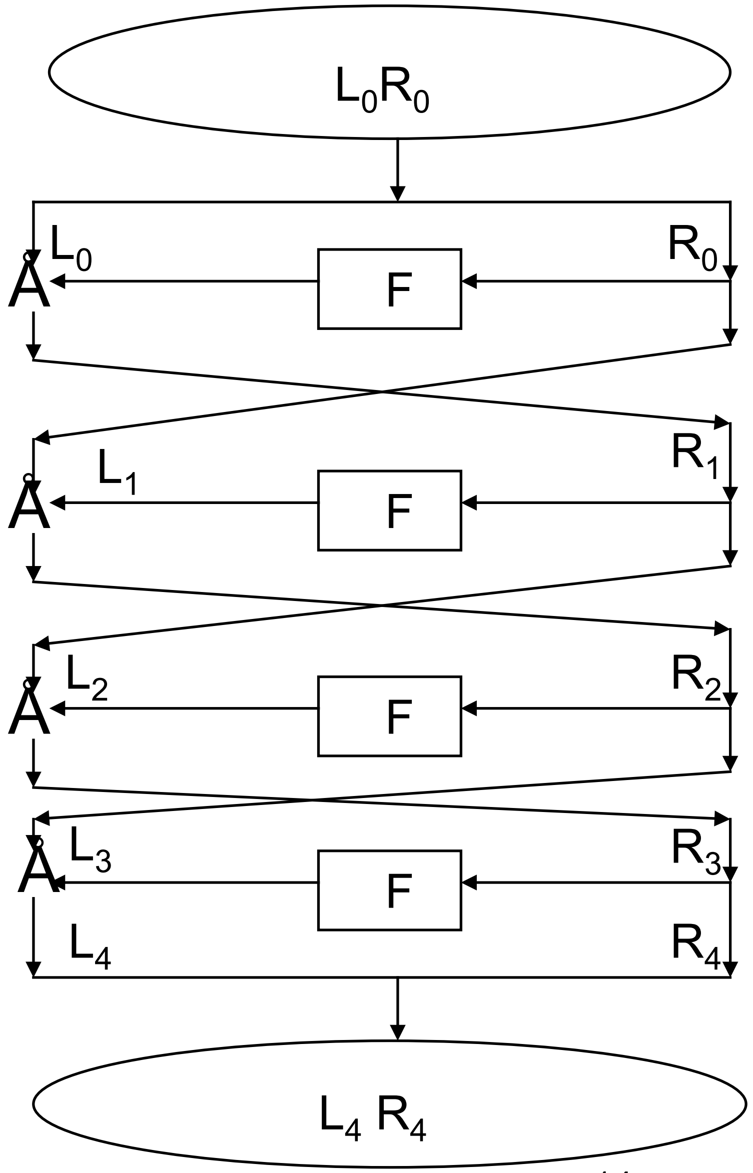
So

```
    f(R0', k1)= 011 010, p=3/8.
```

Thus

```
    L1', R1': 001100 000000, p=3/8.
```

- 3/8 of the pairs with this differential produce this result. 5/8 scatter the output differential at random. These "vote" for 1100 and 0010.



Linear Cryptanalysis

- Basic idea:
  - Suppose $a_i(P)$Å$b_i(C)=g_i(k)$ holds with g_i, linear, for i= 1, 2, …, m.
  - Each equation reduces key search by a factor of 2.
  - Guess (n-m) bits of key. There are $2^{(n-m)}$. Use the constraints to get the remaining keys.
- Can we find linear constraints in the “per round” functions and knit them together?
- No! DES per round functions do not have linear constraints.
- Next idea
  - Can we find a(P)Åb(C)= g(k) which holds with probability p?
  - Each will “vote” for g(k)=0 or g(k)=1.
- p= 1/2+e
  - Breaking cipher requires ce^{-2} texts
  - e is called “bias”.

