Quick Cryptography Intro

Gayn B. Winters, Ph.D.

(c) 2010, 2011, 2012, 2013, 2014, 2015

Topics Today

- Encryption
  - Symmetric (Shared secret key): shifts, substitutions, permutations, stream and block ciphers, DES, AES
  - Asymmetric (Public+Private keys): RSA, El Gamal, Elliptic Curve
- Hash functions and digital signatures
- Session keys, SSL/TLS, HTTPS

Future Talks

- Attacks and Secrecy
- Applications: blind signatures, anonymous communication and email, Tor, pseudonyms, digital cash, open transactions, voting, zero-knowledge proofs, Bitcoin, ...
- Privacy, Off-the-record messaging, startpage...
- Forensic and anti-forensic techniques
- Security: Attack prevention, detection, and recovery
- Quantum and Post-Quantum cryptography

Google yields many great papers, also Wikipedia has excellent, mostly current, articles. YouTube has some good talks. Books tend not to be current … caveat emptor...

History

- Will make some historical comments
- Read: David Kahn's Codebreakers, 1967, 1996 (abridged version is online) and visit david-kahn.com
- Google: History Cryptology/Encryption
- Dorothy E. Denning, Naval Postgraduate School, books and articles. dennin@nps.edu
- Bruce Schneier, www.schneier.com, textbook: Applied Cryptography, 1996; good blog

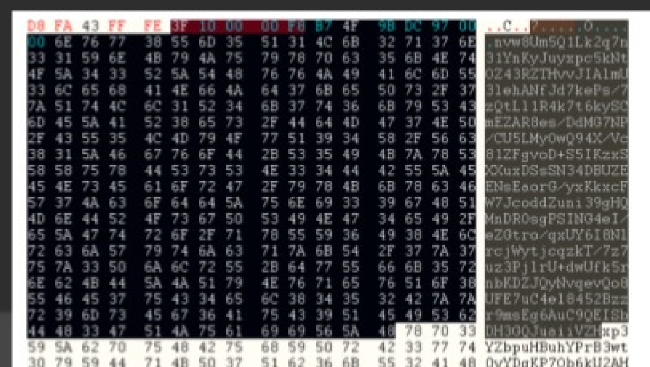
Steganography

- Hiding the message
  - Invisible ink, coded yarn, tatoos,...
  - Embedding in a picture, video, music, radio…
  - Many advanced techniques (Signal processing, coding theory, perception, ...)
- Steganalysis - finding the message
  - Google: John Ortiz
  - Youtube: stenanography
  - Same advanced techniques
  - Problem for Data Loss Prevention
  - Problem for inbound malware
  - Secrets of the Mujahideen

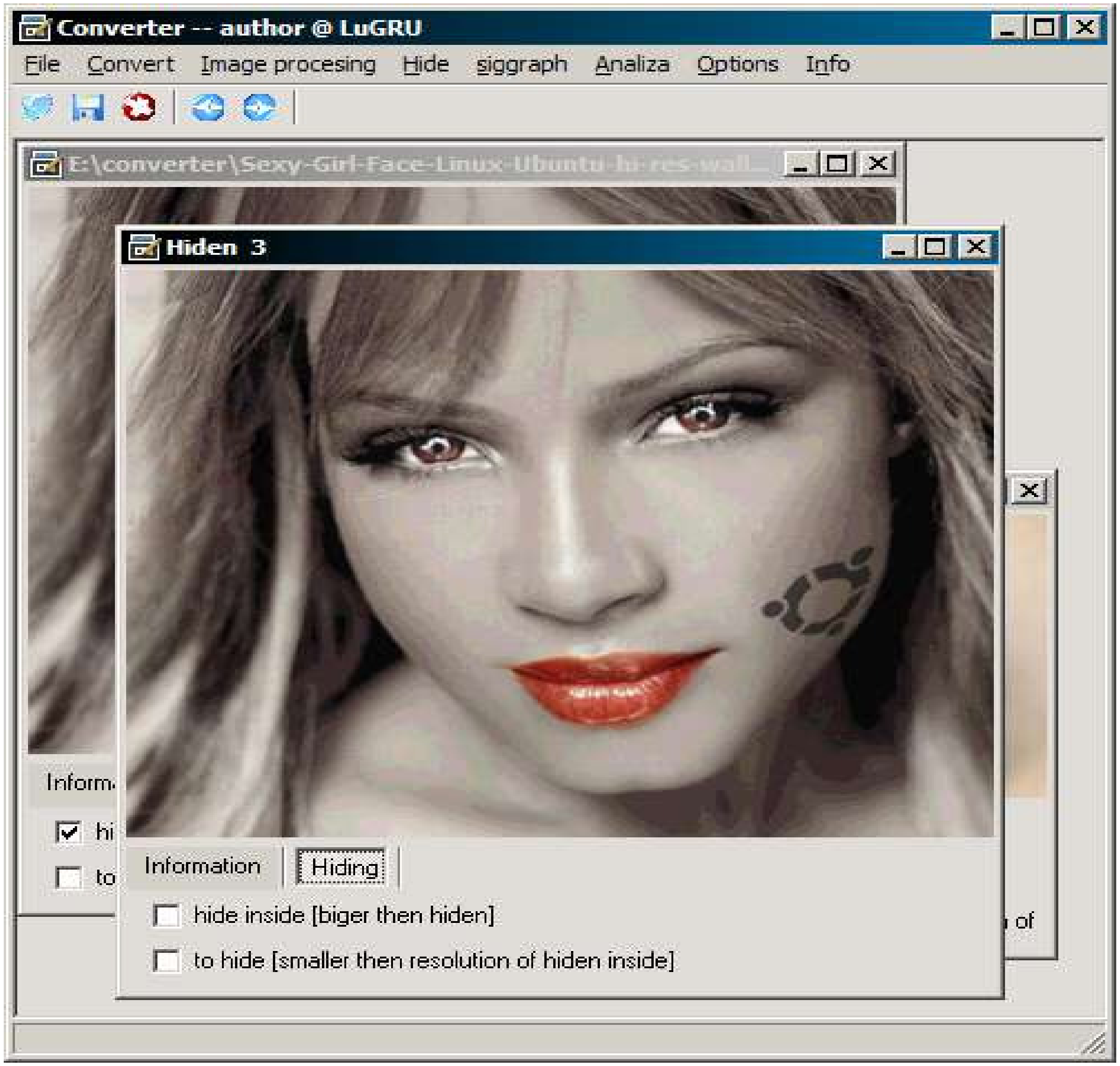
Zeus: Famous Malware

Image looks innocent

But it has appended encrypted data – Zeus config.



