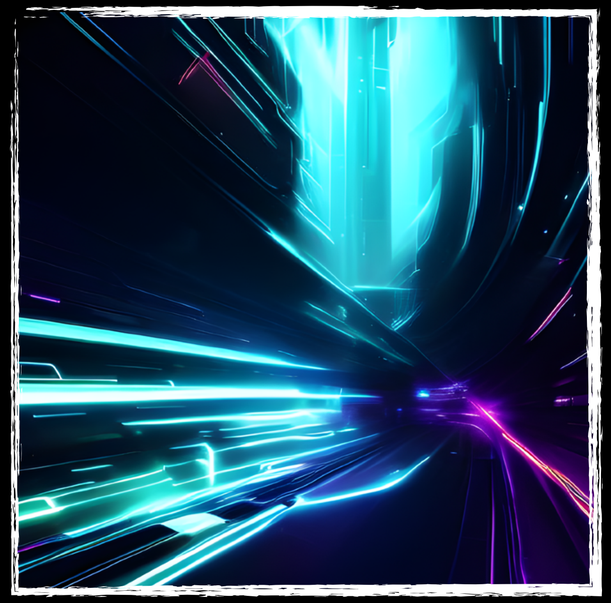
Crypto

Curious @ NCYU Hackers

2023.10.03

>_ 目錄

>_ WhoAmI



- ID : Curious
- DC : curious_lucifer
- SCIST 4rd 總召 / LoTuX CTF 創辦人
- Github : Link
- Blog : Link

>_ Lab

Lab : http://172.104.90.38

Flag Format : NCYU_HACKERS{.*}

編碼

>_ ASCII

- 一張把字元和數字互相轉換的表
- 數字只包含 0 ~ 127
- 用 chr(num)　把數字轉成字元
- 用 ord(char) 把字元轉成數字

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

>_ Base64

1. 把字串分成 3 個一組
2. 把各個字元轉成相對應的 ASCII 數字
3. 把各個數字轉為二進位（一個數字 8 bits）
4. 把 24 bits 分成 6 bits 一組
5. 把 6 bits 的二進位數字轉成對應的字元

文字	M								a								n							
ASCII編碼	77								97								110							
位元	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
索引	19								22				5				46							
Base64編碼	T								W				F				u							

十進位	二進位	字元	十進位	二進位	字元	十進位	二進位	字元	十進位	二進位	字元
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/
填充		=									

>_ Base64

1. 把字串分成 3 個一組，沒有滿的補 \0
2. 把各個字元轉成相對應的 ASCII 數字
3. 把各個數字轉為二進位（一個數字 8 bits）
4. 把 24 bits 分成 6 bits 一組
5. 把 6 bits 的二進位數字轉成對應的字元
6. 如果那 6 bits 都是補的 0，那就轉成 =

文字（1 Byte）	A																							
位元	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位元（補0）	0	1	0	0	0	0	0	1	**0**	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
Base64編碼	Q				Q				=				=											

十進位	二進位	字元	十進位	二進位	字元	十進位	二進位	字元	十進位	二進位	字元
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/
填充		=									

