CSCI 1800 Cybersecurity and International Relations

Secure Communication and Authentication

John E. Savage
Brown University

Outline

- Symmetric Cryptography
- Public-Key Cryptography
- Cryptographic Hash Functions
- Digital Signatures
- Diffie-Hellman Key Exchange

The Cryptographic Problem

- Goal: Alice needs to communicate securely with Bob, but Eve listens or interferes with conversation.
- Approach: Alice and Bob encrypt messages (they create ciphertexts) to keep them secure from Eve.
- Eve engages in cryptanalysis, tries to break cipher.
- Security by obscurity is dangerous. Once obscure method is discovered, all secrets are lost.
- Better to assume encryption method is known but that keys remain secret. Keys can be changed.

Three Types of Notation

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Decimal	Binary	Octal
0	000 000	0 0
1	000 001	0 1
2	000 010	0 2
3	000 011	0 3
4	000 100	0 4
5	000 101	0 5
6	000 110	0 6
7	000 111	0 7
8	001 000	1 0
9	001 001	1 1
10	001 010	1 2
11	001 011	1 3
12	001 100	1 4
13	001 101	1 5
14	001 110	1 6
15	001 111	1 7
16	010 000	2 0

American Standard Code for Information Interchange (ASCII)

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

ASCII notation is an octal triplet

Message Fragment in Binary

- Map message: no mon no fun to ASCII
- n 156 o 157 (space) 040
- 001 101 111 001 101 111 000 100 000
- m 155 o 157 n 156 (space) 040
- 001 101 101 001 101 111 001 101 111 000 100 000
- n 156 o 157 (space) 040
- 001 101 110 001 101 111 000 010 000
- f 146 u 165 n 156
- 001 010 110 001 110 101 001 101 111
- Concatenate bits to form integer message M = 0011011…

Symmetric Cryptography

- They agree on a common encryption method.
- Both Alice and Bob have the same secret key.
- Convert a text message to an integer M.
  - Example: no mon no fun
  - \156\157\040\155\157\156\040\156\157\040\146\165\156
    - Slashes between octal triplets are for humans only
  - M = 001 101 110 001 101 111 000 100 111 …
- Encrypt M as $C = E_K(M)$ using function E and key K.
- Decrypt C same way, $M = E_K(C)$. K is secret. Symmetric!

Eve Attempts to Get Secret Key

- Ciphertext-only attack (least info)
  - Eve only has ciphertext.
- Known-plaintext attack
  - Eve is given plaintext-ciphertext pair(s).
- Chosen-plaintext attack
  - Eve chooses plaintext(s), gets ciphertext(s). She may choose plaintexts adaptively.
- Chosen-ciphertext attack (most info)
  - Eve chooses ciphertext, gets plaintext.

