CRYPTOGRAPHY

Alessandro Tomasi
altomasi@fbk.eu

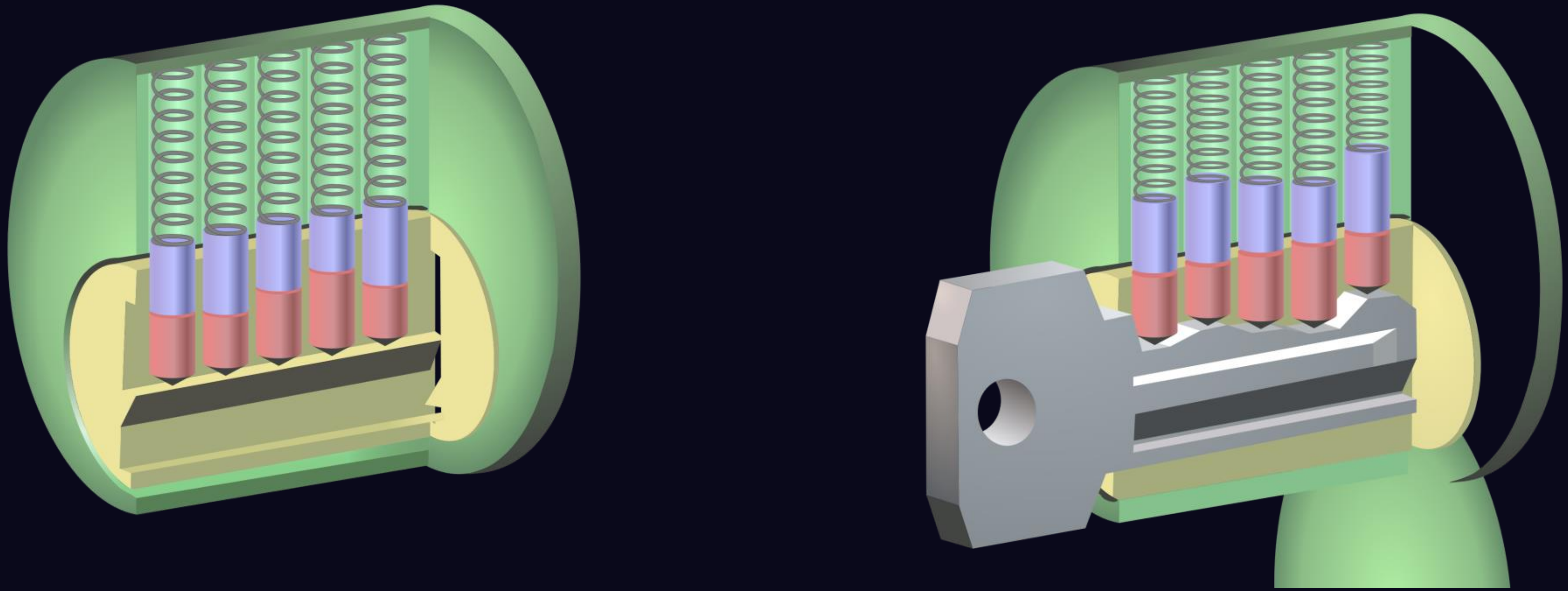
FONDAZIONE BRUNO KESSLER
CENTER FOR CYBERSECURITY
ALEPH APPLIED CRYPTOGRAPHY

Classical ciphers and how to break them

Encoding vs encryption
Transposition and substitution ciphers
Statistical analysis

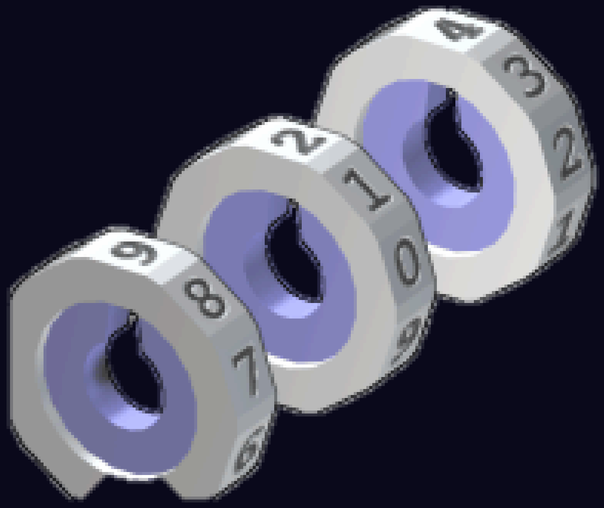
FONDAZIONE BRUNO KESSLER
CENTER FOR CYBERSECURITY
ALEPH APPLIED CRYPTOGRAPHY

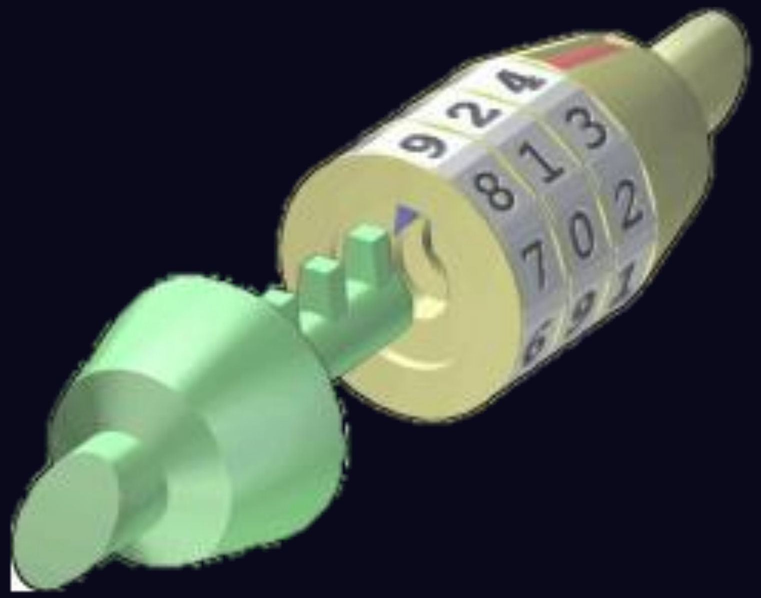
A key is information – tumbler lock



- The physical key is the support on which information is encoded
- To ensure confidentiality, keys must be unpredictable (chosen uniformly at random) and kept secret

A key is information – combination lock





- Brute force: 10^3 possibilities, < 17 minutes

https://en.wikipedia.org/wiki/File:Combination_discs.png
Original: Oldie~commonswiki

Encoding vs encrypting

- Code: **single** deterministic map between Message and Codeword sets, depending on the channel.
  - *Encode* $E: M \to C$
  - *Decode* $D: C \to M$
  - *Error correction:* $|C| > |M|$ *(redundancy)*
  - *Compression:* $|C| < |M|$
  - *Based on human-readability, double-clickability, audio channel, error types etc.*
- Cipher: **one of many** possible maps between sets – the Message space and the Ciphertext space – indexed by a secret key, drawn at random from the Key space.
  - *Encrypt* $E: K \times M \to C$
  - *Decrypt* $D: K \times C \to M$