>_ Base64

Crypto Lab : Base64

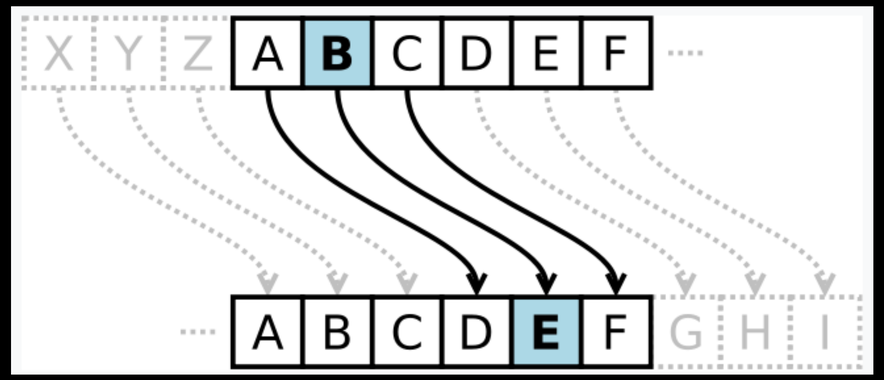
Crypto Lab : Identify

古典密碼學

>_ Caesar Cipher

加密

1. 把明文中所有英文字母轉成大寫（或小寫）
2. 按照設定的偏移量更改字母



（偏移量：3）

解密

1. 按照設定的偏移量把字母移回去

破解方法

反正偏移量只有可能是 `0 ~ 25`（英文大寫或小寫只有 26 個），那就暴力嘗試每一個偏移量

>_ Caesar Cipher

Crypto Lab : Caesar Cipher

>_ Affine Cipher

加密

1. 選取兩個數字 a 和 b，且 a 要和 m 互質
2. 按照規定把明文中各個字元轉成對應的數字
3. 將每一個數字帶入 $y \equiv a \cdot x + b \pmod{m}$ 中的 x
4. 將各個數字計算出的 y 按照規定轉成對應的字元

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

(一種簡單的字元和數字轉換方式，對於這個轉換表 $m = 26$)

解密

1. 按照規定把明文中各個字元轉成對應的數字
2. 將每一個數字帶入 $y \equiv a^{-1} \cdot (x - b) \pmod{m}$ 中的 x
3. 將各個數字計算出的 y 按照規定轉成對應的字元

破解方法

基本上也是可以直接暴力嘗試 a 和 b，嘗試 $(m-1) \cdot m$ 次後就可以找到所有 a 和 b（如果 m 不是質數會更少）

>_ Affine Cipher

Crypto Lab : Affine Cipher

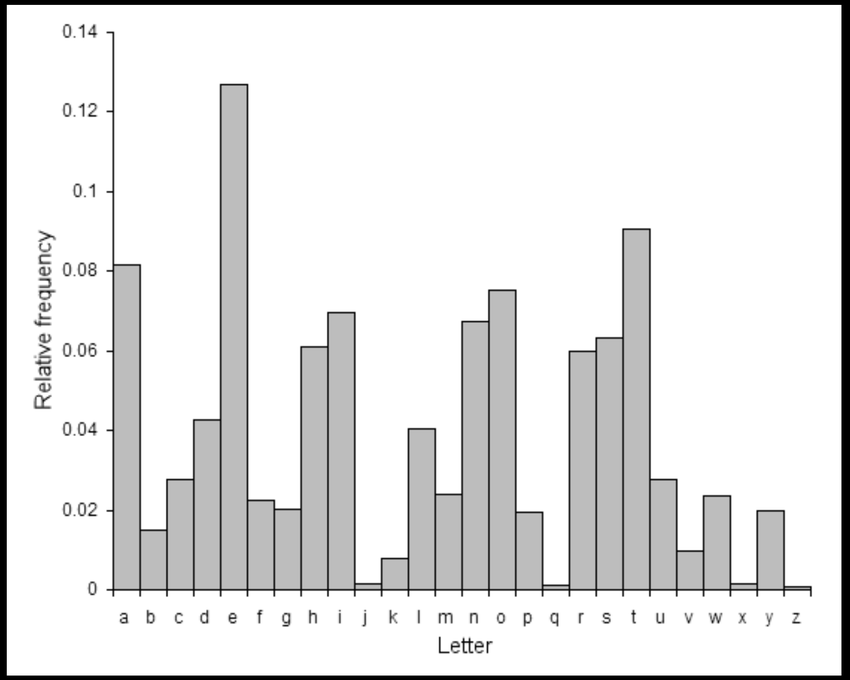
>_ Affine Cipher

簡單的頻率分析

如果想要破解 Affine Cipher 的話，需要暴力計算出 $(m-1)\cdot m$ 個不同 key 所解密出來的明文，如果 m 很大的話，基本上用眼睛去辨別哪一個是對的明文就不太現實。

