Cryptography – Day 4

One Time Pad and Optimality

Review

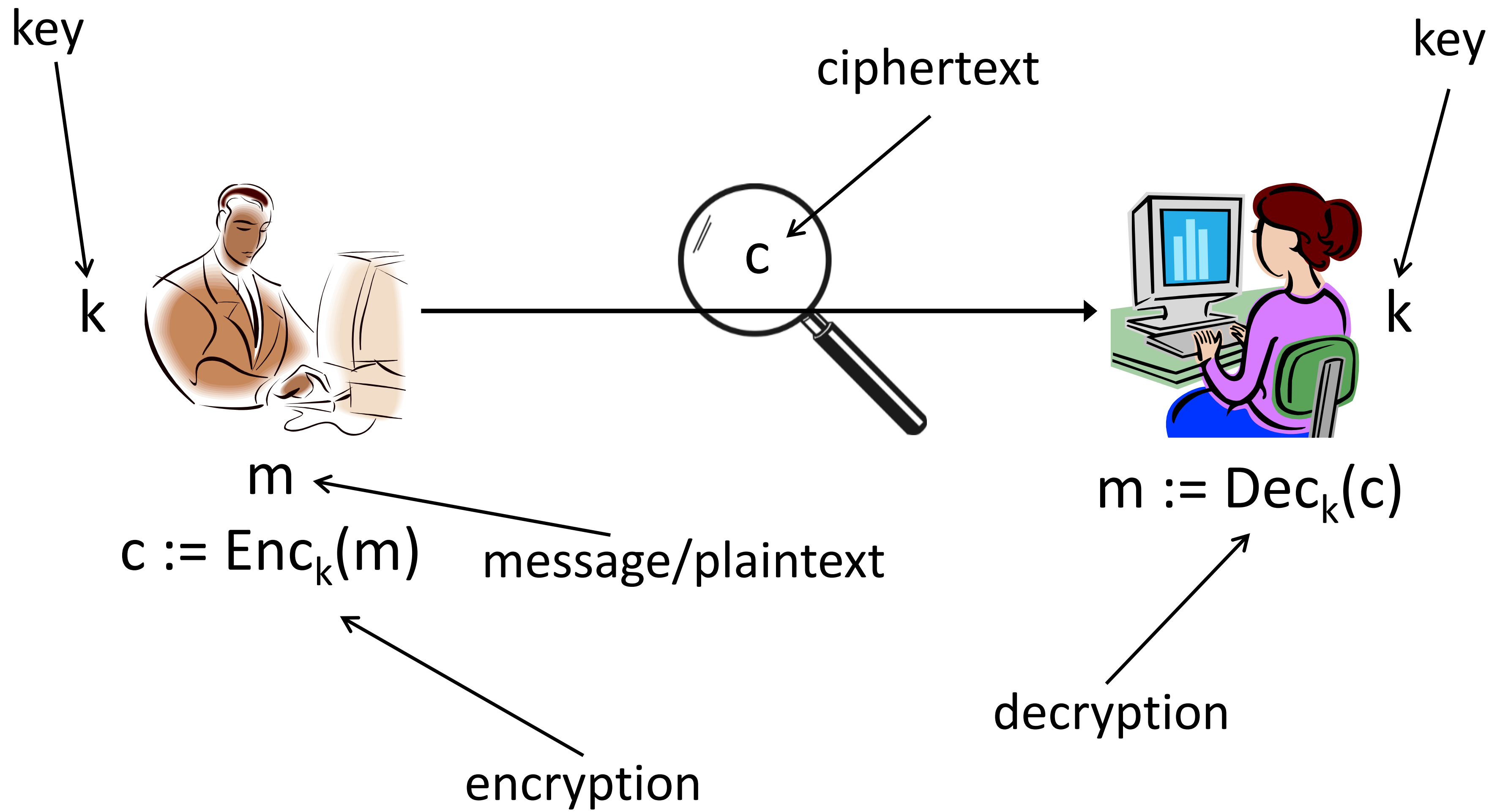
Private-key encryption

A *private-key encryption scheme* is defined by a message space $\mathcal{M}$ and algorithms (Gen, Enc, Dec)

- Gen determines a probability distribution over Key Space.
- Message space has some fixed probability distribution.
- Key Space and Message Space are independent.