Encoding examples

Base64: bin to char

int	bin	char
0	000000	A
26	011010	a
52	110100	0
61	111101	9
62	111110	+
63	111111	/
padding		=

hex

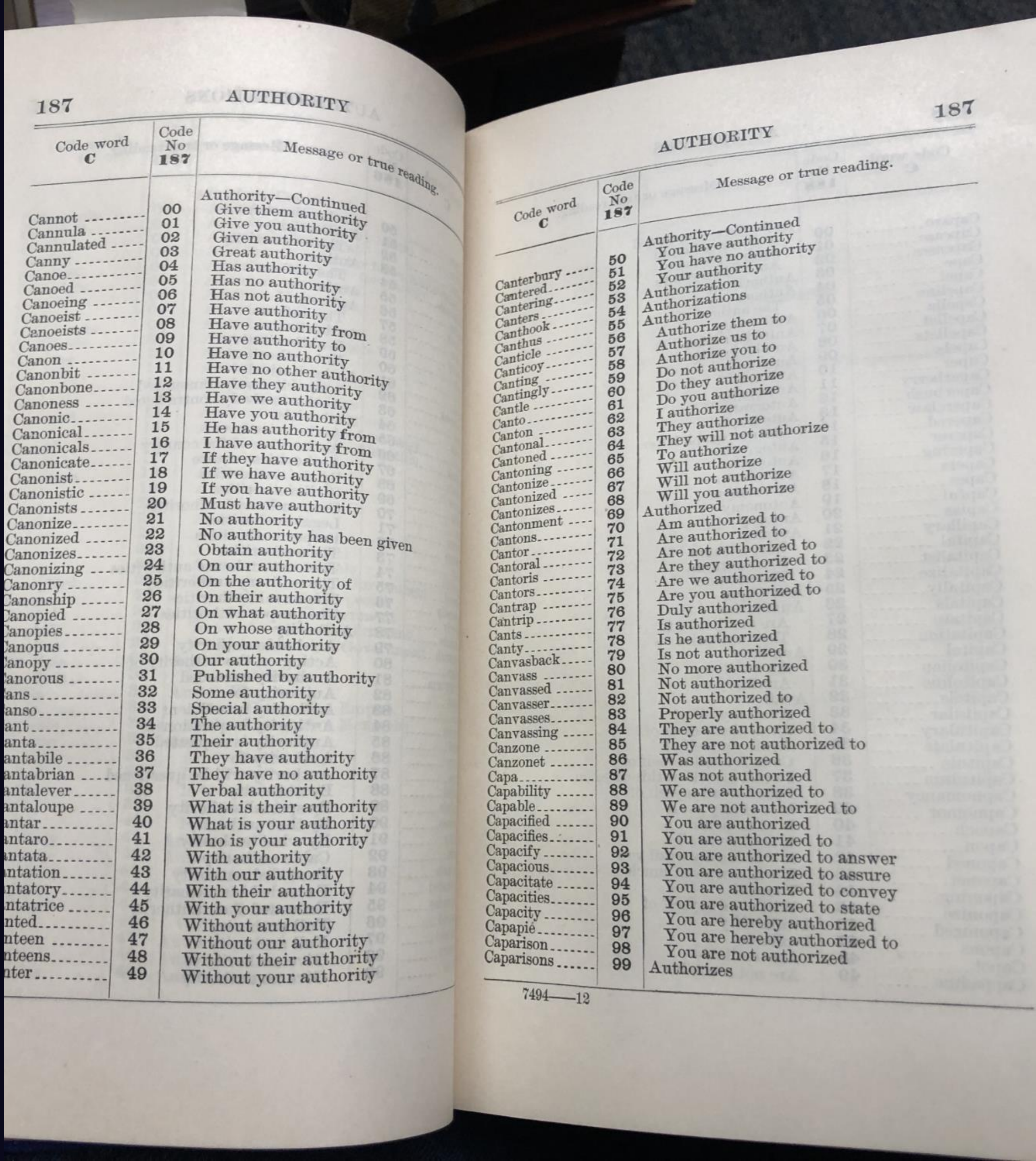
int	bin	hex
0	00000000	\x00
26	00011010	\x1a
63	00111111	\x3f
255	11111111	\xff

Morse

char	signal
A	● ▬
S	● ● ●
O	▬ ▬ ▬

padding forces the encoded output to a multiple of 4 char, when the unencoded bin is not a multiple of 3 bytes
base58 ~ base64 without { 0 O I l + \ }. Less ambiguous to read, double-click selects whole string.

Codebook



187 AUTHORITY

Code word C	Code No 187	Message or true reading.
		Authority—Continued
Cannot	00	Give them authority
Cannula	01	Give you authority
Cannulated	02	Given authority
Canny	03	Great authority
Canoe	04	Has authority
Canoed	05	Has no authority
Canoeing	06	Has not authority
Canoeist	07	Have authority
Canoeists	08	Have authority from
Canoes	09	Have authority to
Canon	10	Have no authority
Canonbit	11	Have no other authority
Canonbone	12	Have they authority
Canoness	13	Have we authority
Canonic	14	Have you authority
Canonical	15	He has authority from
Canonicals	16	I have authority from
Canonicate	17	If they have authority
Canonist	18	If we have authority
Canonistic	19	If you have authority
Canonists	20	Must have authority
Canonize	21	No authority
Canonized	22	No authority has been given
Canonizes	23	Obtain authority
Canonizing	24	On our authority
Canonry	25	On the authority of
Canonship	26	On their authority
Canopied	27	On what authority
Canopies	28	On whose authority
Canopus	29	On your authority
Canopy	30	Our authority
Canorous	31	Published by authority
ans	32	Some authority
anso	33	Special authority
ant	34	The authority
anta	35	Their authority
antabile	36	They have authority
antabrian	37	They have no authority
antalever	38	Verbal authority
antaloupe	39	What is their authority
antar	40	What is your authority
antaro	41	Who is your authority
antata	42	With authority
ntation	43	With our authority
ntatory	44	With their authority
ntatrice	45	With your authority
nted	46	Without authority
nteen	47	Without our authority
nteens	48	Without their authority
nter	49	Without your authority

AUTHORITY 187

Code word C	Code No 187	Message or true reading.
		Authority—Continued
Canterbury	50	You have authority
Cantered	51	You have no authority
Cantering	52	Your authority
Canters	53	Authorization
Canthook	54	Authorizations
Canthus	55	Authorize
Canticle	56	Authorize them to
Canticoy	57	Authorize us to
Canting	58	Authorize you to
Cantingly	59	Do not authorize
Cantle	60	Do they authorize
Canto	61	Do you authorize
Canton	62	I authorize
Cantonal	63	They authorize
Cantoned	64	They will not authorize
Cantoning	65	To authorize
Cantonize	66	Will authorize
Cantonized	67	Will not authorize
Cantonizes	68	Will you authorize
Cantonment	69	Authorized
Cantons	70	Am authorized to
Cantor	71	Are authorized to
Cantoral	72	Are not authorized to
Cantoris	73	Are they authorized to
Cantors	74	Are we authorized to
Cantrap	75	Are you authorized to
Cantrip	76	Duly authorized
Cants	77	Is authorized
Canty	78	Is he authorized
Canvasback	79	Is not authorized
Canvass	80	No more authorized
Canvassed	81	Not authorized
Canvasser	82	Not authorized to
Canvasses	83	Properly authorized
Canvassing	84	They are authorized to
Canzone	85	They are not authorized to
Canzonet	86	Was authorized
Capa	87	Was not authorized
Capability	88	We are authorized to
Capable	89	We are not authorized to
Capacified	90	You are authorized
Capacifies	91	You are authorized to
Capacify	92	You are authorized to answer
Capacious	93	You are authorized to assure
Capacitate	94	You are authorized to convey
Capacities	95	You are authorized to state
Capacity	96	You are hereby authorized
Capapie	97	You are hereby authorized to
Caparison	98	You are not authorized
Caparisons	99	Authorizes