如果我們知道用正確的 key decrypt 出來的明文會是英文，我們就可以利用英文各個字母的頻率分佈去分析哪一個 key decrypt 出來的明文最像英文

公式：$r = \sum_{i=1}^{26} \frac{\text{字母頻率偏差}^2}{\text{英文字母頻率}^2}$



>_ Substitution Cipher

加密

1. 把兩組相同的字元組分別打亂，然後讓他們一一對應
2. 按照剛建立一一對應的規則加密明文

明文為 ABCDEFGHIJKLMNOPQRSTUVWXYZ

密文為 ZEBRASCDFGHIJKLMNOPQTUVWXY

（一一對應的規則大致上像這樣，但這打的不夠亂）

解密

1. 按照設定的規則解密明文

破解方法

基本上就只有頻率分析這條路可以走，而且要分析的包括單個字元出現的頻率、連續字元出現的頻率等等，這實現起來很麻煩，直接用線上工具

>_ Substitution Cipher

Crypto Lab : Substitution Cipher

Crypto Lab : Straddle Checkerboard

>_ 古典密碼總結

- 大部分的古典密碼都可以直接用頻率分析解開
- 其它的部分就要根據加密/解密的方式去特地構造破解方法，不過也不會離頻率分析太遠

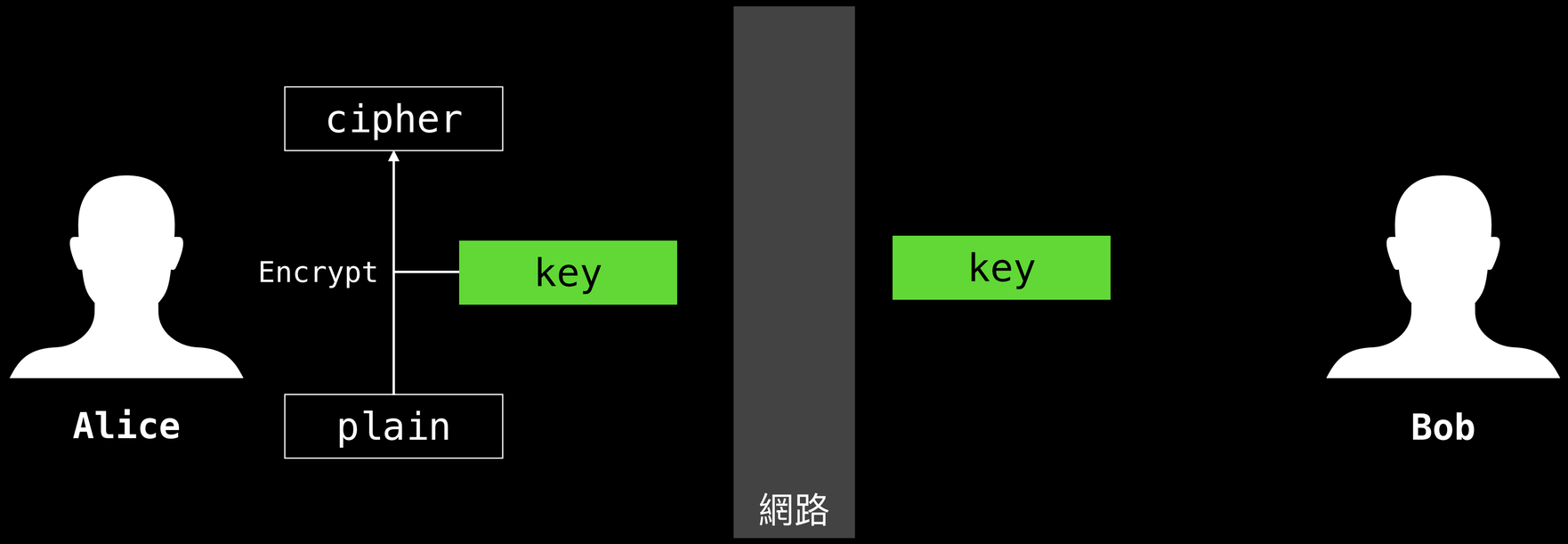
>_ 古典密碼總結