Lots of Tools

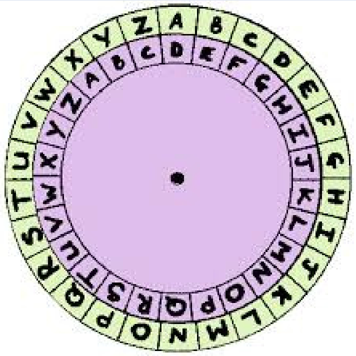


Network Cryptology

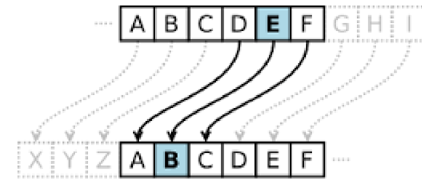
- Make open messages (in transit + in storage)
  - Private: make msg unreadable
  - Authentic: assure sender, receiver, data correct
  - Non-repudiated: sender can't deny sending
  - Other issues: leakage, replay, ...
- WARNING: Level of security of cryptology techniques is a future topic.

Symmetric
Shared Secret Key

- Let k be a shared secret key (Alice and Bob)
- Let M be a message space, C a cipher space
- Let c = E(k,m) be an encryption M $\rightarrow$ C
- Let m = D(k,c) be a decryption; D(k,E(k,m))=m
- Alice wants to send message m to Bob
  - Somehow they share key k; also E, D
  - Alice encrypts m and sends c = E(k,m)
  - Bob decrypts c to get m = D(k,c)



Shift Ciphers



- M = C = $(\text{ASCII})^n$ or $(\text{Unicode})^n$
- Code number wraps modulo N = 2^8 or 2^{16}.
- Key k in Z/N
- m = (m_i) encrypt to get E(k,m) = (c_i); $c_i = k+m_i$
- c = (c_i) decrypt: D(k,c) = (m_i); $m_i = -k+c_i$
- (Can use any regional 8-bit code for ASCII as well as subsets with smaller N)
- Exercise: what are keys if just shift A, B, …, Z ?

Substitution/Permutation Ciphers

- $M = C = (\text{ASCII})^n$ or $(\text{Unicode})^n$
- Key k is a permutation of (ASCII) or (Unicode)
- $m = (m_i)$ encrypt to get $E(k,m) = (c_i)$; $c_i = k(m_i)$
- $c = (c_i)$ decrypt: $D(k,c) = (m_i)$; $m_i = k^{-1}(c_i)$
- There are N! keys k; $N = 2^8$ or 2^{16}.

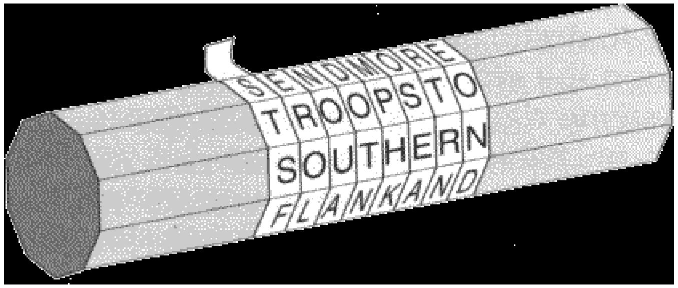
ADFGVX Substitution Ciphers

- ADFGVX chosen for distinct Morse Codes.
- RETREAT → XA DX FG XA FF FG
- 36! Keys (Permutations)

	A	D	F	G	V	X
A	Q	N	5	D	P	K
D	U	F	W	3	I	E
F	O	8	A	T	Y	6
G	2	L	1	V	C	S
V	B	X	M	7	H	9
X	R	4	G	0	J	Z

Rearrangement/Permutation Ciphers

- $M = C = (\text{ASCII})^n$ or $(\text{Unicode})^n$
- k is a permutation of $[0,n]$
- $m = (m_i)$ encrypt to get $E(k,m) = (c_i)$; $c_i = m_{k(i)}$
- $c = (c_i)$ decrypt: $D(k,c) = (m_i)$; $m_i = c_{j(i)}$ $j=k^{-1}$
- There are n! keys k, but usually simple permutations are used such as transpositions



Homophonic Ciphers

- M $\rightarrow$ random choice in a subset of C
- Typically take subset for letter *x* to be proportional to the frequency of *x*. The ciphertext will have a flat distribution.
- Example: letters $\rightarrow$ subsets of 0-99
  - E: 81 86 45 21 08 65 11
  - T: 23 15 48 95 64 01
  - Etc.