7494—12

State Department code book 1899, By ArnoldReinhold - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=82344363

Cipher

- "Series of transformations that converts **plaintext** to **ciphertext** using the Cipher **Key**." [**NIST-CSRC-G**]
- A cipher has three algorithms: [**BS23** #2.1]
  - *Key Generation* $k \sim U(K)$
  - *Encryption* $E(k, p) = c$
  - *Decryption* $D(k, c) = p$

Transposition cipher

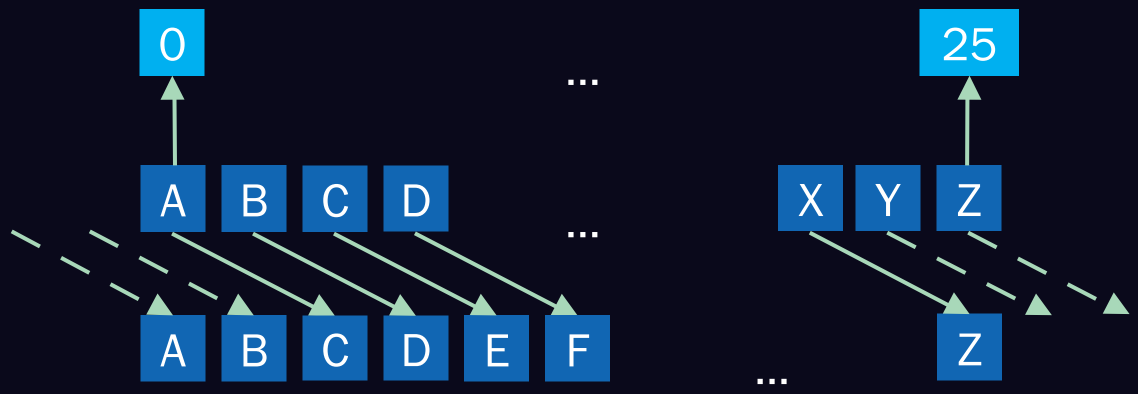
- Rearrange the order of plaintext letters according to a key. Columnar cipher, scytale

Key (word)	C	I	P	H	E	R
Key (order)	1	4	5	3	2	6
Plaintext	S	H	E	S	E	L
	L	S	S	E	A	S
	H	E	L	L	S	B
	Y	T	H	E	S	E
	A	S	H	O	R	E

Ciphertext SLHYA EASSR SELEO HSETS ESLHH LSBEE

Σκυτάλη CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1698345

Caesar cipher



- Encode letters to numbers, p_i
- Encryption: $E(p) = p + k \mod 26$
- Decryption: $D(c) = c - k \mod 26$
- ROT13: special case with $k = 13$
- ROT13 is its own inverse: $ROT(\,ROT(p, 13)\,, 13) = p$
  Try it yourself, e.g. rot13.com
- **Substitution** cipher: one letter is substituted for another
- **Mono-alphabetic** cipher: single substitution alphabet
- **Symmetric** cipher: the same key is used for E and D
- Still used by mobsters and terrorists.

“BA jihadist relied on Jesus-era encryption”

- “Islamic activists who were in touch with [An IT worker from British Airways jailed for 30 years for terrorism offences] had rejected the use of common modern systems such as PGP or TrueCrypt in favour of a system which used Excel transposition tables, which they had invented themselves. […]
- The single-letter substitution cipher they used was invented by the ancient Greeks and had been used and described by Julius Caesar in 55BC. […]
- Despite urging by the Yemen-based al Qaida leader Anwar Al Anlaki, Karim also rejected the use of a sophisticated code program called "Mujhaddin Secrets", which implements all the AES candidate cyphers, "because 'kaffirs', or non-believers, know about it so it must be less secure".”
- https://www.theregister.com/2011/03/22/ba_jihadist_trial_sentencing/, Tue 22 Mar 2011

FONDAZIONE BRUNO KESSLER CENTER FOR CYBERSECURITY
ALEPH APPLIED CRYPTOGRAPHY

Vulnerabilities

- Known plaintext, chosen plaintext, or educated guess -> key recovery

Vgnyl naabhaprq gur pybfher bs nyy aba-rffragvny ohfvarffrf nf vg snprq jung vgf cevzr zvavfgre pnyyrq gur pbhagel'f tenirfg zbzrag fvapr gur frpbaq jbeyq jne, jvgu qrnguf sebz pbebanivehf cnffvat 4,800 ba Fhaqnl.

Gur arj zrnfherf jrer bhgyvarq ol Tvhfrccr Pbagr va n yngr-avtug nqqerff gb gur pbhagel ba Fngheqnl, nsgre n erpbeq 793 crbcyr jrer xvyyrq ol gur ivehf, gur zbfg va n fvatyr qnl naljurer va gur jbeyq.

"Gur qrpvfvba gnxra ol gur tbireazrag vf gb pybfr qbja nyy cebqhpgvir npgvivgl guebhtubhg gur greevgbel gung vf abg fgevpgyl arprffnel, pehpvny, vaqvfcrafnoyr, gb thnenagrr hf rffragvny tbbqf naq freivprf," Pbagr fnvq.