Crypto Lab : Affine Cipher Double Strength

現代密碼學

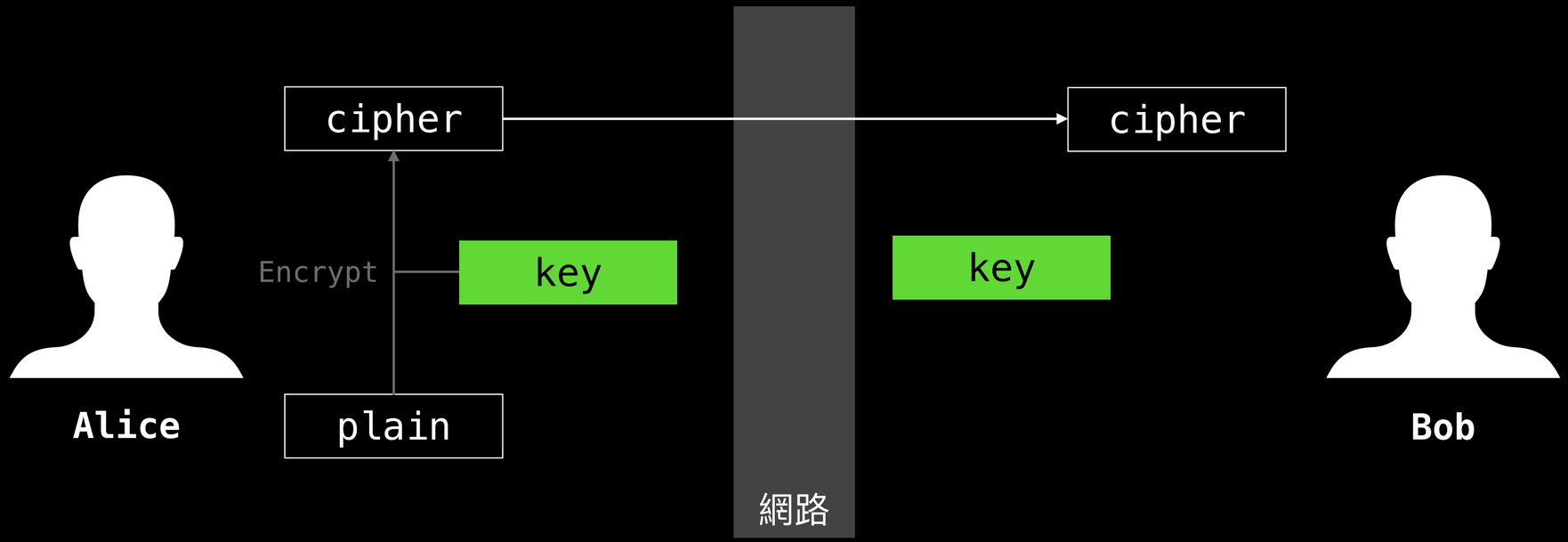
>_ 現代密碼學

- **對稱式加密**
- 非對稱式加密



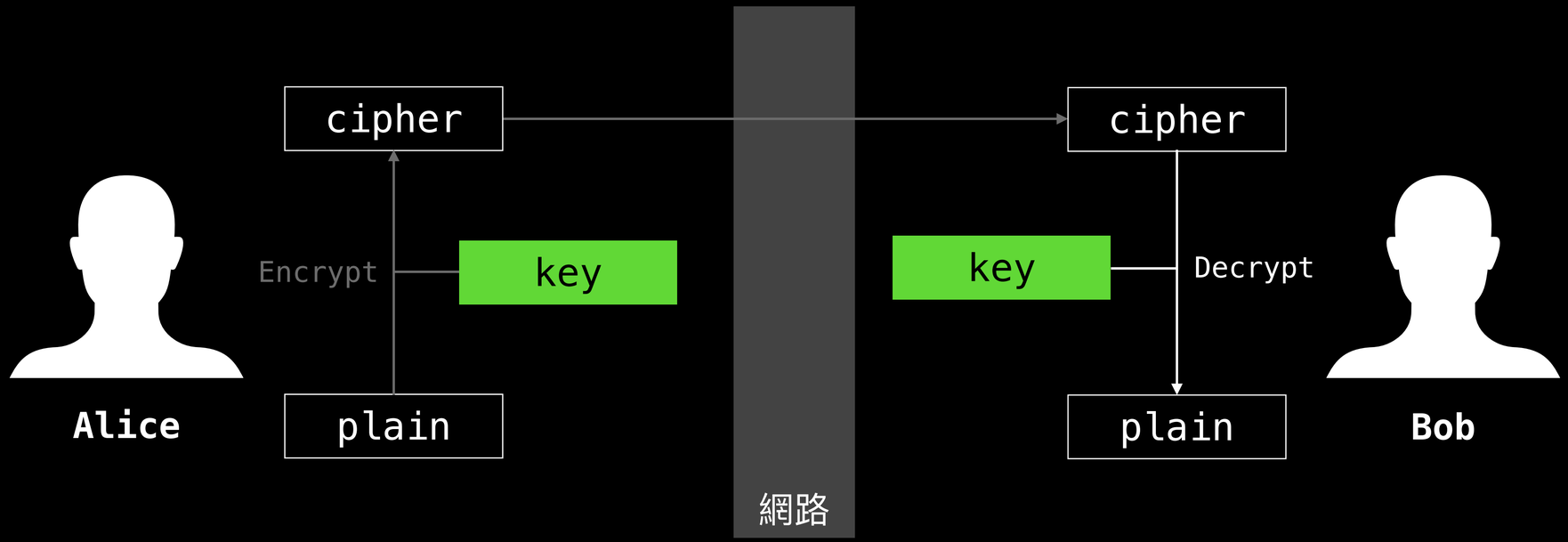
>_ 現代密碼學

- **對稱式加密**
- 非對稱式加密



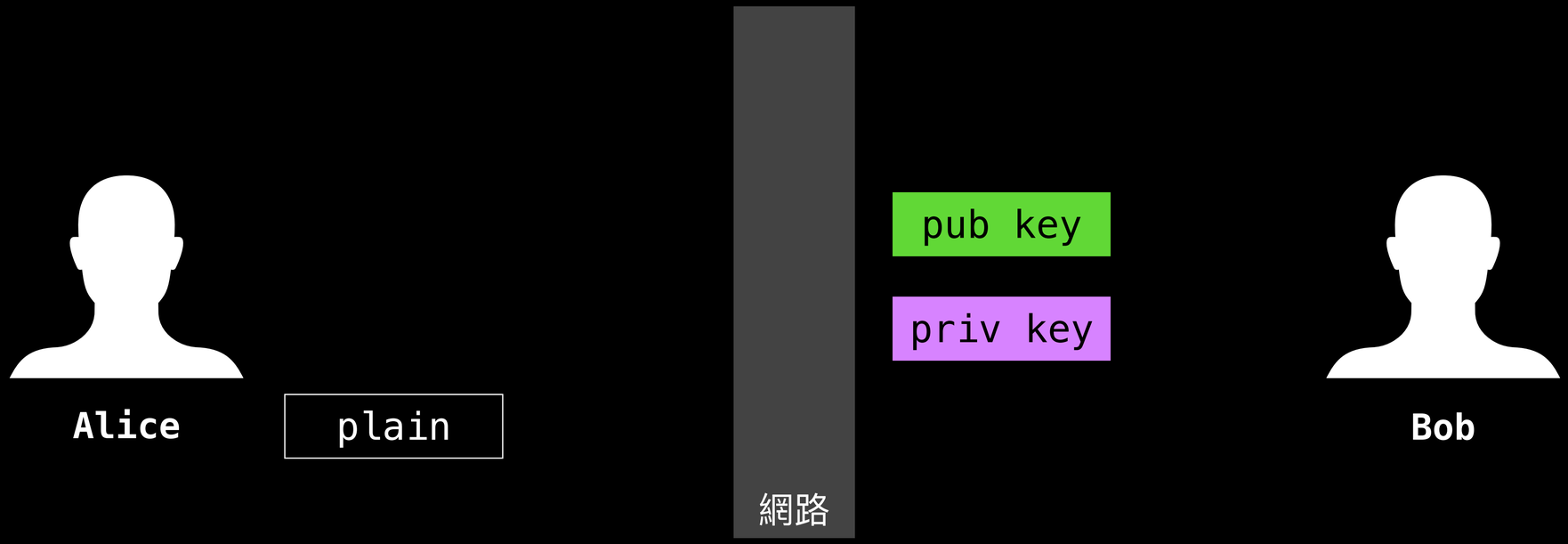