Decreasing difficulty

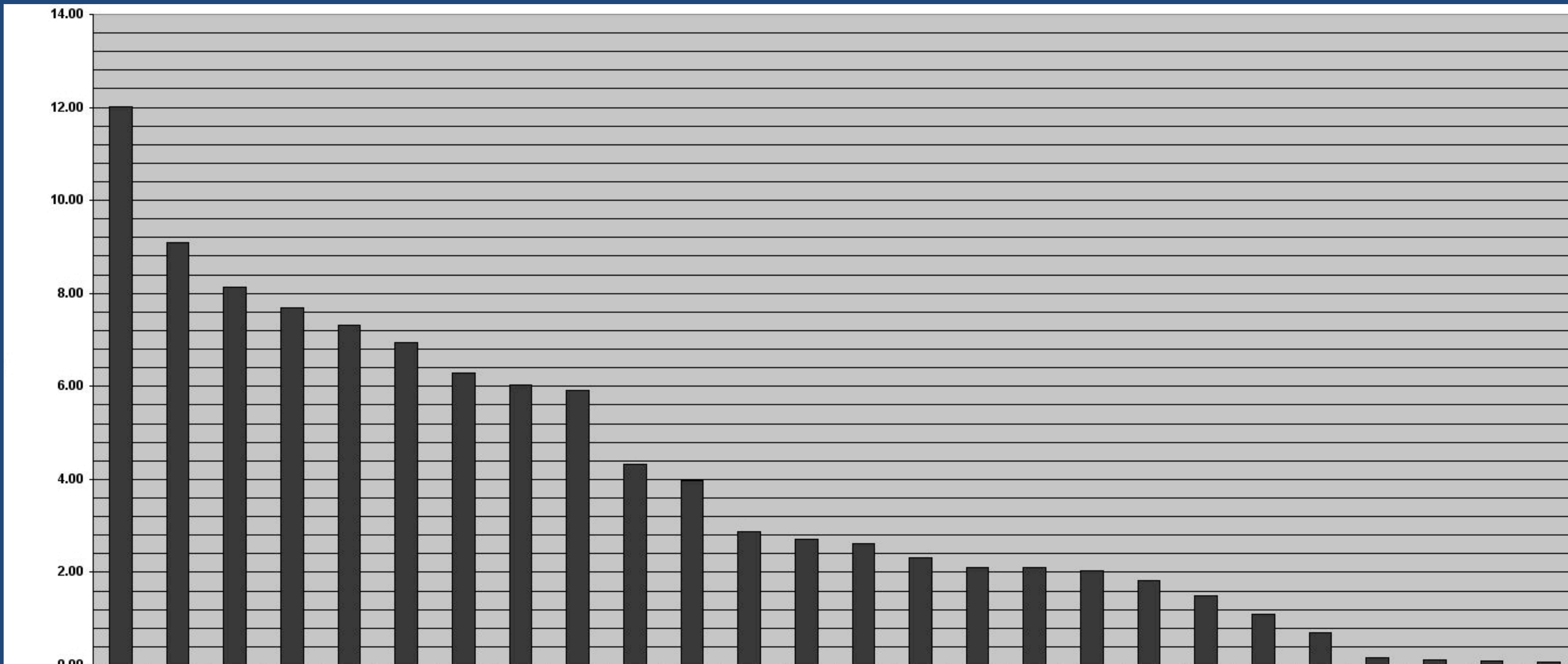
Ciphers Introduced in Today's Lecture

- Substitution ciphers
- Polygraphic substitution ciphers
- One-time pads
- Binary one-time pads
- Advanced encryption standard (AES)
- Public-key cryptography (RSA)
- Digital signatures and hash functions

Substitution Ciphers

- Substitution ciphers permute letters in alphabet
  - E.g. Caesar replaced a letter by one three places away in the Latin alphabet.
  - Caesar(3): a b c d … x y z is replaced by d e f g … a b c
- General substitution cipher – map letters in an alphabet to a fixed permutation of the alphabet.

Frequency of Letters in English



	E	T	A	O	I	N	S	R	H	D	L	U	C	M	F	Y	W	G	P	B	V	K	X	Q	J	Z
Frequency	12.02	9.10	8.12	7.68	7.31	6.95	6.28	6.02	5.92	4.32	3.98	2.88	2.71	2.61	2.30	2.11	2.09	2.03	1.82	1.49	1.11	0.69	0.17	0.11	0.10	0.07

Letter

Breaking Substitution Ciphers

- Substitution ciphers are easily broken
- Compute the frequency of each letter
  - Find the most frequent letter, let's call it α.
  - Almost certainly e maps to α with frequency ~12%
  - Find the second most frequent letter, β.
  - Almost certainly t maps to β with freq. ~ 9%
- Check words that result and fix mapping.

Vigenère Cipher

- Vigenère cipher (1586) is a polygraphic cipher on blocks of m letters. Given m letters $(l_1, l_2, \ldots, l_m)$, l_j is shifted cyclically by k_j places for $0 \le k_j \le 25$.
  - If m = 3, $k_1 = 2$, $k_2 = 1$, $k_3 = 3$, (a,g,z) mapped to (c,h,c).
  - Let's encrypt attackatdawn
  - (a,t,t)(a,c,k)(a,t,d)(a,w,n) ➡ (c,u,w)(c,d,n)(c,u,g)(c,x,p)
  - Encrypted message is cuwcdncugcxp
  - If m is reasonably small, easily broken by statistics.