The

For

And

V'z	I'm
Vg'f	It's
Jr'er	We're
Gurl'er	They're
'f	's

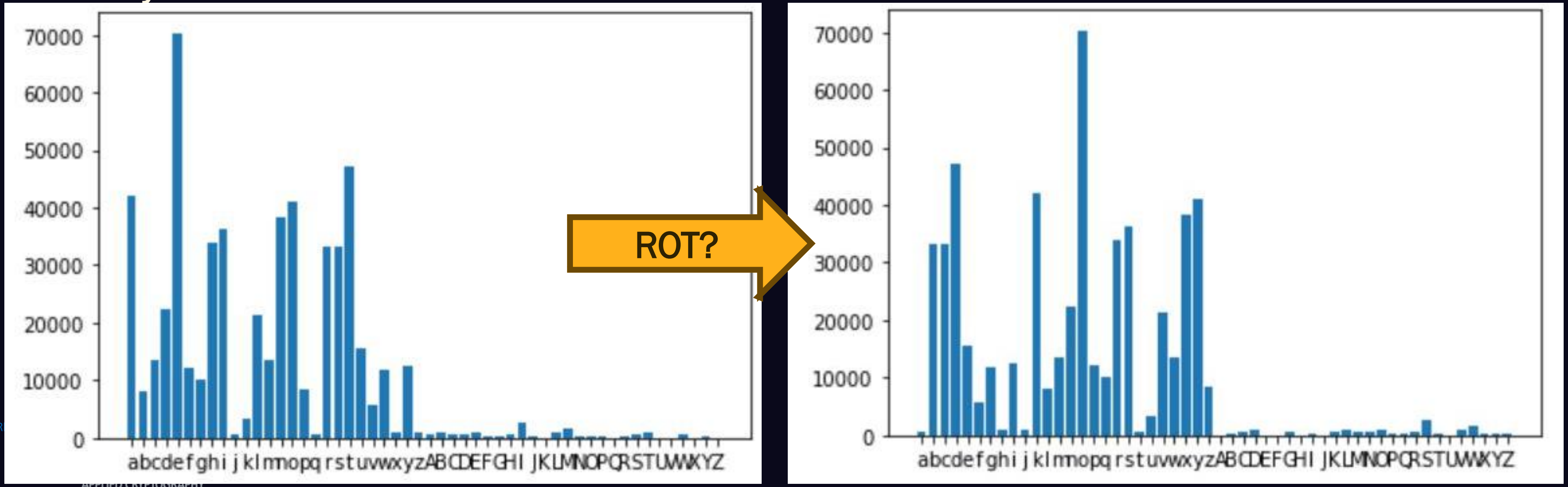
Vulnerabilities

- Known plaintext, chosen plaintext, or educated guess -> key recovery
- Brute force: the number of linear shifts is quite small (26-1).
- The number of **permutations** is more interesting:

$$26! = 403{,}291{,}461{,}126{,}605{,}635{,}584{,}000{,}000 \approx 10^{26}$$

Vulnerabilities

- Known plaintext, chosen plaintext, or educated guess -> key recovery
- Brute force: the number of linear shifts is quite small (26-1).
- Mono-alphabetic: one letter maps to one other. Large ciphertext - frequency analysis



Poly-alphabetic ciphers

- Use more than one alphabet to defeat frequency analysis
- 1467 Alberti: switch alphabet after several words, changing case in ciphertext
- 1553 Bellaso
- 1586 Vigenère
- 1861-1865 confederates "primarily relied upon three key phrases"
- 1863 Kasiski examination

https://en.wikipedia.org/wiki/Alberti_cipher

https://www.cryptomuseum.com/crypto/usa/ccd/index.htm

Vigenère

1586

- Key of length m; apply m Caesar ciphers and repeat
- $E(p_i, k) = (p_i + k_{i \bmod m}) \mod 26$

Key	ABCDABCDABCDABCDABCDABCDABCD
Plaintext	CRYPTOISSHORTFORCRYPTOGRAPHY
Ciphertext	CSASTPKVSIQUTGQUCSASTPIUAQJB

- Same letter in plaintext doesn't map to same letter in ciphertext, and vice versa
- Designed to defeat frequency analysis – except it doesn't

Kasiski examination

1863

- Find repeating sequences – they may not be of the same length
- Repetitions are caused by the same letter appearing at positions that are multiples of the key length

Key	ABCDABCDABCDABCDABCDABCDABCD
Plaintext	CRYPTOISSHORTFORCRYPTOGRAPHY
Ciphertext	CSASTPKVSIQUTGQUCSASTPIUAQJB

- Factor the distances; the most common factor is likely to be the key length m
- Group every m-th letter together; apply frequency analysis m times.

Modern cryptography

Kerckhoffs's principle

1883

- The encryption scheme is public
  - *Public scrutiny of a standard increases confidence and security*
- The key is secret
  - *chosen uniformly at random in a large space*
- A key, rather than an algorithm, is easier to manage:
  - *generate*
  - *keep secret (key storage)*
  - *change (key rotation)*
  - *exchange*
  - *destroy*

[NIST-CSRC-G]/term/key_management

XOR, OTP, and perfect secrecy

$\boldsymbol{a}$	0	0	1	1
$\boldsymbol{b}$	0	1	0	1
$\boldsymbol{c} = \boldsymbol{a} \oplus \boldsymbol{b}$	0	1	1	0

- Exclusive OR, XOR, $\oplus$
- Perfect secrecy:
  - *for any two plaintext (a_1, a_2), the probability of obtaining a ciphertext c is the same.*
- One-Time Pad (OTP): if a new random b is required for every a, then the length of the pre-shared key is the same as the message.
- Used properly, OTP is unconditionally secure.
- Can't we just use the same key again?

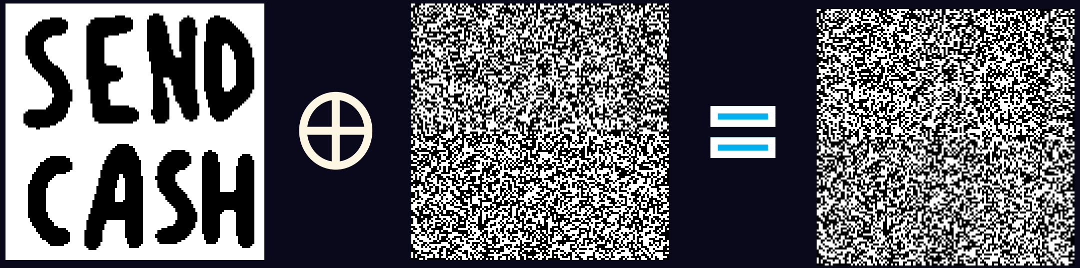
XOR, OTP, and perfect secrecy

a	0	0	1	1
b	0	1	0	1
c = a ⊕ b	0	1	1	0

- If the same key, b, is re-used:
  - $c_1 = a_1 \oplus b$
  - $c_2 = a_2 \oplus b$
  - $c_1 \oplus c_2 = a_1 \oplus b \oplus a_2 \oplus b$
- Recovering $a_1 \oplus a_2$ is bad enough.
- If an attacker
  - *knows* a_2, *known plaintext*
  - *or can influence* a_2, *chosen plaintext*
- they can recover the **key**, and every message encrypted with it.

XOR, OTP, and perfect secrecy

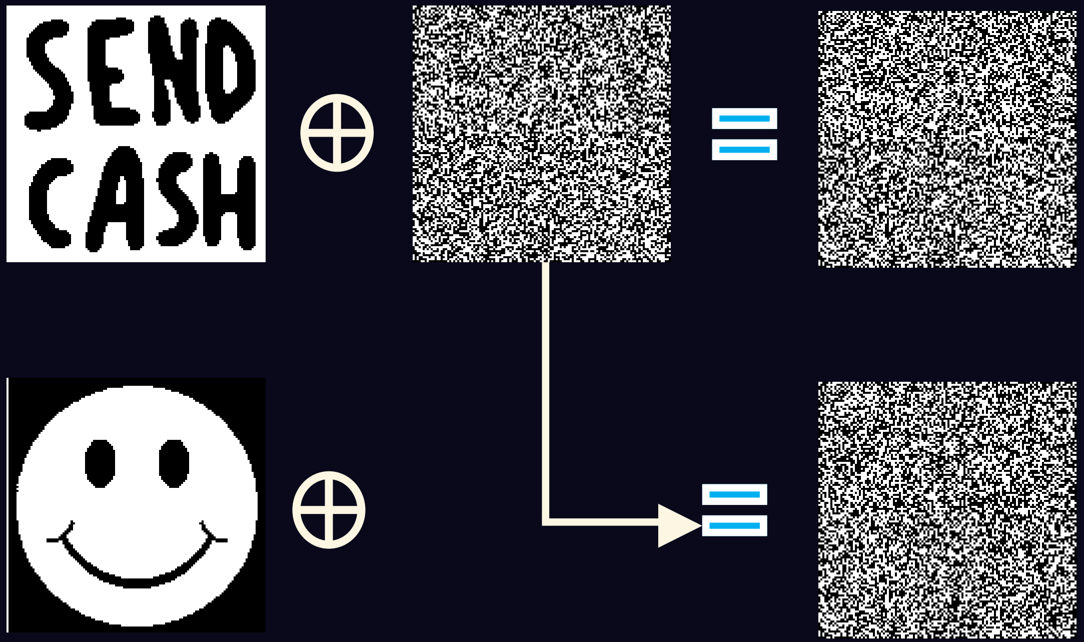
- With pictures:



https://crypto.stackexchange.com/questions/59/taking-advantage-of-one-time-pad-key-reuse