>_ 現代密碼學

- **對稱式加密**
- 非對稱式加密



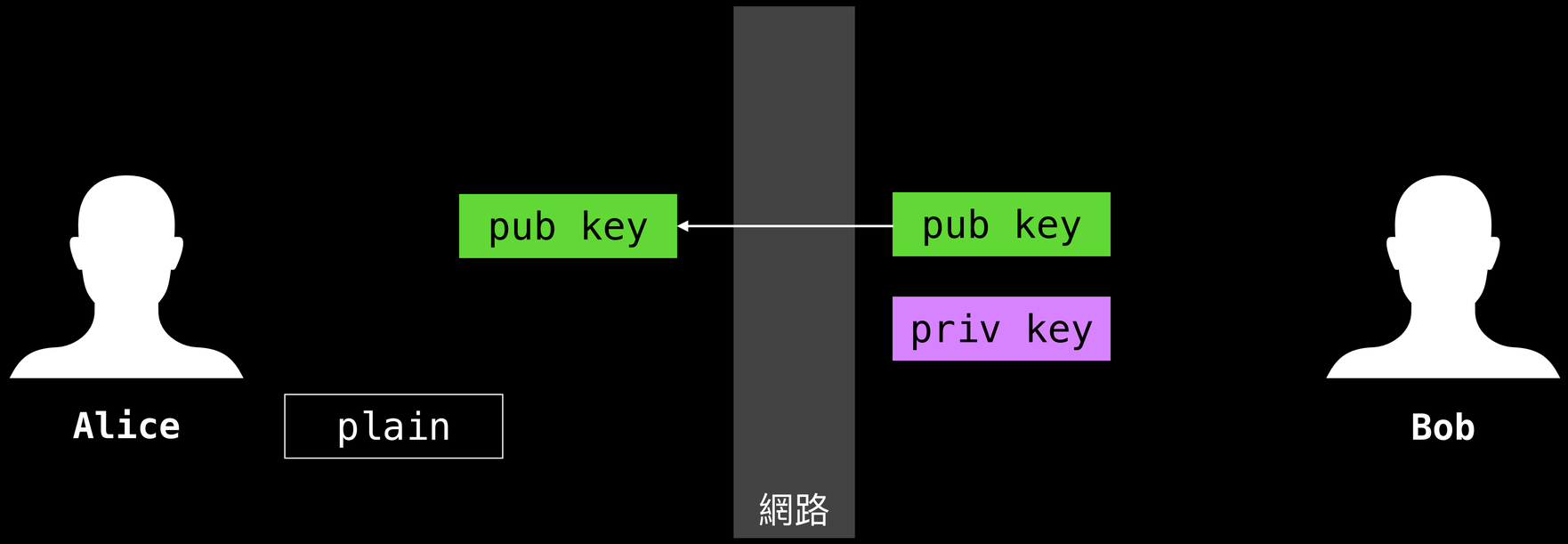
>_ 現代密碼學

- 對稱式加密
- **非對稱式加密**



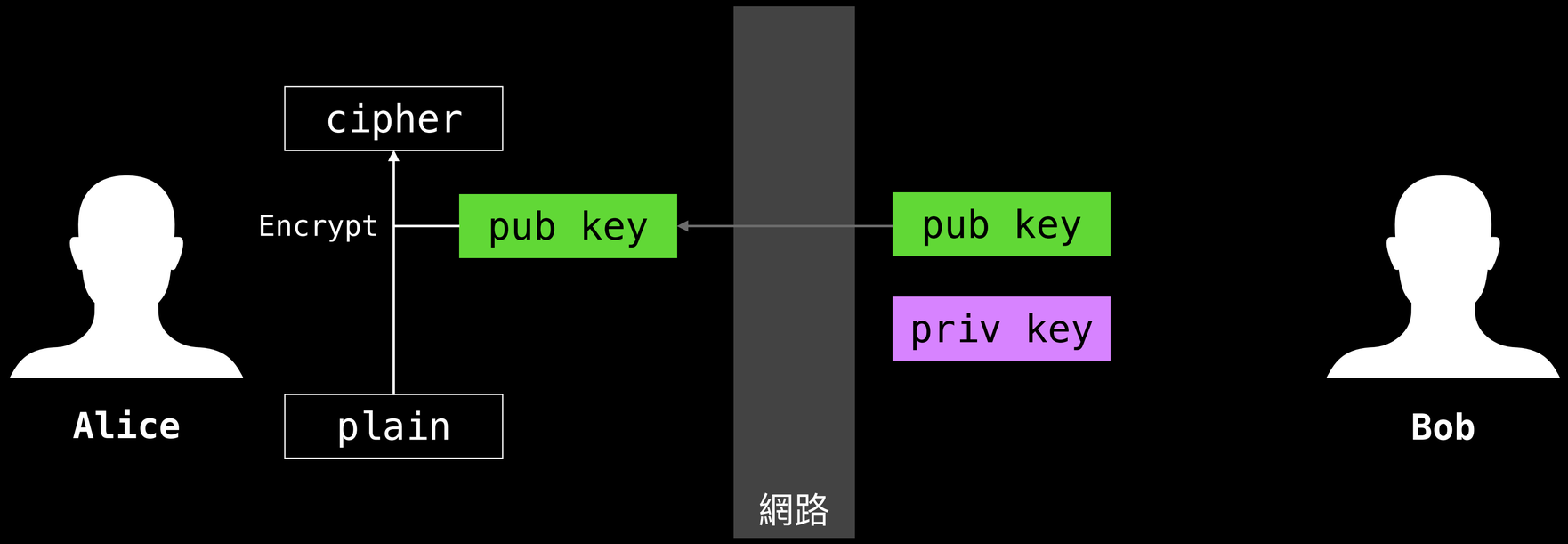
>_ 現代密碼學

- 對稱式加密
- **非對稱式加密**