Private-key encryption

A *private-key encryption scheme* is defined by a message space $\mathcal{M}$ and algorithms (Gen, Enc, Dec)



Crypto definitions (generally)

- Security guarantee/goal
  - What we want to achieve
  - Regardless of any *prior* information the attacker has about the plaintext, the ciphertext should leak no *additional* information about the plaintext
- Threat model
  - What (real-world) capabilities the attacker is assumed to have
  - Attacker Observes only one Ciphertext.

Perfect secrecy (formal)

- Encryption scheme (Gen, Enc, Dec) with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ is *perfectly secret* if for every distribution over $\mathcal{M}$, every $m \in \mathcal{M}$, and every $c \in \mathcal{C}$ with $\Pr[C=c] > 0$, it holds that

$$\Pr[M = m \mid C = c] = \Pr[M = m].$$

Concept Check

- Consider the shift cipher, and the distribution;
  Pr[M='hi'] = 0.3,
  Pr[M='no'] = 0.2,
  Pr[M='in']= 0.5

- What is the Pr[M = 'hi' | C = 'xy']?
  - = Pr[C = 'xy' | M = 'hi'] · Pr[M = 'hi']/Pr[C = 'xy']

Perfectly Secret Encryption

- The shift cipher is not perfectly secret!
  - At least not for 2-character messages

- How to construct a perfectly secret scheme?

One-time pad

- Patented in 1917 by Vernam
  - Recent historical research indicates it was invented (at least) 35 years earlier

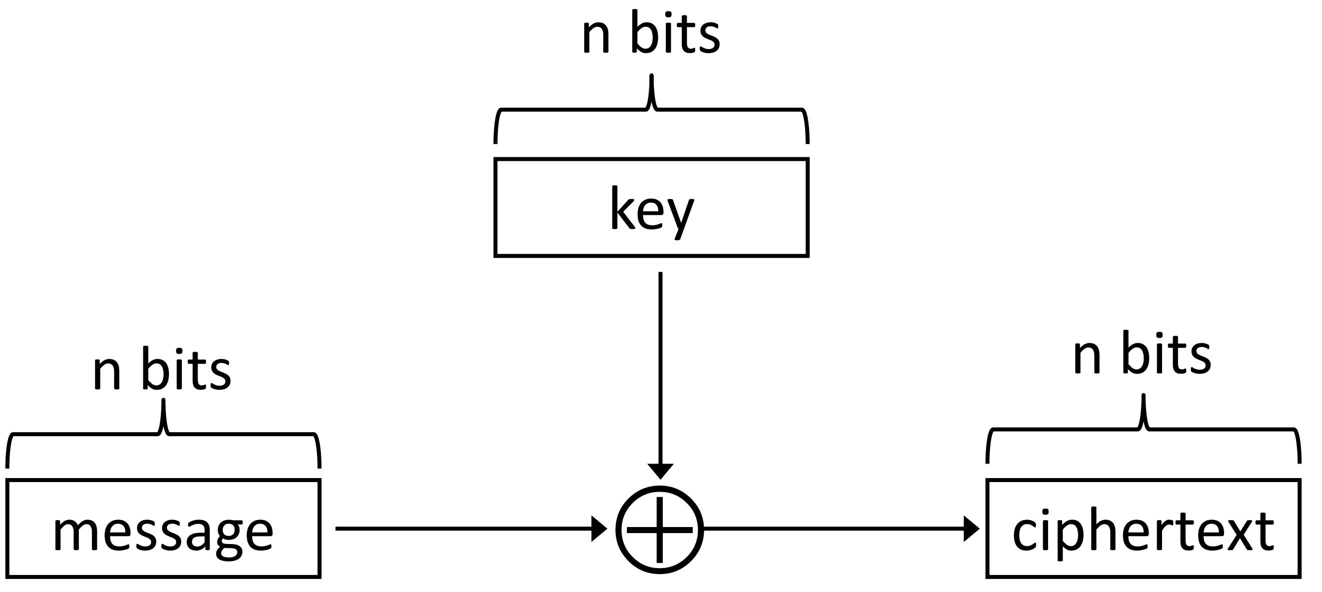
- Proven perfectly secret by Shannon (1949)