XOR, OTP, and perfect secrecy

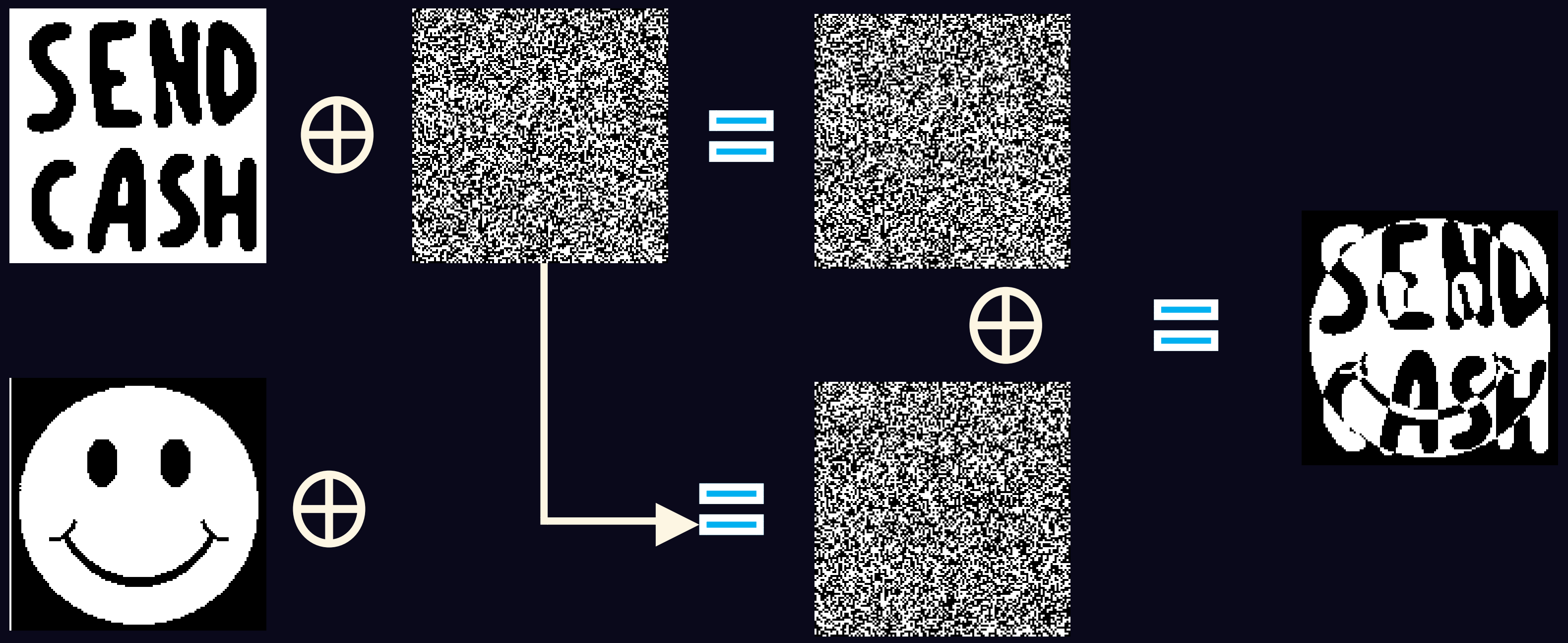
- With pictures:



https://crypto.stackexchange.com/questions/59/taking-advantage-of-one-time-pad-key-reuse

XOR, OTP, and perfect secrecy

- With pictures:



https://crypto.stackexchange.com/questions/59/taking-advantage-of-one-time-pad-key-reuse

OTP IRL: numbers station

[W#, B20]

- Both the US and the Soviet Union made use of encrypted shortwave radio transmissions to send messages to covert spies abroad during the cold war. “Numbers station” transmissions aimed at spies continue to this day.
  - *Shortwave radio **transmitters cover long distances**; it's possible for a single transmitter to get hemispheric or even global coverage.*
  - *Shortwave radio **receivers are readily available commercially** in almost every country and **are not usually suspicious to possess**.*
  - *Shortwave radio signal source is relatively easy to locate, but their wide coverage area makes it **very difficult to infer exactly who or where the intended recipients might be**.*
  - *OTP is **low tech**. It is possible to encrypt and decrypt OTP messages by hand with nothing more than paper and pencil and simple arithmetic.*
  - *OTP is **cumbersome**; you need a secret key as long as all the messages you will ever send, with no part of the key ever re-used for multiple messages. Typically, the key would be printed as a series of digits bound into a pad of paper, with each page removed after use; hence the name "one time pad".*
- Shortwave radio OTP typically consist of decimal digits broadcast in either a mechanically recorded voice or in morse code (more recently, digital transmissions are also used) on designated frequencies at designated times, usually in four- or five-digit groups (hence the term "numbers station").

4-8-15-16-23-42

OTP IRL: numbers station

[W#, B20]

- After copying and verifying a header in the message, the agent would remove the corresponding page from their secret OTP codebook and add each key digit to each corresponding message digit mod 10. The resulting plaintext digits are then converted to text with a simple substitution encoding.
- **Traffic analysis** might reveal to an observer the number of active agents or the volume of messages sent to them
- Numbers stations typically operate on rigidly fixed schedules, sending messages at pre-determined times, whether there is a real message to be sent or not. When there is no traffic for a given timeslot, **random fill traffic** is sent instead, indistinguishable to an outsider from real messages.
- In 2000-2010, the FBI found that Russian numbers were relayed by the Cuban shortwave numbers station
  - *Atención - - - 64202 - - - 65852 - - - 86321 Final, Final*
- Some time around 2005, messages, at random times, suddenly had no 9s (“nueve”)
- The FBI was able to tell when messages were and weren't being sent during the weekly timeslot when the suspect couple was observed in the room where they copied traffic. Empty message slots correlated perfectly with times that the suspect couple was traveling and not able to copy messages

Shannon’s principles

[S49]

- Confusion: each bit of the ciphertext should depend on several parts of the key, obscuring the connections between the two
  - *Make relationship between ciphertext and key as complex as possible*
  - *The key must be well distributed in the ciphertext*
  - *Every bit of the ciphertext should depend on every bit of the key*
- Diffusion: any plaintext structure should be spread over as much of the ciphertext as possible, to force the adversary to collect an infeasible amount
  - *Dissipates statistical structure of plaintext over bulk of ciphertext*
  - *The plaintext must be well distributed in the ciphertext*
  - *Every bit of the ciphertext should depend on every bit of the plaintext*
- Avalanche effect: changing one bit of the (plaintext, key) should potentially change every bit of the ciphertext with a probability of 50%