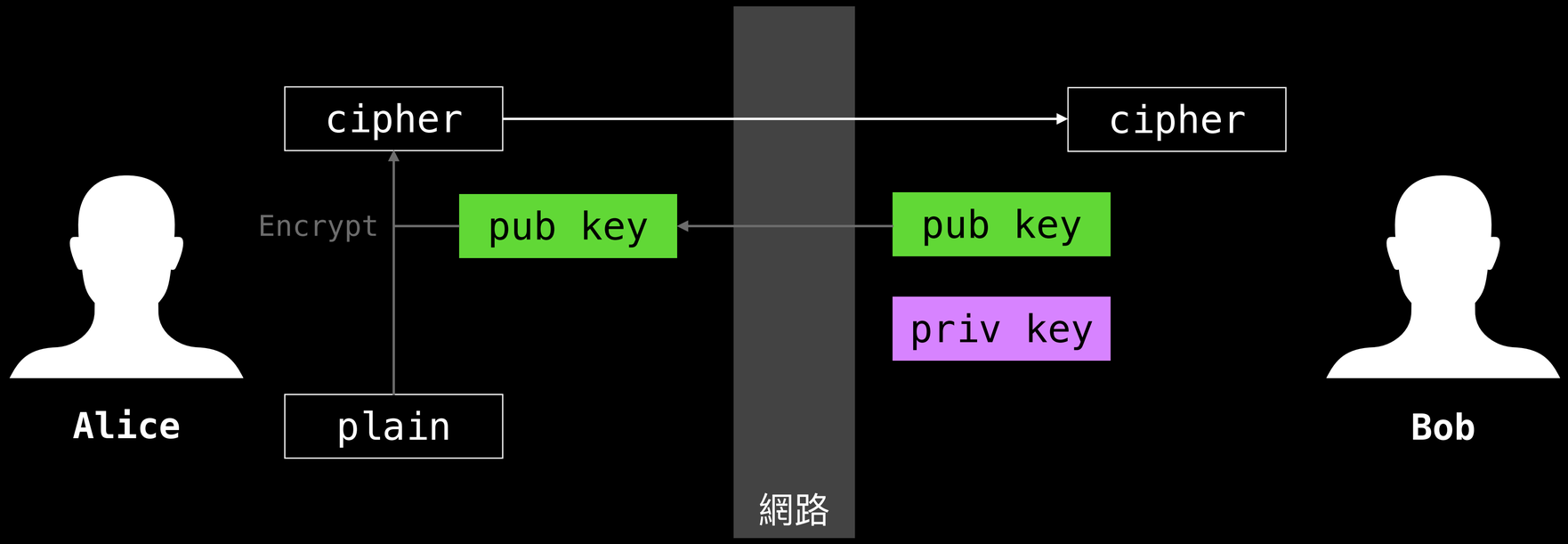
>_ 現代密碼學

- 對稱式加密
- **非對稱式加密**



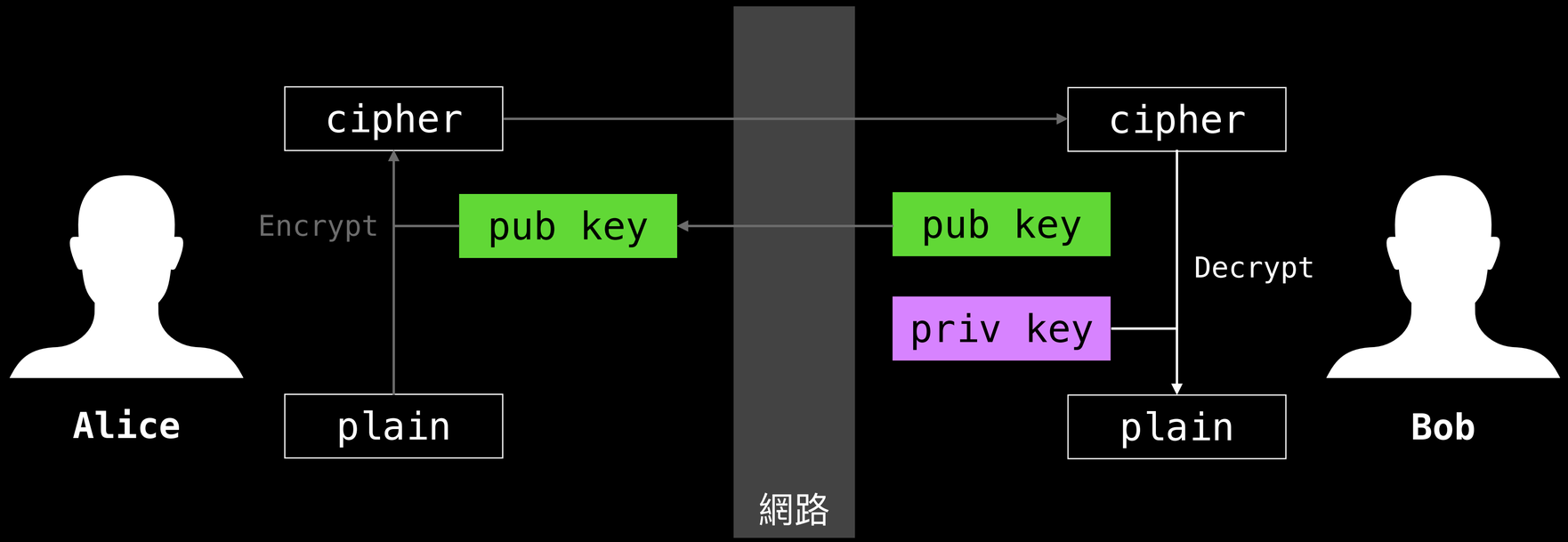
>_ 現代密碼學

- 對稱式加密
- **非對稱式加密**



>_ 現代密碼學

- 對稱式加密
- **非對稱式加密**



資源整理

>_ 資源整理

CryptoHack

Blog

SCIST Crypto CTF