One-time pad

- Let $\mathcal{M} = \{0,1\}^n$
- Gen: choose a uniform key $k \in \{0,1\}^n$
- $\mathrm{Enc}_k(m) = k \oplus m$
- $\mathrm{Dec}_k(c) = k \oplus c$

- Correctness:
  $\mathrm{Dec}_k(\ \mathrm{Enc}_k(m)\ ) = k \oplus (k \oplus m)$
  $= (k \oplus k) \oplus m = m$

One-time pad



Perfect secrecy of one-time pad

- Note that *any* observed ciphertext can correspond to *any* message (why?)
  - (This is necessary, but not sufficient, for perfect secrecy)
- So, having observed a ciphertext, the attacker cannot conclude for certain which message was sent

Implementing the one-time pad

Key generation

- Read desired number of bytes from /dev/urandom
- Output the result to a file

Encryption

- Plaintext = sequence of ASCII characters
- Key = sequence of hex digits

- Read them; XOR them to get the ciphertext

Decryption

- Reverse encryption
- Read ciphertext and key; XOR them to recover the message

Limitations and *Optimality*

One-time pad

- The one-time pad achieves perfect secrecy!
- One-time pad has historically been used in the real world
  - E.g., “red phone” between DC and Moscow
- Not currently used!
  - Why not?

One-time pad

- Several limitations
  - The key is as long as the message
  - Only secure if each key is used to encrypt a *single* message
    - (Trivially broken by a known-plaintext attack)

$\Rightarrow$ Parties must share keys of (total) length equal to the (total) length of all the messages they might ever send

Using the same key twice?

- Say $c_1 = k \oplus m_1$
  $c_2 = k \oplus m_2$

- Attacker can compute
  $c_1 \oplus c_2 = (k \oplus m_1) \oplus (k \oplus m_2) = m_1 \oplus m_2$

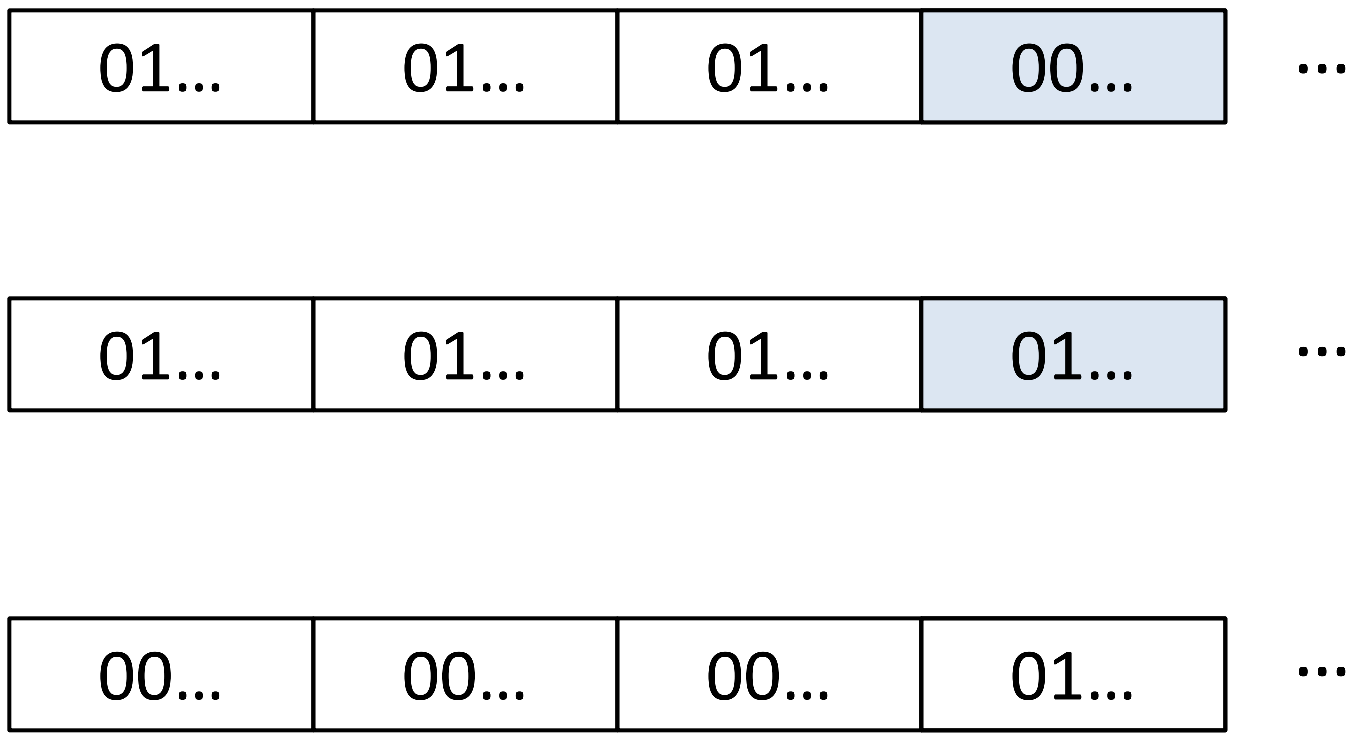
- This leaks information about m_1, m_2!

Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x20	32	Space	0x40	64	@	0x60	96	`
0x21	33	!	0x41	65	A	0x61	97	a
0x22	34	"	0x42	66	B	0x62	98	b
0x23	35	#	0x43	67	C	0x63	99	c
0x24	36	$	0x44	68	D	0x64	100	d
0x25	37	%	0x45	69	E	0x65	101	e
0x26	38	&	0x46	70	F	0x66	102	f
0x27	39	'	0x47	71	G	0x67	103	g
0x28	40	(	0x48	72	H	0x68	104	h
0x29	41	)	0x49	73	I	0x69	105	i
0x2A	42	*	0x4A	74	J	0x6A	106	j
0x2B	43	+	0x4B	75	K	0x6B	107	k
0x2C	44	,	0x4C	76	L	0x6C	108	l
0x2D	45	-	0x4D	77	M	0x6D	109	m
0x2E	46	.	0x4E	78	N	0x6E	110	n
0x2F	47	/	0x4F	79	O	0x6F	111	o
0x30	48	0	0x50	80	P	0x70	112	p
0x31	49	1	0x51	81	Q	0x71	113	q
0x32	50	2	0x52	82	R	0x72	114	r
0x33	51	3	0x53	83	S	0x73	115	s
0x34	52	4	0x54	84	T	0x74	116	t
0x35	53	5	0x55	85	U	0x75	117	u
0x36	54	6	0x56	86	V	0x76	118	v
0x37	55	7	0x57	87	W	0x77	119	w
0x38	56	8	0x58	88	X	0x78	120	x
0x39	57	9	0x59	89	Y	0x79	121	y
0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x3B	59	;	0x5B	91	[	0x7B	123	{
0x3C	60	<	0x5C	92	\	0x7C	124	|
0x3D	61	=	0x5D	93	]	0x7D	125	}
0x3E	62	>	0x5E	94	^	0x7E	126	~
0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Source: http://benborowiec.com/2011/07/23/better-ascii-table/

- Letters all begin with 01…
- The space character begins with 00…
- XOR of two letters gives 00…
- XOR of letter and space gives 01…
- Easy to identify XOR of letter and space!

In pictures



In pictures

01…	01…	01…	00…	…

01…	01…	01…	01…	…

00…	00…	00…	01010000	…

$01010000 = 00100000 \oplus ??$

One-time pad

- Drawbacks
  - Key as long the message
  - Only secure if each key is used to encrypt *once*
  - Trivially broken by a known-plaintext attack

- These limitations are *inherent* for schemes achieving perfect secrecy

Optimality of the one-time pad

- Theorem: if (Gen, Enc, Dec) with message space $\mathcal{M}$ is perfectly secret, then $|\mathcal{K}| \geq |\mathcal{M}|$.
- Intuition:
  - Given any ciphertext, try decrypting under every possible key in $\mathcal{K}$
  - This gives a list of up to $|\mathcal{K}|$ possible messages
  - If $|\mathcal{K}| < |\mathcal{M}|$, some message is not on the list