Symmetric Cryptography

- Symmetric cryptography:
  - *"A cryptographic algorithm that uses the same secret key for its operation and, if applicable, for reversing the effects of the operation (e.g., an AES key for encryption and decryption)." [**NIST-CSRC-G**]*
- Key Generation $k \sim U$
- Encryption $E(k, p) = c$
- Decryption $D(k, c) = p$

Asymmetric Cryptography

- Asymmetric cryptography:
  - *"Cryptography that uses two separate keys to exchange data, one to encrypt or digitally sign the data and one for decrypting the data or verifying the digital signature. Also known as public key cryptography" [**NIST-CSRC-G**]*
- Key Generation — public $k_1 \sim U$, secret $k_2 \sim U$
- Encryption / verification — $E(k_1, p) = c$
- Decryption / signature — $D(k_2, c) = p$

References

- [B20] "A Cryptologic Mystery – Did a broken random number generator in Cuba help expose a Russian espionage network?" Matthew Blaze, 18.09.2020. https://www.mattblaze.org/blog/neinnines/
- [BS23] A Graduate Course in Applied Cryptography. Dan Boneh, Victor Shoup, v0.6, 2023. https://toc.cryptobook.us/
- [NIST-CSRC-G] Computer Security Resource Center - Glossary csrc.nist.gov/glossary
- [S49] "Communication theory of secrecy systems", Claude E. Shannon, 1949. doi: 10.1002/j.1538-7305.1949.tb00928.x
- [W#] "Numbers station" https://en.wikipedia.org/wiki/Numbers_station

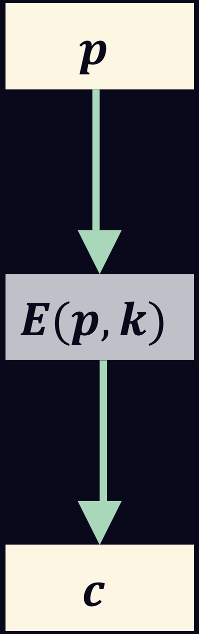
Block ciphers

Summary

- A block cipher operates on blocks of a fixed size
- Messages are usually longer than one block – ciphers have modes of operation
- Messages are rarely exact multiples – use padding schemes
- Both modes and padding can introduce vulnerabilities, even if the underlying cipher is as good as it gets

Block cipher

- Encryption

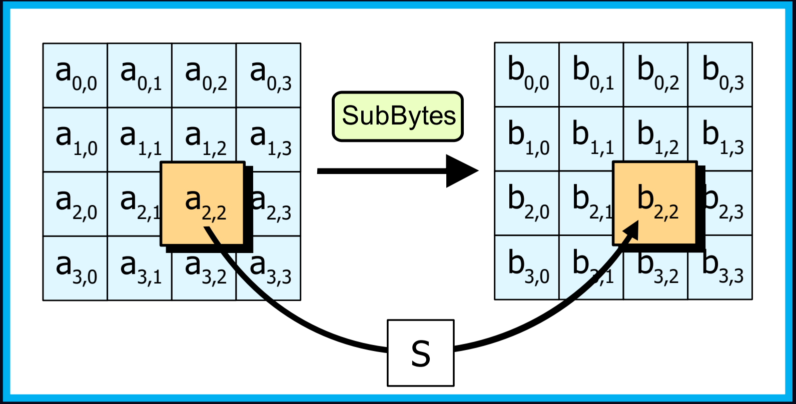


- Plaintext size n: block size
- keyed permutation:
  - *permutation over all n-bit strings (blocks)*
  - *key determines exactly which block is mapped to which.*
- round function iterated r times over the input message
- key schedule algorithm produces r subkeys from one master key.

AES round function

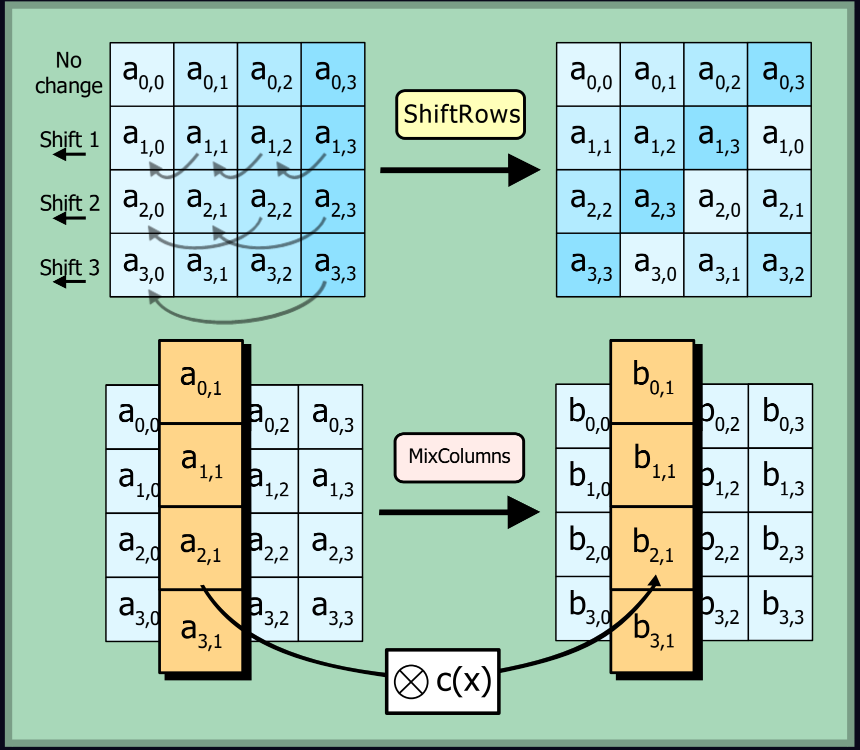
[FIPS 197]

