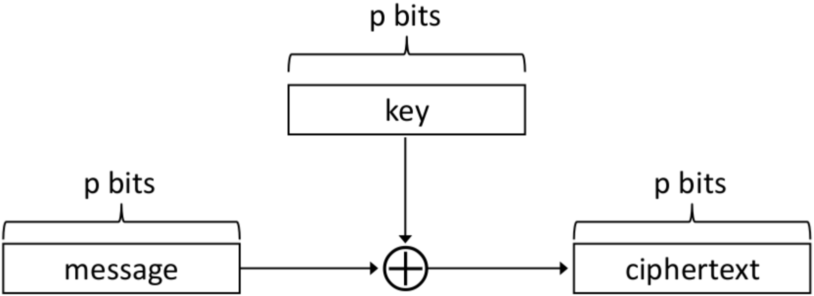
One-time Pad



Private-key encryption

A private-key encryption scheme is defined by a message space $\mathbb{M}$, (key space $\mathbb{K}$) and algorithms $(\mathsf{KeyGen}, \mathsf{e}, \mathsf{d})$:

- KeyGen (key-generation algorithm): outputs $\boldsymbol{k} \in \mathbb{K}$. Usually: $\boldsymbol{k} \in \mathbb{K}$ uniformly random. (This algorithm is sometimes left implicit in the book)
- e (encryption algorithm): takes as input key $\boldsymbol{k}$ and message $\boldsymbol{m} \in \mathbb{M}$; outputs ciphertext $\boldsymbol{c} \leftarrow \mathsf{e}_{\boldsymbol{k}}(\boldsymbol{m})$
- d (decryption algorithm): takes as input key $\boldsymbol{k}$ and ciphertext $\boldsymbol{c}$; outputs $\boldsymbol{m}$ or "error": $\boldsymbol{m} = \mathsf{d}_{\boldsymbol{k}}(\boldsymbol{c})$

One-time Pad

- Let $\mathbb{M} = \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n}}$
- KeyGen: choose a uniform key $\boldsymbol{k} \in \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n}}$
- $\mathsf{e}_{\boldsymbol{k}}(\boldsymbol{m}) = \boldsymbol{k} \oplus \boldsymbol{m}$
- $\mathsf{d}_{\boldsymbol{k}}(\boldsymbol{c}) = \boldsymbol{k} \oplus \boldsymbol{c}$
- $\mathsf{d}_{\boldsymbol{k}}(\mathsf{e}_{\boldsymbol{k}}(\boldsymbol{m})) = \boldsymbol{k} \oplus (\boldsymbol{k} \oplus \boldsymbol{m}) = (\boldsymbol{k} \oplus \boldsymbol{k}) \oplus \boldsymbol{m} = \boldsymbol{m}$

Perfect Secrecy of One-time Pad

Theorem

The One-time Pad satisfies perfect secrecy.

Intuition

- Having observed a ciphertext, the attacker cannot conclude for certain which message was sent

One-time Pad and Brute-force Attacks

The same ciphertext	Decrypted with this key...	...gives this plaintext
SMAIJIZJSIFPSTWFI	STHIHYZQRRBPIOWNP	ATTACKATBREAKFAST
	BIHRFIGIODRYOGIRV	RETREATBEFORENOON
	MYARVOMGKVDHBRLBQ	GOAROUNDINCIRCLES
	ATAVGOGQORURAAOUX	STANDUTTERLYSTILL
	AENCQMLCSTQRAFJZQ	SINGTWOHAPPYSONGS
	AFMOQIHYEOCPAEINQ	SHOUTASLOUDASPOSS
	IIWTQUGJHXHXQMDLW	KEEPTOTALLYSCHTUM
	SBPUPPKPZTRXALVUE	ALLOUTPUTPOSSIBLE

- OTP resists even a brute-force attack
- Decrypt a ciphertext with every key returns every possible plaintext (incl. every ASCII/English string)
- No way of telling the correct plaintext

Image credit: https://nakedsecurity.sophos.com

Perfect Secrecy of One-time Pad

Proof.