Linear Cryptanalysis: 3 round DES

$X[17]\ \text{Å}Y[3,8,14,25]= K[26]\ \text{Å}1$, $p= 52/64$

- Round 1

$X_1[17]\text{Å}Y_1[3,8,14,25]= K_1[26]\text{Å}1$

$P_R[17]\text{Å}P_L[3,8,14,25]\text{Å}R_1[3,8,14,25]= K_1[26]\ \text{Å}1$

- Round 3

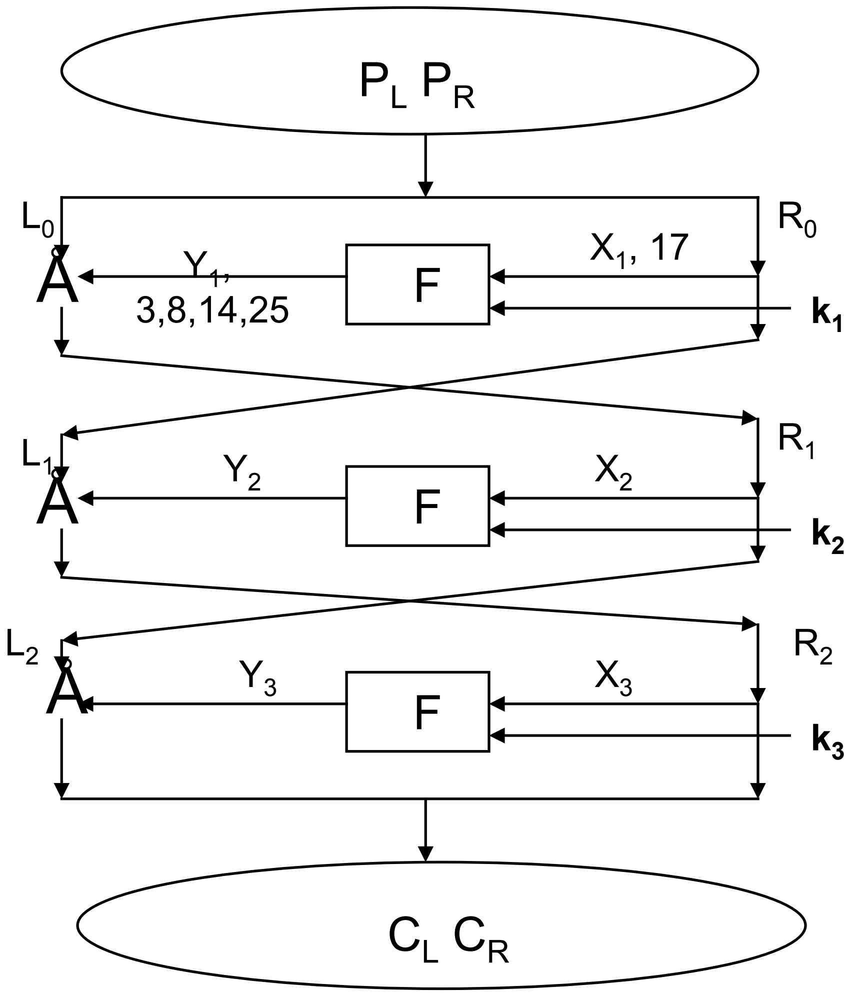
$X_3[17]\text{Å}Y_3[3,8,14,25]= K_3[26]\text{Å}1$

$R_1[3,8,14,25]\text{Å}C_L[3,8,14,25]\text{Å}C_R[17]= K_3[26]\ \text{Å}1$

- Adding the two get:

$P_R[17]\text{Å}P_L[3,8,14,25]\text{Å}C_L[3,8,14,25]\text{Å}C_R[17]=$
$K_1[26]\text{Å}K_3[26]$

Thus holds with $p= (52/64)^2+(12/64)^2=.66$



Matsui’s Per Round Constraints

	SBox	Sbox Equation	Prob	Round Equation
A	5	X[2]ÅY[1,2,3,4]= K[2]Å1	12/64	X[17]ÅY[3,8,14,25]=K[26]
B	1	X[1,4,5,6]ÅY[1,2,3,4]= K[1,4,5,6]Å1	22/64	X[1,2,4,5]ÅY[17]=K[2,3,5,6]
C	1	X[2]ÅY[1,2,3,4]= K[2]Å1	30/64	X[3]ÅY[17]=K[4]
D	5	X[2]ÅY[1,2,3]= K[2]	42/64	X[17]ÅY[8,14,25]=K[26]
E	5	X[1, 5]ÅY[1,2,3]= K[1,5]Å1	16/64	X[16,20]ÅY[8,14,25]=[25,29]

Rounds	Equation	Prob	Eqns used
5	P_L[17]ÅP_R[1,2,4,5,3,8,14,25]ÅC_L[17]ÅC_R[1,2,4,5,3,8,14,25] = K_1[2,3,5,6]ÅK_2[26]ÅK_4[26]ÅK_5[2,3,5,6]	$\frac{1}{2}+1.22\times2^{-6}$	BA-AB
15	P_L[8,14,25]ÅP_R[16,20]ÅC_L[3,8,14,25]ÅC_R[17]= K_1[9,13]ÅK_3[26]ÅK_4[26]ÅK_5[26]ÅK_7[26]ÅK_8[26]ÅK_9[26]ÅK_{11}[26]ÅK_{12}[26]ÅK_{13}[26]ÅK_{15}[26]	$\frac{1}{2}+1.19\times2^{-22}$	E-DCA-ACD-DCA-A
16	P_L[8,14,25]ÅP_R[16,20]ÅC_L[17]ÅC_R[1,2,4,5,3,8,14,25] = K_1[9,13]ÅK_3[26]ÅK_4[26] ÅK_5[26]ÅK_7[26]ÅK_8[26]ÅK_9[26]ÅK_{11}[26]ÅK_{12}[26]ÅK_{13}[26]ÅK_{15}[26]ÅK_{16}[2,3,5,6]	$\frac{1}{2}-1.49\times2^{-24}$	E-DCA-ACD-DCA-AB