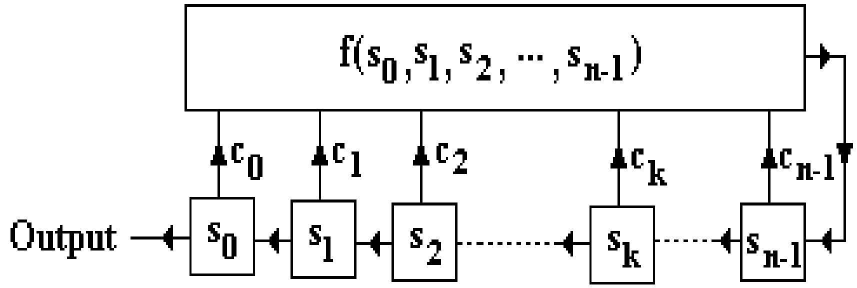
One Time Pad
(Vernam Ciper, AT&T, Patented 1917 Invented much earlier)

- Let $K = M = C = \{0,1\}^n$
- Define $E(k,m) = k$ <u>xor</u> m ; $D(k,c) = k$ <u>xor</u> c
- Number of keys k is $|K| = |M| = 2^n$

- If k is truly random, OTP is totally secure, [Shannon, '47?; Bell STJ papers '49, '51]
- Truly random? How about Pseudo-random?
- Red phone: DC and Moscow STILL???

(Linear) Feedback Shift Registers (LFSR)

- Need shift register of n bits $s_0, \ldots, s_{n-1}$
- Use s_0 as next pseudo-random bit, then
- Let f be (linear) polynomial function
- Set $s_i := s_{i+1}$ for $i<n-1$ and $s_{n-1} := f(s_0, \ldots, s_{n-1})$
- Can generate sequence of 2^n-1 bits
- Only need 2n values to predict all, if linear.



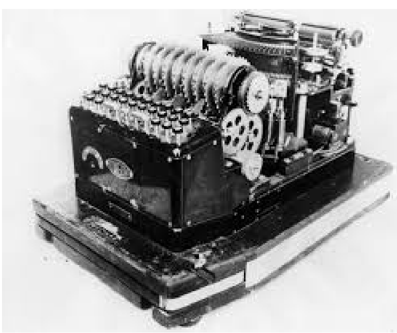
Multiple-Shift Ciphers

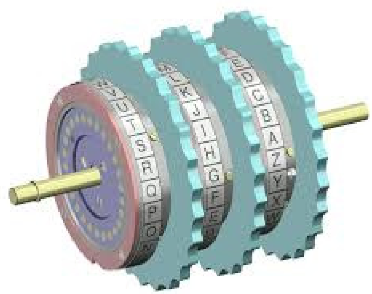
- Misattributed to Blaise de Vigenère
- $M = C = (\text{ASCII})^n$ or $(\text{Unicode})^n$
- Instead of one key k, use a sequence $k = (k_i)$
- $E(k,m) = c_i = m_i + k_i$ modulo $N = 2^8$ or 2^{16}.
- $D(k,c) = m_i = c_i - k_i$ modulo $N = 2^8$ or 2^{16}.
- Cycle k_i when key list is exhausted
- Encoding/decoding via mechanical disk/drum keyed to the sequence k.

Confederate Cipher Drum

Multiple-Permutation Ciphers

- Ditto, but k_i are permutations
- Enigma and Hagelin machines
  - commercial and military
  - Polish and British efforts: cracking machines
  - Books and movies … Story of Alan Turing





Stream and Block Ciphers

- Stream Cipher is typically bit, character, or word at a time
  - All previous examples are stream ciphers
- Block Cipher chunks up the message into fixed sized blocks, e.g. n = 64 or 128 bit blocks, and both E and D depend on n.
- Last block usually padded, e.g., with bits 1, 0,...0 so that each block has exactly n bits.

Stream Ciphers

- Small and fast. Many popular applications
- Synchronous and asynchronous
- Self-synchronizing ciphers
- Serious security problems historically
- Many more examples: RC4, A5/N (GSM), E0 (Bluetooth), PY, HC-128, Trivium, Grain, …
- Serious work, competitions, analysis, … Need smaller and faster for new comm devices.

Cryptographic Nonces

- Address the problem of replay: send $E(k,m)$ once and only once
- Generate non-repeating integer *nonces* n_i and define $E'(k,m) = E(k,n_i||m)$ if m is received with duplicate nonces, subsequent ones are rejected.
- Often time is encoded in a nonce