Vigenère Cipher

- If m is reasonably small, the Vigenère cipher is easily broken by statistics.
  - How would you do that?
- The integers can be derived from a text string
  - thequickbrownfoxjumpsoverthelazydog
  - Start alphabet at 0; a $\leftrightarrow 0$, b $\leftrightarrow 1$, …, t $\leftrightarrow 19$, …, z $\leftrightarrow 25$,
  - 19 7 4 16 20 8 2 10 1 17 14 22 13 5 14 23 9 20 12 15 18 14 21 4 17 19 7 11 0 3 14 6
  - Does this look like a random string?
    - How many times are digits repeated?

One-Time Pad

- One-time pad (Miller 1882) uses m random integers $\{k_j \mid 1 \le j \le m\}$, $0 \le k_j \le 25$, to shift letters in a string of length $\le m$.
  - The j^{th} letter is shifted by k_j positions.
    - A real one-time pad might have edible pages of digits.
  - Both sender and receiver need to know shifts
  - Provides perfect security when m ≥ message length
  - Fails when pad is reused or string is longer than m.
    - One-time pad encryption broken during Cold War.

Binary One-Time Pad Again

- Message represented as n-bit binary string.
  - E.g. $\underline{M}$ = 010011 (a vector)
- Generate random n-bit string K (the key or one-time pad)
  - E.g. $\underline{K}$ = 100110 (a vector)
- XOR (⊕) is defined as $1 \oplus 0 = 0 \oplus 1 = 1$ and $0 \oplus 0 = 1 \oplus 1 = 0$
- XOR message $\underline{M}$ with key $\underline{K}$ bit-by-bit to encrypt as X.

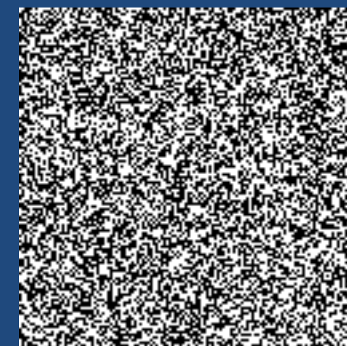
$$\underline{X} = E_{\underline{K}}(\underline{M}) = \underline{M} \oplus \underline{K}$$

  - E.g. $E_{\underline{K}}(\underline{M}) = (0 \oplus 1)\ (1 \oplus 0)\ (0 \oplus 0)\ (0 \oplus 1)\ (1 \oplus 1)\ (1 \oplus 0) = 110101$
- Decrypt by encrypting $\underline{X}$ with $\underline{K}$

$$E_{\underline{K}}(\underline{X}) = \underline{X} \oplus \underline{K} = (\underline{M} \oplus \underline{K}) \oplus \underline{K} = \underline{M} \oplus (\underline{K} \oplus \underline{K}) = \underline{M} \oplus \underline{0} = \underline{M}$$

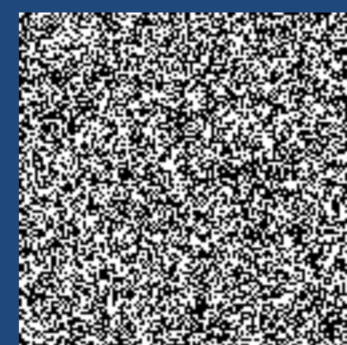
Reuse of One-Time Pad Dangerous

=



$\underline{C_1}$

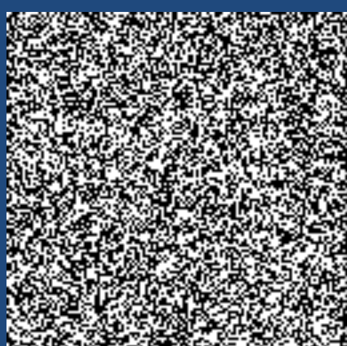
=



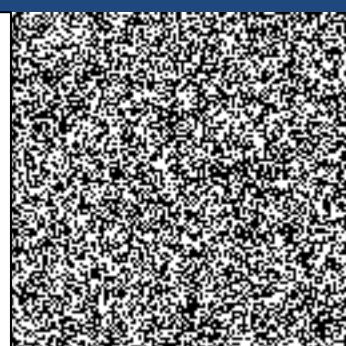
$\underline{C_2}$

XORing Two Encrypted Images

$\underline{C}_1 = \underline{K} \oplus \underline{M}_1$ $\underline{C}_2 = \underline{K} \oplus \underline{M}_2$ $\underline{C}_1 \oplus \underline{C}_2 = \underline{M}_1 \oplus \underline{M}_2$