Hadamard transforms: linear approximations

- For f: $GF(2)^n \rightarrow GF(2)$, define $F(w)= 2^{-n}S_x (-1)^{f(x)Åw\cdot x}$
- Let *a* be the number of x: f(x)= w•x; *d* the number of x for which f(x)≠w•x.
  $a-d= 2^n F(w)$ and $a+d=2^n$ so Prob(f(x)= w•x)= $a/2^n$= ½ (1+F(w)).
- Example:
  - S-box 5 Y[1,2,3,4]ÅX[2]= K[2] Å1
- Also allows us to calculate the polynomial representation of f(x) over GF(2).

```
Walsh transform of 0f dot S5
   0.0000    0.0000   -0.1250   -0.1250    0.0000    0.1250    0.0000   -0.1250
   0.1250   -0.1250    0.0000    0.0000   -0.1250    0.0000   -0.1250    0.0000
  -0.6250    0.1250    0.0000    0.0000    0.0000    0.1250    0.0000    0.1250
  -0.1250   -0.1250    0.0000    0.0000    0.0000    0.1250    0.0000   -0.1250
   0.0000    0.0000   -0.3750   -0.1250    0.0000   -0.1250    0.0000   -0.1250
   0.0000    0.0000    0.1250    0.1250    0.0000    0.1250    0.0000   -0.1250
   0.1250    0.1250    0.0000    0.0000    0.0000    0.1250    0.0000   -0.1250
   0.0000    0.0000   -0.1250    0.1250    0.1250    0.0000    0.1250    0.0000
Position of largest (w): 16, F(w):   0.6250, p:   0.8125
```

Note: High weight components have $(.625)^2+(.375)^2$=53% of the probability distribution

Public Key Systems and attacks

- RSA (Factoring)
- El Gamal (Discrete Log)
- Elliptic Curve Cryptosystem (Elliptic Curve Discrete Log)
- McEliece (Based on Coding theory)

RSA Public-Key Cryptosystem

<u>Alice (Private Keyholder)</u>

- Select two large random primes p and q.
- Publish the product n=pq.
- Use knowledge of p and q to compute Y.

<u>Bob (Public Key Holder)</u>

- To send message Y to Alice, compute $Z=Y^X \bmod n$.
- Send Z and X to Alice.

Rivest, Shamir and Adleman, "On Digital Signatures and Public Key Cryptosystems." CACM, 2/78.

RSA Example

- p=691, q=797, n=pq=550727. f(n)= 690 x 796= 2^3x3x5x23x199.
- Need (e, f(n))=1, pick e=7.
- 1= 7 x 78463 + (-1) f(n), so d= 78463.
- 78463= 2^{16}+ 2^{13}+ 2^{12}+ 2^9+ 2^6+ 2^5+ 2^4+ 2^3+ 2^2+ 2^1+ 2^0 = 65536+8192+4096+512+64+32+16+8+4+2+1. Use this in the successive squaring calculation.

- Public Key: <n=550727, e=7>
- Private Key: <p=691, q=797, d=78463>.
- Encrypt 10. 10^7 (mod n)= 86914.
- Decrypt: $(86914)^{78463}$ (mod n)=10.
- Successive squares: 86914, 271864, 268188, 407871, 97024, 79965, 460755, 375388,444736, 362735, 289747, 500129, 378508,532103, 446093, 371923, 66612.

Attacks on RSA: factoring

- We want to factor n= pq.
- Basic trick (Krachick)
  - Find x,y such that $x^2-y^2= (x-y)(x+y)=0 \pmod{n}$. Compute gcd(x-y,n).
- One way to find x, y:
  - $f(x)= x^2+1 \pmod{n}$.
  - $x_{i+1}= f(x_i) \pmod{n}$. Loop expected after about Ö(pn/2) steps).
  - Example: n=1517.
    - $f(952)= 952^2+1 \pmod{1517}= 656$
    - $f(360)= 360^2+1 \pmod{1517}= 656$
    - $952^2-360^2= (952-360)(952+360)$.
    - 952-360=592, (592, 1517)= 37.