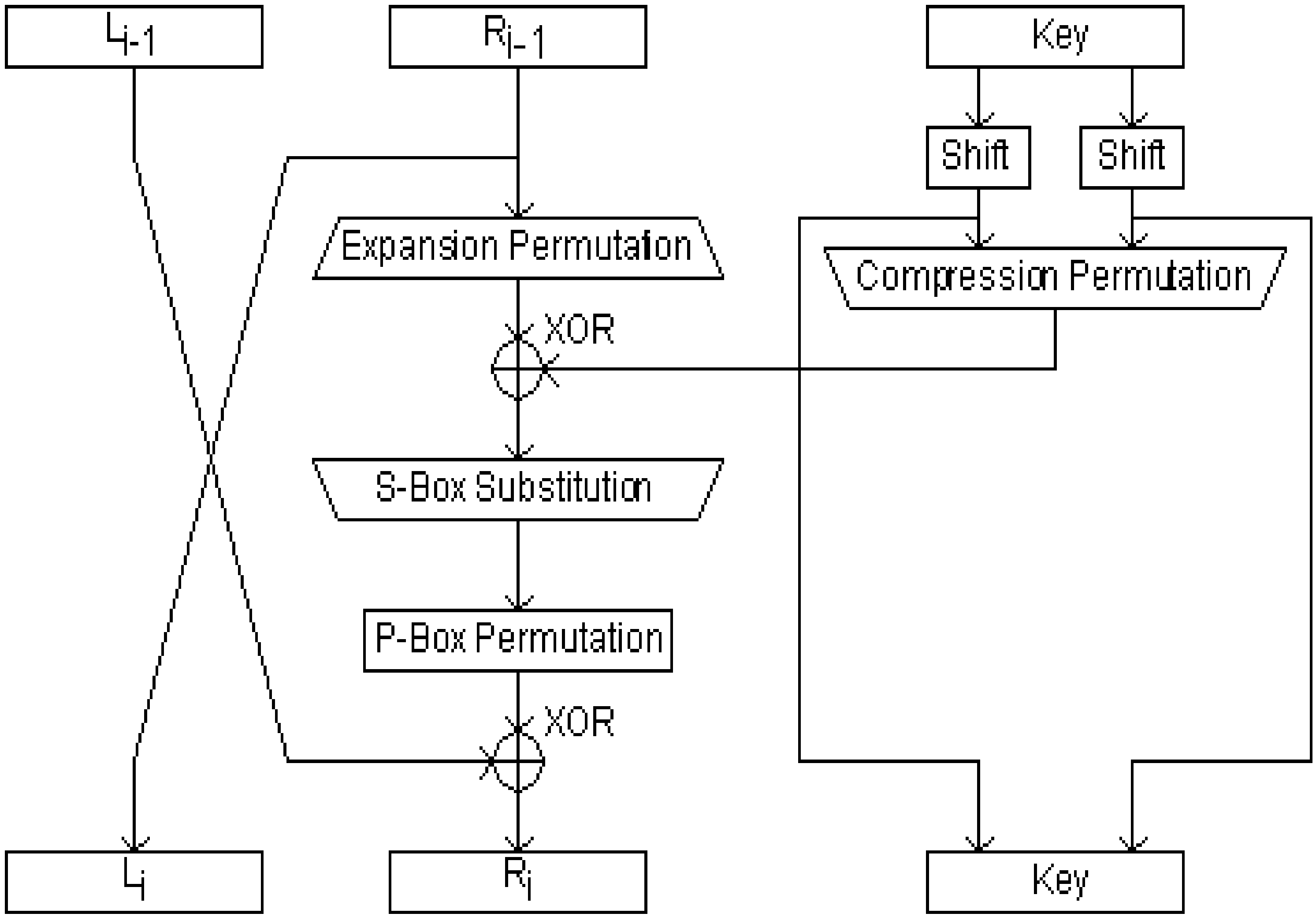
The WEP Saga 802.11

- 40 bit key + 24 bit IV = 64 bit RC4 key for confidentiality and CRC-32 for integrity.
- Key will repeat after some 5000 messages
- Easily cracked in a few minutes.
- Now WEP uses 256 bit keys, stronger...
- Many laptops are unsecured. TJ Maxx breach was result of WEP.
- Bluetooth, barcode readers, PDAs, wireless printers, etc. can be hacked.

Data Encryption Standard - DES

- NBS competition for commercial encryption, IBM (H. Feistel) “won”, 1976 FIPS standard, 64 bit blocks
- NSA forced 64 → 56 bit key – “easy” brute force attacks. Slow Triple DES extended life. Still used.
- Algorithm makes sixteen 48 bit subkeys k_i from key k. 16 rounds: take a 32 bit half block, expand it to 48 bits, <u>xor</u> k_i, divide into 8 parts, apply 8 non-linear (“S block”) lookups, permute.

DES



L_{i-1}
R_{i-1}
Key
Shift
Shift
Expansion Permutation
Compression Permutation
XOR
S-Box Substitution
P-Box Permutation
XOR
L_i
R_i
Key