Substitution



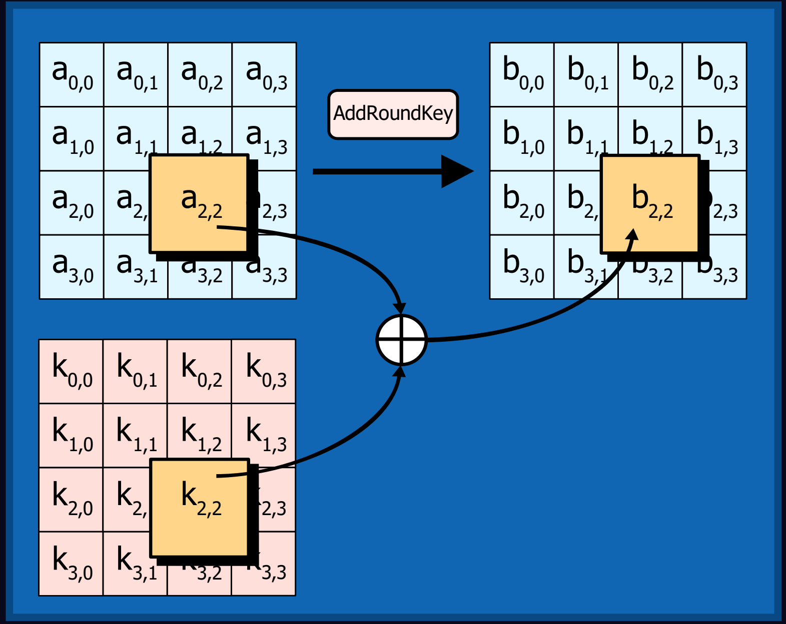
Multiplicative inverse
sum of rotations

Permutation



Multiplication by constant polynomial

Add round key

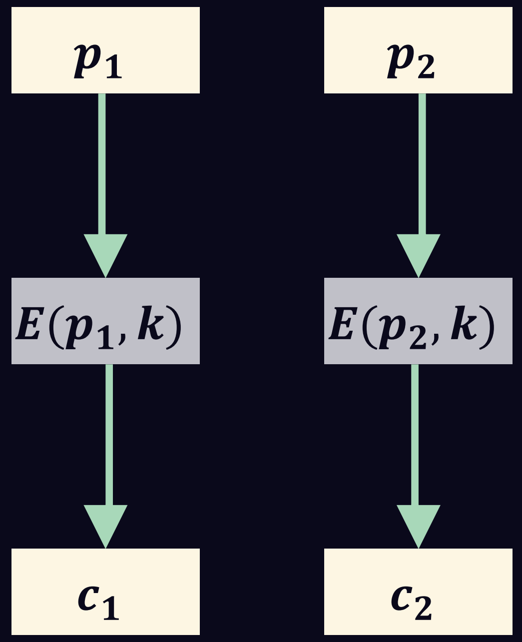


Repeat r rounds

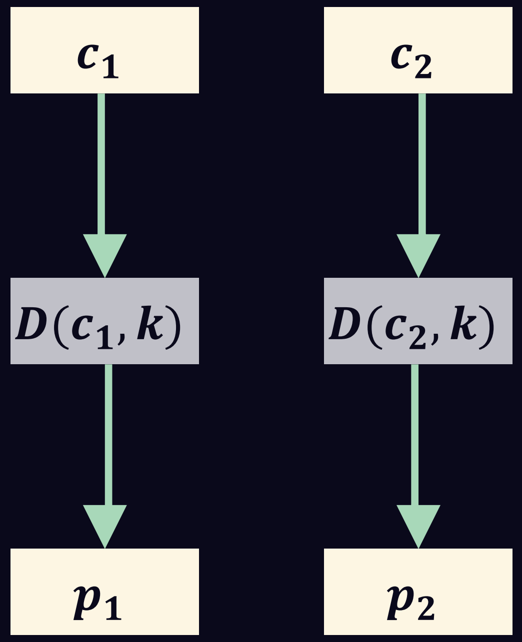
ECB – electronic codebook

[SP 800-38a]

- Encryption



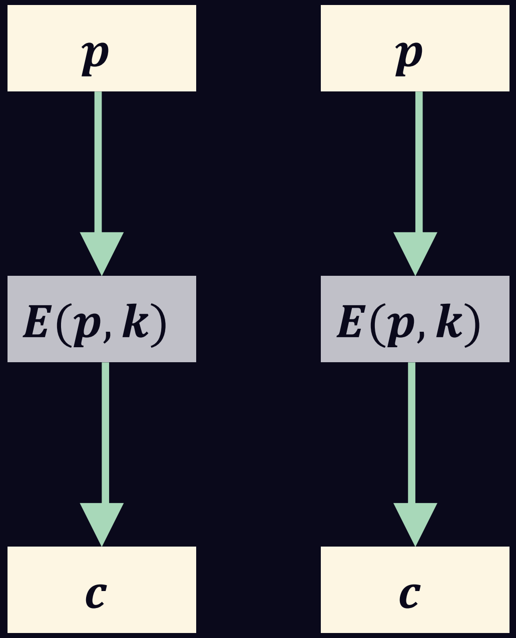
- Decryption



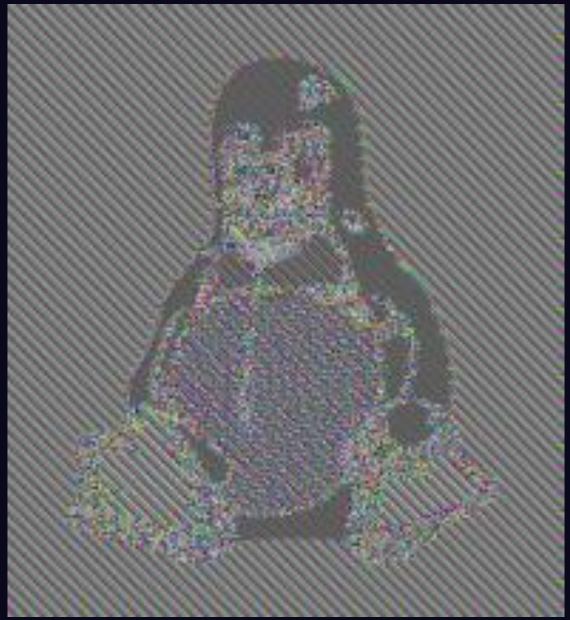
ECB – electronic codebook

[SP 800-38a]

- Encryption



- Does not ensure confidentiality of entire plaintext, only single blocks



Tux the Penguin, the Linux mascot. Larry Ewing [*lewing@isc.tamu.edu]* with The GIMP.

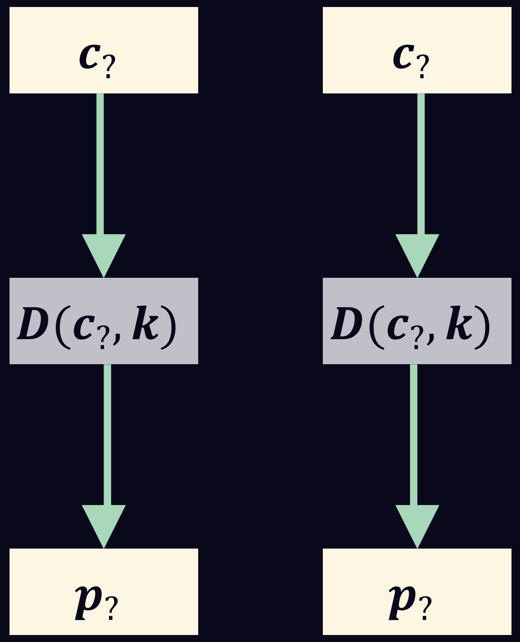
FONDAZIONE BRUNO KESSLER CENTER FOR CYBERSECURITY

ALEPH APPLIED CRYPTOGRAPHY

ECB – electronic codebook

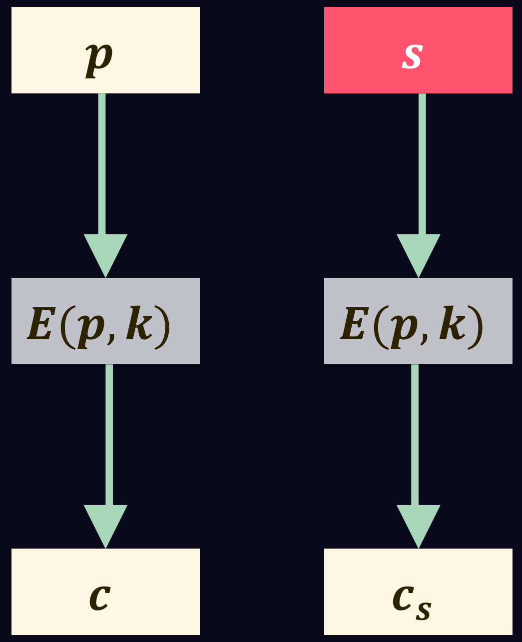
[SP 800-38a]