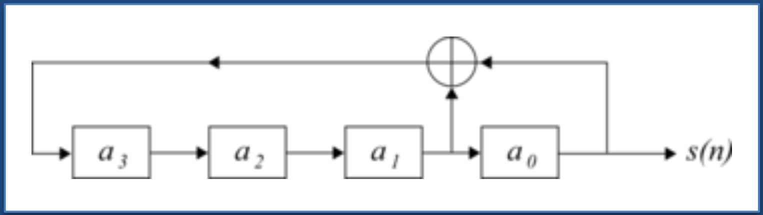
$\oplus$



=

Pseudo-Random Number Generators

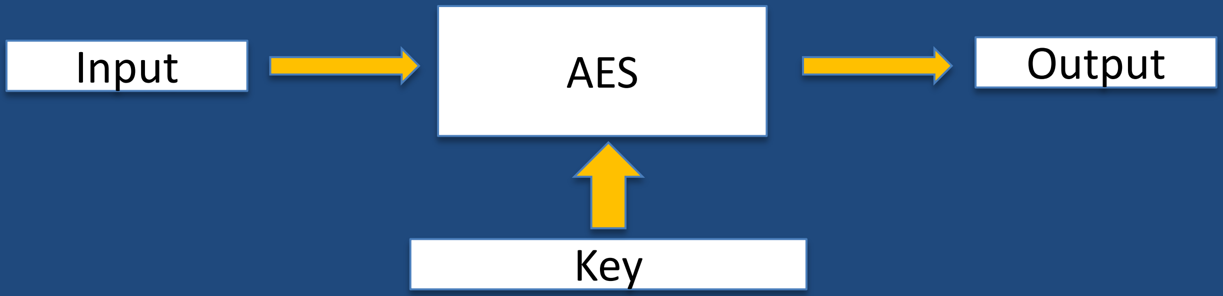
- It is expensive to produce true random nos.
- Pseudo-random number generators (PRNGs) generate numbers that "look" random.



- Encryption algorithms can be used as PRNGs.
  – Encrypt a fixed string and represent it in binary
  – E.g. E(attackatdawn) = 0100110101001110110

Advanced Encryption Standard (AES) (Rough Sketch)

- AES (circa 2001) is a symmetric cipher whose inputs and outputs are 128-bit blocks. It uses an encryption key K of length 128, 192 or 256 bits, denoted AES-128, AES-192, AES-256.



128, 192 or 256 bits

Advanced Encryption Standard (AES)

- When $\underline{K}$ has 128 bits, AES computes $\underline{X}_0 = \underline{M} \oplus \underline{K}$ and then executes 10 rounds.
  - Each round does a substitution, permutation, mixing of results, and an XOR'ing step.
  - It is too complicated to explain here.
- AES is highly secure but can be attacked using the time spent computing – this is a side channel attack
- In 2010 AES-256 was considered highly secure.
- AES-192 and AES-256 approved for US Top Secret!

Public-Key Cryptography

- Each party has public & private keys
  - Alice: $Priv_{Alice}$, Pub_{Alice}; Bob: $Priv_{Bob}$, Pub_{Bob}.
- Alice encrypts message M for Bob with

$$X = E_K(M) \text{ where } K = Pub_{Bob}.$$

- Bob decrypts Alice's encrypted message with

$$M = E_{K*}(X) \text{ where } K^* = Priv_{Bob}.$$

- Decrypt using same algorithm E with private key

Origin of Public-Key Cryptography

- James Ellis, Clifford Cocks, Malcolm Williamson, invented it at GCHQ (British intelligence agency) by 1973, made public in 1997
- Diffie and Hellman propose idea publicly in ’76.
- Rivest, Shamir and Adleman (RSA) gave first practical implementation in 1977.