How fast can we factor classically

- Factor bases
- Quadratic Sieve
- ECM
- Number Field Sieve

- Roughly same asymptotic performance:
  - Define $L_n[u,v]= \exp(v(\lg(n))^u(\lg(\lg(n)^{(1-u)})$.
  - Quadratic Sieve: Sieving time is $L_n[1/2, v+1/(4v)]$, solving sparse equations is $L_n[1/2, 2v+o(1)]$. Total time is minimized when $v=1/2$ and is $L_n[1/2, 1+o(1)] = \exp(\lg(n)^{1/2}\lg(\lg(n))^{1/2})$. Same for ECM.
  - Number Field Sieve is a little better $L_n[1/3, (64/9)^{1/3}]$ or $\exp((64/9)^{1/3}\lg(n)^{1/3}\lg(\lg(n))^{2/3})$.

El Gamal cryptosystem

- Alice is the private keyholder. She:
  - Picks a large prime, p, where p-1 also has large prime divisors and a generator, g, for F_p^*. $\langle g \rangle = F_p^*$. Alice also picks a random number, a (secret), and computes $A=g^a \pmod p$.
  - Alice's public key is <A, g, p>.
- To send a message, m, Bob
  - Picks a random b (his secret) and computes $B= g^b \pmod p$. Bob transmits $(B, mA^b)= (B, C)$.
- Alice decodes the message by computing $CB^{-a}=m$.

El Gamal Example

- Alice chooses
  - p=919. g=7.
  - a=111, A= 7^{111}= 461 (mod 919).
  - Alice’s Public key is <919, 7, 461>
- Bob wants to send m=45, picks b= 29.
  - B=7^{29} =788(mod 919), 461^{29}= 902 (mod 919),
  - C= (45)(902)= 154(mod 919).
  - Bob transmits (788, 154).
- Alice computes $(788)^{-111}$= 902^{-1}(mod 919).
  - (54)(902)+(-53)(919)=1. 54= 902^{-1} (mod 919)
  - Calculates m= (154) (54)=45 (mod 919).

Attack on El Gamal systems: Index Calculus

- $g^x = y \pmod{p}$. B= $(p_1, p_2, \ldots, p_k)$.
- Precompute
  - $g^{x_j} = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$
  - $x_j = a_{1j} \log_g(p_1) + a_{2j} \log_g(p_2) + \ldots + a_{kj} \log_g(p_k)$
  - If you get enough of these, you can solve for the $\log_g(p_i)$
- Solve
  - Pick s at random and compute $y\, g^s = p_1^{c_1} p_2^{c_2} \ldots p_k^{c_k}$ then
  - $\log_g(y) + s = c_1 \log_g(p_1) + c_2 \log_g(p_2) + \ldots + c_k \log_g(p_k)$

- $L_p[1/3, (64/9)^{1/3}]$ for fastest implementation.

- LaMacchia and Odlyzko used Gaussian integer index calculus variant to attack discrete log.

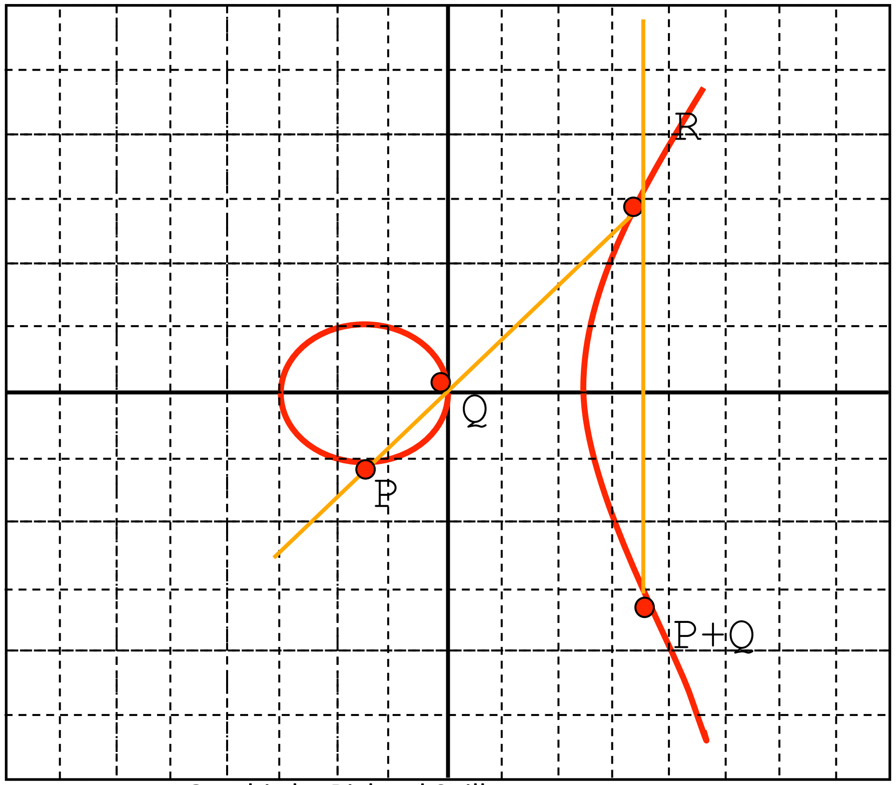
Are there better groups for the DL problem?