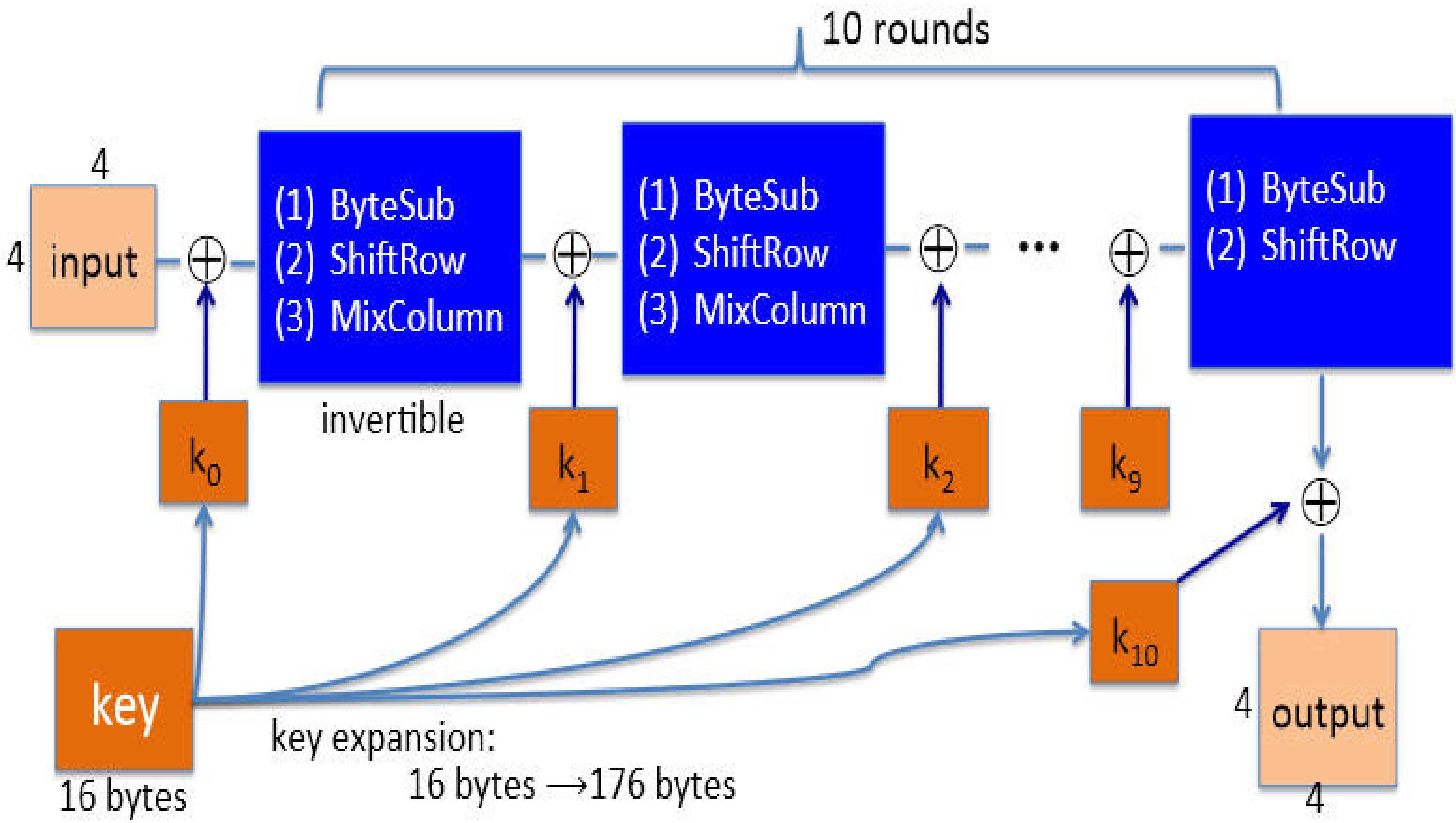
Advanced Encryption Standard – AES

FIPS 197 Replaced DES in 2001

Belgians Joan Daemen and Vincent Rijmen

- 128 bit block ciphers of key sizes 128, 192, and 256 bits which take (fast) substitution-permutation rounds of 10, 12, and 14 cycles.
- Code at aesencryption.net (asym, PHP, Java)
- As of 2014, there are some attacks that take less than key-size time, but no practical ones.

AES-128 schematic



Sharing Keys

- Usually, cryptography just assumes the encryption E and decryption D functions are known. The problem is how to share keys...
- No sharing is necessary with Public Key Encryption (PKE). Every individual has two keys. One private, secret key k_{Asec} that only the individual Alice knows, the other is public k_{Apub}, that Alice publishes on a public web site for all to see.

Asymmetric Public Key Encryption - PKE

- (G,E,D,K,K',M,C) is a PKE iff
  - Key Generator G: { } $\rightarrow$ K x K' where $G() = (k_{pub}, k_{priv})$
  - Encryption E: K x M $\rightarrow$ C
  - Decryption D: K' x C $\rightarrow$ M
  - $D(k_{priv}, E(k_{pub}, m)) = m$
- Each user of the (G,E,D) PKE gets a pair of keys from G. The keys k_{pub} and the functions E and D are made public.
- Philosophy: to find k_{priv} from k_{pub}, must solve a hard problem taking unfeasible compute power.

(Textbook) RSA

(Rivest, Shamir, Adleman, 1978)

- Hard problem: factor large n into primes.
- Choose large primes p and q of similar size, and set $n = pq$ (keep $\phi(n)$, p and q secret) where $\phi(n) = (p-1)(q-1) = |Z/n^*|$. For G: pick e in $Z/\phi(n)^*$ and compute $d = e^{-1}$.
  Then $k_{pub} = (n,e)$ and $k_{priv} = (n,d)$.
- For message m in Z/n, define $E(k,m) = m^e$ and $D(k,m) = m^d \bmod n$.
- Theorem. $m^{ed} = m \bmod n$

Homework: Why RSA works

- Since $ed = 1 \bmod \phi(n)$, $ed = 1+k(p-1)(q-1)$
- In Z/n, $D(d,c) = c^d = m^{ed} = m^{1+k(p-1)(q-1)} = m(m^{\phi(n)})^k = m$ if m is invertible in Z/n; if not, then $\gcd(m,n) > 1$ is a factor of n, say $m = rp$. Then $m^{1+k(p-1)(q-1)} = rp((rp)^{p-1})^{k(q-1)} = rp \bmod q$. Hence both m and $m^{1+k(p-1)(q-1)} = 0 \bmod p$ and $= rp \bmod q$. By CRT they are equal mod $pq = n$.
- Hard to compute d from e: one must know $\phi(n) = (p-1)(q-1)$. In which case, $p+q = n-\phi(n)+1$ and $p-q = \mathrm{sqrt}((p+q)^2-4n)$ and $p = (p+q)/2 + (p-q)/2$ and $q = (p+q)/2 - (p-q)/2$. Thus knowing n and $\phi(n)$ yields the factors p and q.

Beware for RSA

- Primes p, q are “safe” iff p-1 and q-1 have large prime factors (Z/n will have large cyclic subgroups.)
- Primes p and q cannot have same number of digits; else, search for p,q starting at sqrt(n)
- Public key e cannot be too small
- Stop using 1024 bit RSA, quadratic and number-field sieves are effective. 2048 is slow. ECC better.
- Always pad message m to get m' (more on this later)
- Use well-tested, well-analyzed implementation

Padding RSA

- Problems with textbook RSA
  - (Malleable) if $c = m^e$ and $c' = c*2^e$, decrsypting c' gives 2m. i.e. can make predictable changes to ciphertexts.
  - (Deterministic = not semantically secure) can distinguish between plain text m and m' by encrypting both with public key.
- Basic idea is to pad m with random bits r and encrypt m||r to get c. Decrypt c to get m||r and hence m. Neither Malleable nor Deterministic.

Optimal Asymmetric Encryption Padding (Wikipedia: OAEP)

Given, n = modulus of RSA, k0 fixed integer, G expands k0 bits to n-k0 bits, H reduces n-k0 bits to k0 bits.