* http://en.wikipedia.org/wiki/Public-key_cryptography

Symmetric vs Public Key Crypto

- Symmetric key system has one key per user pair
  - Thus, there are n(n-1)/2 (pairs) keys for n users
  - If $n = 10^4$, that's about 50×10^6 keys!
- In public-key system, 2n keys suffice.
  - Each party publishes one key, keeps other secret
- Symmetric key system faster than public key.
  - PK systems often used to create/exchange secret symmetric keys

RSA Public-Key System

- Modular arithmetic
  - add and multiply integers modulo n
  - result is the remainder after dividing by n.
  - E.g. (3+4) mod 5 = 2, (4*3) mod 3 = 0
- Bob's public key Pub_B is the integer pair (e,n).
- Bob's secret key is $Priv_B$ = d. n = pq, two primes
- Require that e, d, and n satisfy

  X^{de} mod n = X for any integer X in {0,1,2,…n-1},

RSA Public-Key System

- Alice encrypts M for Bob as $C = M^e \bmod n$
  - Recall $Pub_B = (e,n)$
- Bob decrypts C by computing $C^d \bmod n = M$. This follows because

  $$C^d \bmod n = (M^e)^d \bmod n = M^{de} \bmod n = M$$

- Bob can also encrypt M as $C = M^d \bmod n$ and decrypt with $C^e \bmod n$ because

  $$C^e \bmod n = (M^d)^e \bmod n = M^{de} \bmod n = M!$$

Security of RSA

- Security dependent on difficulty of finding d given e and n.
- Security closely tied to factoring n. So far integer factorization is considered very hard to do.
- A mathematical proof of security of RSA is a very important open problem.

Cryptographic Hash Functions

- A cryptographic hash function compresses a message M into fixed-length sequence H(M). Mapping is one-way and collision-resistant.
  - A function is one-way if it is computationally difficult to find M given H(M).
  - It is weakly collision-resistant if it is difficult to find a message M’ with H(M’) = H(M) given just H(M).
  - It is strongly collision-resistant if is difficult to find both M and M’ with H(M’) = H(M).

Digital Signatures

- A digital signature of a message is a way for an entity to prove that the sender sent message M.
- Alice computes H(M), hash of M, and forms S_{Alice}(M) by encrypting H(M) with her private key.
- She sends Bob (M, S_{Alice}(M)).
- Bob confirms that M has not changed in transit and that Alice sent it but computing H(M) and comparing it to the decryption of S_{Alice}(M)) with her public key.

Diffie-Helman Key Exchange

- Symmetric encryption is much faster than public-key encryption.
- Diffie and Helman invented a technique that two parties can use to agree on a secret key
- Both parties can use this key for symmetric encryption.

Diffie-Helman Key Exchange

- B & A choose prime p & primitive root g mod p.
  - g is primitive if for each r integer in {0,1,2,…, p-1}, r satisfies r = g^k mod p for some integer k.
- Alice's secret is a and Bob's secret is b.
  - A sends $r = g^a$ mod p to B.
  - B sends $s = g^b$ mod p.
  - A computes s^a mod p.
  - B computes r^b mod p.
- Let Q = s^a mod p = $(g^b \bmod p)^a = g^{ba}$ mod p = g^{ab} mod p = r^b mod p. The common secret is Q!

Security of Diffie-Hellman

- The values of a and b are secret.
  - Alice sends $r = g^a \bmod p$ to B in the clear.
  - Bob sends $s = g^b \bmod p$ to Alice in the clear.
- These transmissions reveal a and b IF it is possible to deduce a from $r = g^a \bmod p$ or b from $s = g^b \bmod p$.
- This is the *discrete logarithm* problem.
- No polynomial time algorithm is known for it.

Review

- Symmetric Cryptography
- Public-Key Cryptography
- Cryptographic Hash Functions
- Digital Signatures
- Diffie-Hellman Key Exchange