- Abstract Discrete Log problem: In group, G (using multiplicative notation), given $y=g^n$, find n.
- In abelian group with additive notation this is: given y= ng, find n.
- Enter elliptic curves
- A non-singular Elliptic Curve is a curve, having no multiple roots, satisfying the equation: $y^2=x^3+ax+b$ in a field F.
  - Can define an “addition” operation on rational points.
  - Over a finite field, say, F=GF(p), for large p, point group is finite (and has elements of “large order”.
  - DL problem in Elliptic Curve, Given P=mB, find m.

Elliptic curve addition

- The addition operator on a non-singular elliptic curve maps two points, P and Q, into a third "P+Q". Here's how we construct "P+Q" when $P \neq Q$:
  - Construct straight line through P and Q which hits E at R.
  - P+Q is the point which is the reflection of R across the x-axis.

- If $P=(x, y)$, then $-P= (x, -y)$.
- Put $P=(x_1, y_1)$ and $Q=(x_2, y_2)$
- Can calculate $P+Q=(x_3, y_3)$ by:
  - $\lambda=(y_2-y_1)/(x_2-x_1)$ (mod p) if $P \neq Q$
  - $\lambda=(3(x_1)^2+a)/(2y_1)$ (mod p) if $P=Q$
  - $x_3= l^2-x_1-x_2$
  - $y_3 =\lambda(x_1-x_3)-y_1$ (mod p)
- The order of P is smallest n: nP=O



Graphic by Richard Spillman

Elliptic Curves

- Motivation:
  - Full employment act for number theorists
  - *Index calculus attack doesn't work on elliptic curves.*
  - Even for large elliptic curves, field size is relatively modest so arithmetic is faster.
  - Security/bit is higher

ECC	RSA	AES
163	1024	
256	3072	128
384	7680	192
521	15360	256

- We need to:
  - Find an elliptic curve whose arithmetic gives rise to large finite groups with elements of high order
  - Figure out how to embed a message in a point multiplication.
  - Figure out how to pick "good" curves.

Elliptic curve El Gamal

- Alice choses a finite field F_p and an elliptic curve $E_p(u,v)$ and a base point B on $E_p(u,v)$.
  - Alice picks rendom a (secret) and computes P=aB.
  - Alice’s public key is <$E_p(u,v)$, B, P>.
- Bob encrypts m by:
  - Selecting a random number b, and finding a point on the curve P_m corresponding to m.
  - The ciphertext consists of two points on the curve <bB, P_m+bP>
  - To decipher, Alice multiplies the first point by a and subtracts the result from the second point to get P_m
- Example:
  - $E_{8831}(3,45)$, B=(4,11), a=3, P=aB=(413,1808)
  - P_m= (5, 1743), b=8, bB= (5415, 6321).
  - Cipher text is <(5415,6321), (6626,3576)>.
  - Decryption: 3 (5415, 6321)= (673, 146), P_m= (6626,3576)-(673,146)= (6626,3576)+(673,-146)= (5, 1743).

McEliece Cryptosystem

- Bob chooses G for a large [n, k, d] linear code, we particularly want large d (for example, a [1024, 512, 101] Goppa code which can correct 50 errors in a 1024 bit block). Pick a k x k invertible matrix, S, over GF(2) and P, an n x n permutation matrix, and set G_1=SGP. G_1 is Bob's public key; Bob keeps P, G and S secret.
- To encrypt a message, **x**, Alice picks an error vector, **e**, and sends $\mathbf{y}=\mathbf{x}G_1+\mathbf{e}$ (mod 2).
- To decrypt, Bob, computes $\mathbf{y_1}=\mathbf{y}P^{-1}$ and $\mathbf{e_1}=\mathbf{e}P^{-1}$, then $\mathbf{y_1}=\mathbf{x}SG+\mathbf{e_1}$. Now Bob corrects $\mathbf{y_1}$ using the error correcting code to get $\mathbf{x_1}$. Finally, Bob computes $\mathbf{x}=\mathbf{x_1}S^{-1}$.
- Error correction is similar to the "shortest vector problem" and is believed to be "hard." In the example cited, a [1024, 512, 101] Goppa code, finding 50 errors (without knowing the shortcut) requires trying ${}_{1024}C_{50}>10^{85}$ possibilities.
- A drawback is that the public key, G_1, is large.

McEliece Cryptosystem example

- Using the [7, 4] Hamming code

$$G = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{matrix}$$

- **m**=1011.
- $S = \begin{matrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{matrix}$ $\quad P = \begin{matrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{matrix}$

Quantum Attacks

- General Attacks
  - Shor (Factoring and discrete log)
  - Grover (Search --- symmetric key)
- Special attacks

Shor’s Algorithm

- Factor N. lg(N)=m.
- Best know classical algorithm is O(exp($m^{1/3}$) lg(m)$^{2/3}$).