- pad m with k1 zeroes to be m' of n-k0 bits
- Pick random k0 bit string r
- X = m' <u>XOR</u> G(r), Y = r <u>XOR</u> H(X)
- Encrypt X || Y to get c; decrypt c to get X || Y
- Recover r = Y <u>XOR</u> H(X), m' = X <u>XOR</u> G(r)
- Strip k1 zeroes off m' to get m

El Gamal
(Avoided RSA Patent)

- Hard problem: compute discrete logs mod p for large prime p, i.e. solve $y=g^x$ for x mod p
- Choose large p and generator g of Z/p*
- G: pick random d in Z/p*, compute $e = g^d$.
  Then $k_{pub} = e$ and $k_{priv} = d$.
- To encrypt m in Z/p, choose random (secret) integer k and compute $r = g^k$ and $t = e^k m$; discard k. E(e,m) = (r,t) and D(d,c=(r,t)) = $t*r^{-d}$. Exercise: D(d,E(e,m)) = m.
- Choose a different k for every (block) m.

Homework: Why El Gamal works

- $D(d,E(e,m)) = D(d,(g^k,e^k m)) = e^k m(g^k)^{-d} = g^{dk} m(g^k)^{-d} = m$
- Exercise: $D(e,E(d,m)) = m$
- Hard: to discover d from e, one must solve $e = g^d$ for $d = \log_g(e)$. This is the discrete log problem.
- BEWARE: if same k is used for two blocks m and m', then m' can be recovered from m.

Diffie-Hellman

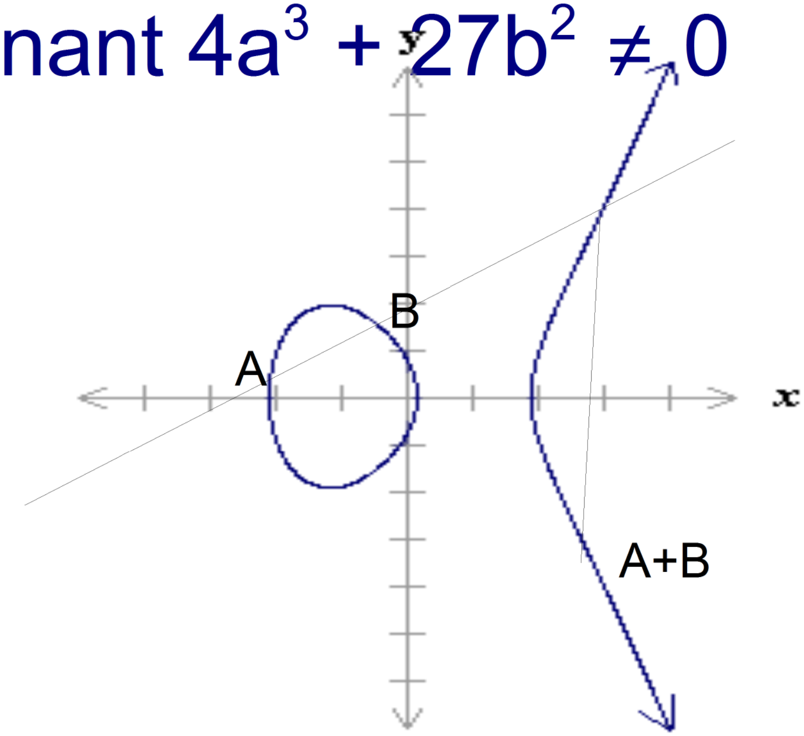
- Pick a large prime p of 600 digits ~ 2000 bits
- Pick a finite cyclic group G = (g) of order n
- G could be Z/p* or an elliptic curve of char p
- Alice chooses random secret a in Z/n and sends $A = g^a$ to Bob
- Bob chooses random secret b in Z/n and sends $B = g^b$ to Alice
- $A^b = B^a = g^{ab}$ is a shared secret key in G.

Session Keys

- Suppose G() = (kpub,kpriv) for E, D. Let k_{Apub} and k_{Apriv} be public and private keys for Alice.
- For Bob to share a secret key k with Alice, he just encrypts k with k_{Apub} and sends the result $c = E(k_{Apub}, k)$ to Alice who can retrieve $k = D(k_{Apriv}, c)$ using her private key.
- Session keys used by many network protocols

Elliptic Curves

- Weirstrauss eqn $y^2 = x^3 + ax + b$ where the discriminant $4a^3 + 27b^2 \neq 0$



$y^2 = x^3 - 4x + 0.67$

Points on an Elliptic Curve

- Write down equations for A+B, and get a finite abelian group **E**(F) (assoc law tedious) over finite field F.
- Elliptic Discrete Logs: given Y = rX find r .
- Choices are made to improve performance and difficulty of EDL problem. Also need a (public) message embedding i:{m} $\rightarrow$ **E**(F) or a way to use only the x-coordinates.

Elliptic El Gamal

- For elliptic curve **E** over F, pick a "base point" G with (G) = **E**(F) with i:{m} → **E**(F)
- A private key is a random integer a; compute public A = aG. For a message m, pick random integer k and
  - Encrypt E(A,m) = (kG, kA+i(m)).
  - Decrypt by D(a,(R,T)) = -aR+T
- D(a,E(A,m)) = D(a,(kG,kA+i(m))) = -akG+kA+i(m) = -kA + kA + i(m) = i(m)

Choosing Fields and Equations for Elliptic Encryption

- Focus on $F = F_q$ where $q = 2^m$ or q = large p; there are q distinct elliptic curves over F_q.
- For $q=2^m$, **E**: $y^2 + xy = x^3 + ax + b$, $4a^3 + 27b^2 \neq 0$
- |F| and |Curve| need to be large. Eqn needs to be simple for easy computation. The base point (generator) G is chosen so that its multiples rG are easy to compute.
- NIST has recommendations (FIPS 186), but there is a fog of suspicion (NY Times, 2013, and multiple other recent papers) due to NSA involvement. Non-NIST curves are gaining popularity Cf. Bernstein and Lange: http://safecurves.cr.yp.to