- Fix arbitrary distribution over $\mathbb{M} = \{0,1\}^n$, and choose arbitrary $m, c \in \{0,1\}^n$
- Check if

$$\mathbf{Pr}[M = m | C = c] = \mathbf{Pr}[M = m]$$

Perfect Secrecy of One-time Pad

Proof.

- Recall (Bayes' theorem)

$$\mathbf{Pr}[M = m | C = c] = \frac{\mathbf{Pr}[C = c | M = m]\ \mathbf{Pr}[M = m]}{\mathbf{Pr}[C = c]}$$

- We can see that $\forall c, m$

$$\mathbf{Pr}[C = c | M = m] = \mathbf{Pr}[M \oplus K = c | M = m] =$$
$$= \mathbf{Pr}[m \oplus K = c] = \mathbf{Pr}[K = c \oplus m] = 2^{-n}$$

Perfect Secrecy of One-time Pad

Proof.

By law of total probability:

$$\begin{aligned}
&\mathbf{Pr}[\boldsymbol{C} = \boldsymbol{c}] = \\
&= \sum_{m'} \mathbf{Pr}[\boldsymbol{C} = \boldsymbol{c} | \boldsymbol{M} = \boldsymbol{m'}] \, \mathbf{Pr}[\boldsymbol{M} = \boldsymbol{m'}] \\
&= \sum_{m'} \mathbf{Pr}[\boldsymbol{K} = \boldsymbol{m'} \oplus \boldsymbol{c} | \boldsymbol{M} = \boldsymbol{m'}] \, \mathbf{Pr}[\boldsymbol{M} = \boldsymbol{m'}] \\
&= \sum_{m'} \mathbf{2}^{-n} \, \mathbf{Pr}[\boldsymbol{M} = \boldsymbol{m'}] \\
&= \mathbf{2}^{-n} \sum_{m'} \mathbf{Pr}[\boldsymbol{M} = \boldsymbol{m'}] = \mathbf{2}^{-n}
\end{aligned}$$

Perfect Secrecy of One-time Pad

Proof.

$$\begin{aligned}
&\mathbf{Pr}[M = m | C = c] = \\
&= \frac{\mathbf{Pr}[C = c | M = m]\,\mathbf{Pr}[M = m]}{\mathbf{Pr}[C = c]} \\
&= \frac{\mathbf{Pr}[K = m \oplus c | M = m]\,\mathbf{Pr}[M = m]}{2^{-n}} \\
&= \frac{2^{-n}\,\mathbf{Pr}[M = m]}{2^{-n}} \\
&= \mathbf{Pr}[M = m]
\end{aligned}$$

□

One-time Pad

- The One-time Pad achieves perfect secrecy!
- Resists even a brute-force attack
- Not currently used! Why?

One-time Pad

Limitations of OTP

1. The key is as long as the message
2. A key must be used only once
   - Only secure if each key is used to encrypt a single message
   - (Trivially broken by a known-plaintext attack)

$\implies$ 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

$$\boldsymbol{c_1 = k \oplus m_1}$$
$$\boldsymbol{c_2 = k \oplus m_2}$$

- Attacker can compute

$$\boldsymbol{c_1 \oplus c_2 = (k \oplus m_1) \oplus (k \oplus m_2) = m_1 \oplus m_2}$$

- This leaks information about $\boldsymbol{m_1, m_2}$

Using the Same Key Twice?

$\boldsymbol{m_1 \oplus m_2}$ leaks information about $\boldsymbol{m_1, m_2}$

Is this significant?

- $\boldsymbol{m_1 \oplus m_2}$ reveals where $\boldsymbol{m_1, m_2}$ differ
- No longer perfectly secret!
- Exploiting characteristics of ASCII...

ASCII table (recall)

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

https://hubpages.com/technology/What-Are-ASCII-Codes

Using the Same Key Twice: recall ASCII

Observatoins

- Letters begin with `0x4`, `0x5`, `0x6` or `0x7`
  - $\implies$ **letters all begin with 01...**
- ASCII code for the space character `0x20` = **00100000**
  - $\implies$ **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!**

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