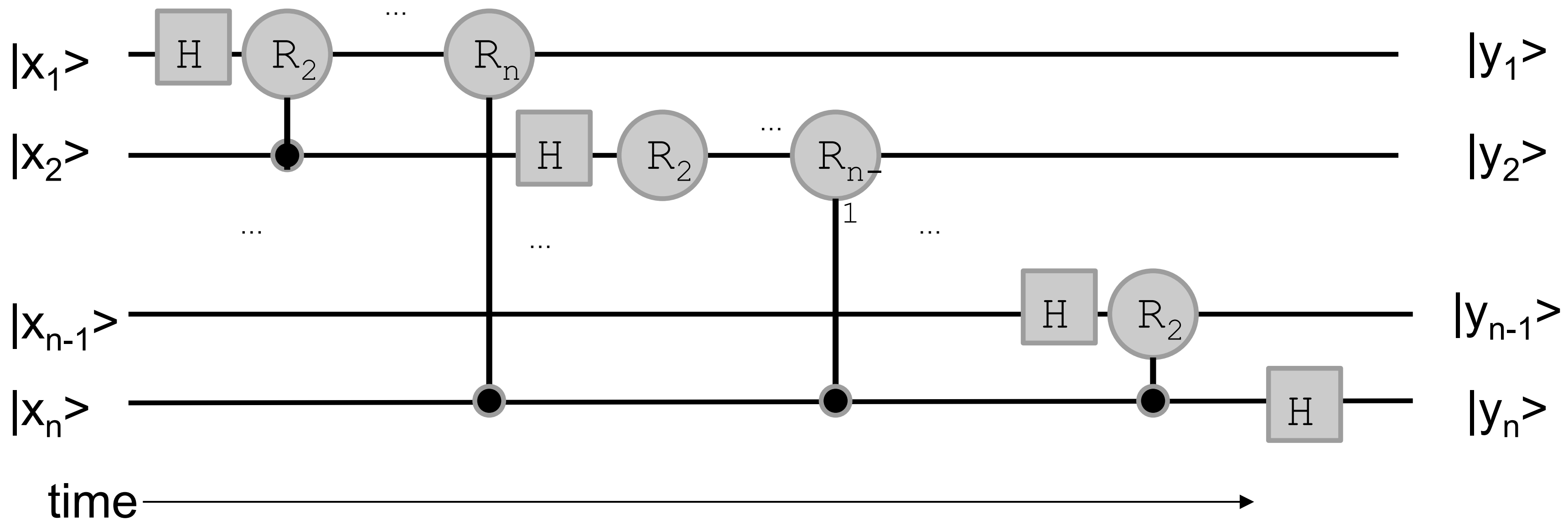
- Classical Part:
  1. Pick a<N
  2. Compute (a, N)
  3. If (a, N)=1
  4. Find period, r, of f(x)= a^x (mod N) using quantum period finding algorithm.
  5. If r is odd, repeat from 2
  6. If $a^{r/2}$=-1, repeat from 2
  7. Compute ($a^{r/2}$±1,N)

Period finding quantum algorithm

1. $Q=2^q$, N£Q£2N^2, period is r.
2. Initialize |s>= $Q^{-1/2}$ $S_{x=0}^{Q-1}$ |x>|0>. This can be accomplished with the quantum circuit ($H^{\otimes n}\otimes$1)(|x>)|0>.
3. Pick a. Construct quantum circuit to compute f(x)= a^x (mod N) and apply to |s> obtaining $Q^{-1/2}$ $S_{x=0}^{Q-1}$ |x>|f(x)>.
4. Measure second register to get |a^t (mod N)>. After measurement, first register is in the state C(|t>+|t+r>+|t+2r>+…+|t+mr>). C is normalization factor.
5. Apply QFT to first register to get C$Q^{-1/2}$ S_x S_y w^{xy} |y>
6. Measure first register to get |k>. k is likely to be j(Q/r).
7. Apply continued fraction approximation to k/Q to obtain j’/p’:
   - If |k/Q-j’/r’|<Q/2, the probability that r=r’ is high
8. Check f(x)=f(x+r’), output r’.

Quantum Fourier Transform

- QFT$_Q$(|x>)= 1/ÖQ $S_{y=0}^{Q-1}$ exp[(2pixy/Q)] |y>
- Built out of rotation gates and Hadamard gates O(n^2).
- Notation: R$_n$= $\begin{pmatrix}1 & 0\\0 & \exp(i\theta)\end{pmatrix}$, q= p/2^{n-1} .