Bernstein's Curve25519

- Dan Bernstein: lucid paper on encryption performance and security with Curve25519
- $p = 2^{255} - 19$, $F = F_p = Z/p$, $g = 9$
- $y^2 = x^3 + 486662x^2 + x$ (Montgomery form)
- Keys are 32 byte x-coordinates via map $\mathbf{E} \to F$
- Generates 32 byte shared secret key
- Uses floating point registers for fast arithmetic
- Many applications today use Curve25519

Cryptographic Hash Functions

- H:Data → Values where |Values| << |Data|
  - (a) Easy to compute; use entire data/message
  - (b) Infeasible to invert (to find preimage)
  - (c) Infeasible to modify w/o (large) value change (to find 2nd preimage)
  - (d) Infeasible to find collisions
  - (e) Given H(m), H(m'), cannot compute H(m||m')
- If |Values| = 2^n then want Prob(b) = Prob(c) = $1/2^n$ and Prob(d) = $1/2^{n/2}$. "Security" = n/2.
- **Data → Blocks → State $\xrightarrow{f}$ ... $\xrightarrow{f}$ State → Output**

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Rounds	Operations	Security (bits)	Example Performance (MiB/s)[28]
MD5 (as reference)		128	128 (4× 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or	<64 (collisions found)	335
SHA-0		160	160 (5× 32)	512	$2^{64} - 1$	80		<80 (collisions found)	-
SHA-1		160	160 (5× 32)	512	$2^{64} - 1$	80		<80 (theoretical attack[29] in 2^{61})	192
SHA-2	*SHA-224* *SHA-256*	224 256	256 (8× 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112 128	139
	SHA-384 *SHA-512* *SHA-512/224* *SHA-512/256*	384 512 224 256	512 (8× 64)	1024	$2^{128} - 1$	80		192 256 112 128	154
	SHA3-224 *SHA3-256*	224 256		1152 1088				112 128	

Avalanche Effect

Using RHash implementation (not official)

SHA3-256("The quick brown fox jumps over the lazy dog")=

0x
69070dda01975c8c120c3aada1b282394e7f032fa9cf32f4cb
2259a0897dfc04

SHA3-256("The quick brown fox jumps over the lazy dog.")=

0x
a80f839cd4f83f6c3dafc87feae470045e4eb0d366397d5c6ce
34ba1739f734d

Hash Applications

- File/message integrity: publish hash value, recompute it after file/message transfer. “Message Authentication Code” = MAC = hash value
- Password storage: only store the hash value (usually store (salt,H(salt||password)) to avoid knowing Alice and Bob have the same pswd or precomputing H(common words).)
- Digital signatures (analog of ink): if k is a shared secret key for (E,D) then S(k,m) = E(k,H(m)) is a signature, and can send (m,S(k,m)) in the clear.
  - Has the usual key sharing problem
  - How about using public key encryption?

Digital (Public Key) Signatures

- Want authentication and non-repudiation: If Alice provides a signature, verify authentic, and prove she cannot later deny that it is hers.
- Scheme-type hard problems
  - Integer factorizations (RSA, Rabin)
  - Discrete Logarithms (El Gamal, Schnorr, DSA, Nyberg-Rueppel)
  - Elliptic Curves (ECDSA)

RSA Signatures

- Pick large primes p and q with $n = pq$. Pick $ed=1$ in $Z/\phi(n)^*$ where $\phi(n) = (p-1)(q-1) = |Z/n^*|$
- d is private key, e is public key.
- To sign m in Z/n, compute $h = H(m)$, then $s=h^d \bmod n$ is the signature. Verify $s^e=h$ in Z/n.
- Authentication: $s^e = h^{ed} = h^{1+k\phi(n)} = h$ (exercise)
- Non-repudiation: only holder of d could have created s

El Gamal Signatures

- Let p be a large prime, g a generator of Z/p*
- Alice's private key d with $1 < d < p-1$. $e=g^d$ is the public key. Note p, g, e, and hash fcn H are public.
- To sign m in Z/p, pick random k, $1 < k < p-1$, $\gcd(k,p-1)=1$. Compute $h = H(m)$, $r = g^k$, and $s = (h-dr)k^{-1} \bmod p-1$. If $s = 0$, pick a new k. (r,s) is the signature.
- Accept (r,s) if $0<r<p$ & $0<s<p-1$ & $g^h = e^r r^s \bmod p$
- If e,d are Alice's keys, then $e=g^d$ and $r=g^k$, hence $g^h = g^{ks}g^{dr}g^{t(p-1)} = e^r r^s$ since $g^{p-1} = 1 \bmod p$
- Given $g^h = e^r r^s \bmod p$, is s Alice's signature?

Schnorr Signatures
(Patent expired in 2008)

- Let $G = (g)$ have prime order q, e.g. G a subgroup of Z/p^*, let H be a crypto hash fcn. Let $1<d<p-1$ be the private key, $e = g^d$ the public key. To sign a finite bit string message m, choose a random k, $1<k<p-1$ and let $r = g^k$ be represented as a bit string. Let $h = H(m||r)$. Let $s = k-hd \bmod p-1$. The signature is (s,h). Since $r = g^{s+hd+t(p-1)} = g^s e^h$ in Z/p, $h = H(m||g^s e^h)$
- Accept (s,h) if $h = H(m||g^s e^h)$
- Nice: with Schnorr, no inversions are necessary to compute or verify the signature (s,h)

“The” Digital Signature Algorithm DSA (Your tax dollars at work)

- Now FIPS 186-4, with H = SHA 1 or 2.
- Choose an N bit prime q. N<outputsize(H)
- Choose an L bit prime p: p-1=mq.
- Choose g in Z/p of order q, e.g. $g = h^{(p-1)/q}$
- Now apply El Gamal with (p,q,g)