Shor example

- n=21, q=9, Q=2^9=512, a=11. 21^2£512<2(21^2).
- Start with |000000000> Apply $H^{\otimes q}$ to get $|s_0$>= $2^{-9/2}$ (|000000000> + |000000001> + |000000010> + … + |111111111>).
- We'll write as $|s_0$>= $2^{-9/2}$ (|0>+|1>+|2>+ … +|511>)
- Suppose $U_{f,a,N}$(x) is the quantum circuit that computes f(x)=a^x (mod N).
- Apply $U_{f,a,n}$(x) to get $|s_2$>= $2^{-9/2}$ (|0,f(0)>+|1,f(1)>+… +|511, f(511)>)= 1/Ö(512) (|0,1>+|1,11|+|2,16>+…+|511,11>)
- Measure the second set of qubits. Say we get |2>, we obtain:

  $|s_3$>= 1/Ö(85) (|5,2>+|11,2|+…+|509,2>).
- Apply the QFT $|s_3$> to get $|s_4$>=1/Ö(85)(g(0)|0>+g(1)|1>+…+g(511)| 511>).

Shor Example

- g(k) is the DFT of 0,0,0,0,0,1,0,0,0,0,0,1…,0,0,0,0,0,1 of period 6.
- The coefficients g(0), g(85), g(171), g(256), g(341), g(427) are large (around 3.1) because they are multiples of 512/6=85 although at this point in the algorithm we don't know r.
- Now measure the state of $|s_4>$, the sum of the amplitudes of |0>, |85>, |171>, |256>, |341>, |427> is about .456 so if we do this a few times we are likely to get one of these values.
- Say we get t=427. We're looking for the period r (6 in this case) and we know 6·85 » 512 and j·85 » 427.
- Apply the continued fraction method to find j/r close to 427/512.
- We get 5/6. 6= 2·3.
- (21, 6)=3.

Shor discrete log finder

- Problem: $a^{q-1}=1$ (mod q), b= a^d (mod q), find d. q is large prime.
- Uses three quantum registers

1. As usual, set up initial state of first two registers as

   $|s_0\rangle = 1/q \; S_x \; S_y \; |x,y,0\rangle$

2. Construct quantum circuit for $|x\rangle|y\rangle|0\rangle \rightarrow S \; |x,y, a^x b^y \text{ (mod q)}\rangle$ and apply this to $|s_0\rangle$.
3. Measure third register to get $|a^t\rangle$. Each component $|x,y\rangle$, of first two registers must satisfy t=x+dy (mod q-1). This state is

   $|s_1\rangle = C \; S_{y=0}^{\;q-1} \; |t-dy,y\rangle$

4. Apply QFT to first two registers to get $1/q \; S_{y'=dx' \text{ (mod q-1)}} \; w_0^{tt'}|x', y'\rangle$.
5. Similar approximations yield d.

Elliptic Curve discrete log

- Proos and Zalka, “Shor’s discrete logarithm quantum algorithm for ellliptic curves.”
- Remember, for classical attacks, the work to attack ECC was worse for ECC than RSA, for equivalent classical RSA, ECC protection, quantum attacks on ECC make it worse!

RSA n	RSA qubits	RSA time	ECC n	ECC qubits	ECC time
512	1024	$.54 \times 10^9$			$.5 \times 10^9$
1024	2048	4.3×10^9	163	1000	1.6×10^9
3072	6144	120×10^9	256	1500	6×10^9

Grover’s Algorithm

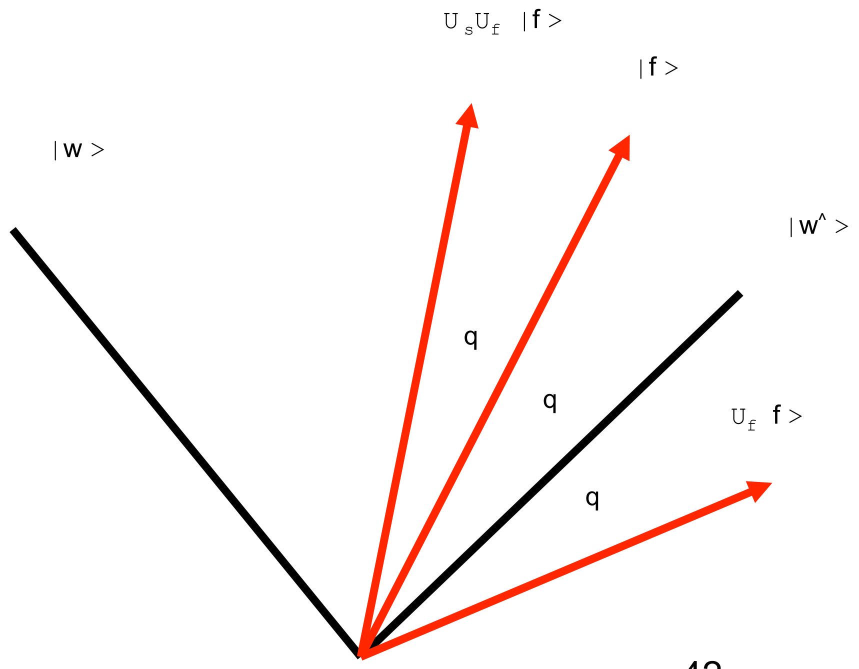
- Given an unsorted sequence, $<a_0, \dots , a_{N-1}>$ find b.
- Best known classical algorithm is O(N).
- Obtain [lg(N)] qubits. W an observable on H with basis |0>, …, |N-1> with distinct eigenvalues $l_1,\dots, l_{N-1}$.
- Proceed as follows:
  1. Construct quantum circuit simulating $U_w|x>=|x>$, x^1w, $U_w|w>=-|w>$.
  2. Initialize $|s>= N^{-1/2}\ S_{x=0}^{N}\ |x>$
  3. Perform Grover iteration r(N)= p/4 $N^{1/2}$ times
     a. Apply U_w
     b. Apply U_s
  4. Perform measurement on W.

Grover's Algorithm

- $N= 2^n$
- At first iteration:
  - $<w|s>=<s|w>=N^{-1/2}$, $<s|s>= N^{1/2}$.
- $U_w|s>= |s>- 2N^{-1/2}|w>$,
- $U_s(|s>- 2N^{-1/2}|w>)=$

  $(N-4)/N |s> + 2N^{-1/2}|w>$.
- After applying U_w and U_s the amplitude for desired element increases from 1/N to » 9/N.
- Grover iterate moves |s> to |w>. After about p/2 ÖN iterates they coincide more or less with high probability.

- $q= \ 2^{-n/2}$
- $U_G= U_sU_f$
- Apply k» p/4 $2^{-n/2}$



Effects of Quantum Computing (conventional wisdom)

- Switch from factoring and discrete log based public key systems to McEliece or something
- Double symmetric key size, hash size too

- Your job is safe, no need to learn quantum computing

Quantum Hadamard

- $(H^{\otimes n})(|x>_n) = 2^{-n/2} \sum_y (-1)^{x \cdot y} |y>_n$.
- $(H^{\otimes n} \otimes 1) U_f (H^{\otimes n} \otimes H)(|0>_n |1>_1) =$

  $2^{-n} \sum_y \sum_x (-1)^{f(x) \text{Å} x \cdot y} |y> 1/\sqrt{2}(|0>-|1>)$.
- Measure y, linearity test till satisfied.
- Have found y: $y \cdot x$ is a good linear approximation of f(x)

Happy Birthday, Mike and thanks!

How likely?

- Let d=d(e) be probability that single boolean has at least one Walsh coefficient ≥e then the n bits of the block cipher in some linear combination have probability $p_e = 1-(1-d)^N \geq Nd$ of having a Walsh coefficient ≥e, $N=2^n$. Say $p_e=1/2$, this happens if $d \geq 2^{-(n+1)}$.
- What is d for $e=2^{-25}$?
- Let $\mathcal{A}_n$ be the set of affine functions in n variables, $|\mathcal{A}_n|=2^{n+1}$.
- Let $\mathcal{F}_n=\{f: GF(2)^n \rightarrow GF(2)\}$, $|\mathcal{F}_n|=2^N$, $N=2^n$.
- The number of vectors in $GF(2)^N$ within hamming distance h of v is $N_h(v)=(1+{}_NC_1+{}_NC_2+\ldots+{}_NC_h)$. So the total number of boolean functions within Hamming distance h of an affine function is $2^{n+1}N_h(0)$.
- Probability that a boolean function is within distance h of an affine function is $p_{n,h}= 2^N/(2^{n+1}N_h(0))$.
- What h makes $e=2^{-25}$?
- $d_e= 2^N/(2^{n+1}N_h(0))$.

Non-linearity

- nf(f)= $\text{min}_{\text{lÎAFG(n,2)}}$(dist(f,l))

- Theorem: Let f be a boolean function from GF(2)$^n \rightarrow$GF(2). The number, $N_{n,t}$ of {f: nl(f)=t} is (2^n choose t) 2^{n+1} if t < 2^{n-2} and 2^{n+1}(2^n choose 2^{n-2})–(2^{n-1}) [(2^{n-1} choose 2^{n-1}) + (2^{n-1} choose 2)] if t= 2^{n-2}

- Let X be a random variable representing the minimum distribution for distance to linear function. m_X= 2^{n-1} – C_n √(n2^n), $\text{lim}_{n\rightarrow \text{¥}}C_n$= √(2ln(2)). s_X^2=Var(X)= 2^n.

- P(m-ns, m+ns)= erf(n/√2), erf(x)= 2/√p $\int_{[0,x]}$exp(-t^2).