ECDSA – sign

(Additive El Gamal)

- Elliptic **E**, G base point of prime order n, d_A in Z/n is Alice's private key, $Q_A = d_A G$ her public key, cryptographic hash H . To sign message m in Z/n:

  1. Select random k in Z/n*, different for all signatures
  2. Calculate $(x_1,y_1) = kG$; convert x_1 to an integer $\bar{x}_1$
  3. Calculate $r = \bar{x}_1 \bmod n$. If $r = 0 \bmod n$, goto 1
  4. Calculate $e = H(m)$. If $e+rd_A = 0 \bmod n$, goto 1
  5. Calculate $s = k^{-1}(e+rd_A)$ in Z/n*
  6. Output (r,s) as the signature

ECDSA - verify

- Assume Bob has certified copy of Alice's credentials, e and m.
- Verify signature (r,s)
  - Validate r and s are in Z/n*
  - Calculate $w = s^{-1}$, $u_1 = ew$, $u_2 = rw \bmod n$
  - Calculate $C = (x_2, y_2) = u_1 G + u_2 Q_A$
  - If C = O, reject signature
  - Convert x_2 to an integer $\overline{x}_2$ mod n
  - Signature valid iff $r = \overline{x}_2 \bmod n$

ECDSA - proof

- Why does verification work?
- If signature (r,s) was computed by Alice, then $Q_A = d_A G$, $r = \bar{x}_1 \bmod n$ where $(x_1,y_1) = kG$ for k in Z/n*, and $s = k^{-1}(e+rd_A)$ in Z/n* where e = H(m). Write $C = (x_2,y_2) = u_1 G + u_2 Q_A$ where $u_1 = es^{-1}$ and $u_2 = rs^{-1} \bmod n$. Thus $C = (es^{-1})G + (rs^{-1}d_A)G = (e+rd_A)s^{-1}G = (e+rd_A)k(e+rd_A)^{-1}G = kG = (x_1,y_1)$, and hence $r = \bar{x}_1 = \bar{x}_2 \bmod n$
- Conversely, suppose Bob receives (r,s) as a signature. He computes $C = (x_2,y_2) = u_1 G + u_2 Q_A$ where $u_1 = es^{-1}$, $u_2 = rs^{-1} \bmod n$, and e = H(m). Bob verifies that $r = \bar{x}_2 \bmod n$. Write C = kG. We know $Q_A = d_A G$. Thus $kG = C = (es^{-1})G + rd_A s^{-1}G = (e+rd_A)s^{-1}G$. Thus $k = (e+rd_A)s^{-1}$ in Z/n, and $s = (e+rd_A)/k$. In other words, r and s are determined, and the signature (r,s) must have been created using Alice's private key d_A .

Sony Playstation3 ECDSA Hack
Repeating use of k

- Given (r,s) and (r,s') for messages m and m', with hashs e and e'; if same k, note that
- $s-s' = k^{-1}(e-e') \bmod n$, so $k = (e-e')/(s-s')$ and one can solve $s = k^{-1}(e+rd_A)$ for Alice's private key d_A.

Ref: Console Hacking 2010

Certificates
Authentication, Public Keys, etc

- Certificate Contents
  - Certification Authority – CA
  - Root CA – certifies its own keys!
  - Certificate Owner
  - Expiration Date
  - Owner's Public Key
  - Certificate serial number
  - Other identifying info
  - Digital Signature(s).

Secure Socket Layer, SSL 2,3 → Transport Security Layer, TSL 3.1,...

- Secure TCP connection = Key exchange method, encryption algorithm, and content authentication hash algo
- Handshake:
  - client hello: cipher proposal, 32 random bytes
  - server hello: select cipher, 32 random bytes, certificate, hello done
  - client key exchange: 48 byte secret encrypted with server public key, change to cipher msg
  - Server change to cipher msg, finished record encrypted and MAC'd
- For some applications, server may request client certificate
- Record Processing: cuts msg into blocks, opt. compresses, hashes, encrypts block, sends to Transport Layer

HTTPS

- HTTPS requires SSL/TLS to be used
- Some overhead, often accelerated with hw
- No client certificates.
- Marking cookies “secure” tells browser to only send cookie data, e.g. session Ids, via SSL/TLS. (Cookies should also be marked “HTTPonly” to inhibit javascript client-side attacks.)

Recommended Key Lengths

- Need longer and longer keys over time
  - Hardware improvements
  - Algorithm improvements
- Ask how long your encryption should last! 50 years is reasonable....
- There are legal issues around both time and key storage. Don't lose your keys!!!
- NIST, ANSSI, BSI, NSA publish recom-mendations; also check www.keylength.com

What to use and trust?

- OTR: Off the Record messaging
- Tor
- StartPage – privacy browser
- Tails – a live OS that can boot from an external drive. Used to preserve privacy.
- GPG, GPG4Win (Gnu freeware impl of OpenPGP)
- TrueCrypt – might be back online; does disk encryption
- MiniLock email uses Curve25519
- File Erasure – PGP does only one overwrite
- Air Gapped Computers – transfer via USB still tricky
- SSL/TLS??? OpenSSL? Not BGP due to router infections.
- Sage – open source math tool

Final Thoughts

- Cryptography is only the non-people part of security.
- Known attacks prove future attacks will become more sophisticated and widespread with many actors.
- While credit card and IP theft is on the rise, a wave of ICS cyber-terrorism (stuxnet-style) has yet to hit big. We are not prepared for either.
- The economics of security will soon change as the cost of cyber-crime is fairly allocated.
- Encryption is hard to implement correctly, and Crypt-analysis is only in its infancy. Cryptography should be taught to undergraduate engineers. It is basic math and basic engineering.
- Backup your systems offline to protect from ransomware