- Decryption



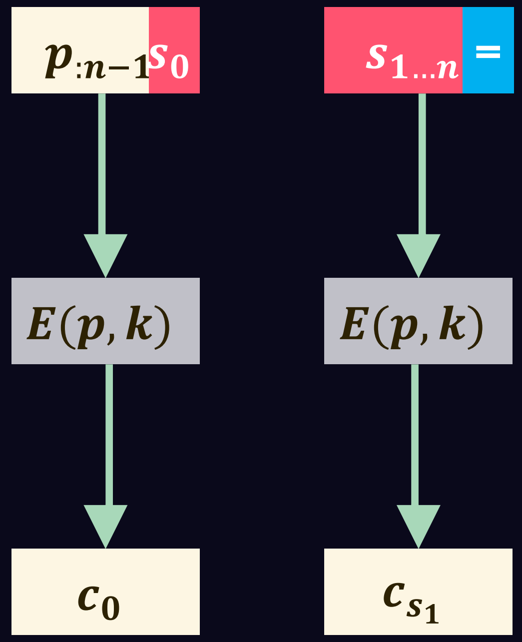
- Does not ensure confidentiality of entire plaintext, only single blocks
- Does not preserve order – attackers can shuffle ciphertext
- Does not protect against replay – attackers can duplicate or insert from other ciphertext encrypted with the same key
- ECB does not guarantee integrity

ECB oracle



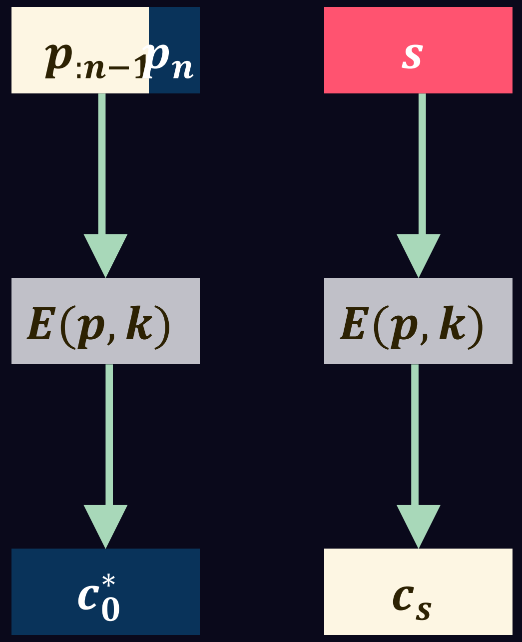
- Suppose:
  - *plaintext* p *chosen by attacker,*
  - *fixed secret* s *set by server,*
  - *the oracle always provides* $c = ECB(k, p||s)$

ECB oracle attack



- Suppose:
  - *plaintext p chosen by attacker,*
  - *fixed secret s set by server,*
  - *the oracle always provides* $c = ECB(k, p||s)$
- Shorten p by one byte, obtain c_0

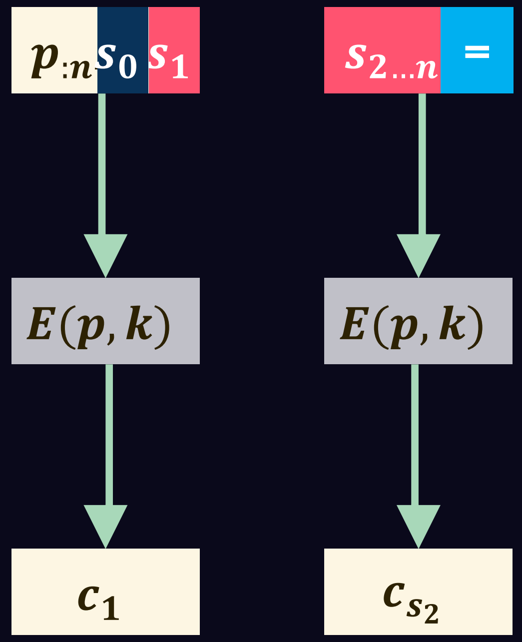
ECB oracle attack



- Suppose:
  - *plaintext p chosen by attacker,*
  - *fixed secret s set by server,*
  - *the oracle always provides $c = ECB(k, p||s)$*
- Shorten p by one byte, obtain c_0
- Brute force s_0:
  - *256 possible chosen plaintext p_n*
  - *up to 256 oracle output c_0^**

FONDAZIONE BRUNO KESSLER CENTER FOR CYBERSECURITY ALEPH APPLIED CRYPTOGRAPHY

ECB oracle attack



- Suppose:
  - *plaintext p chosen by attacker,*
  - *fixed secret s set by server,*
  - *the oracle always provides* $c = ECB(k, p||s)$
- Shorten p by one byte, obtain c_0
- Brute force s_0
- Shorten p by another byte, obtain c_1
- Brute force s_1 as before, repeat

ECB oracle attack power

- Guessing the whole key by brute force: # possibilities (worst case)

$$2^{128} = 256^{16} \approx 3.4 \cdot 10^{38}$$

- ECB oracle: # possibilities (worst case)

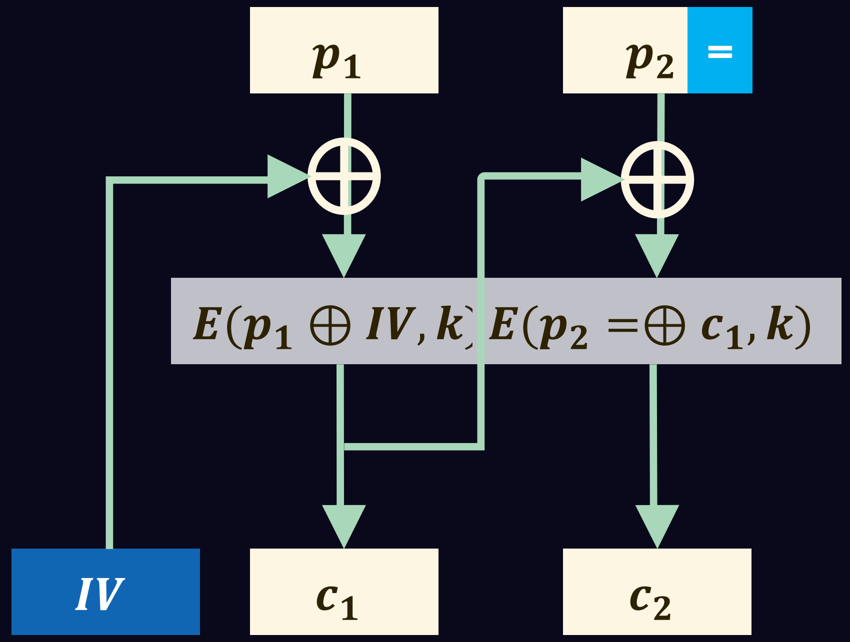
$$256 \cdot 16 = 4096$$

- on **any** 128-bit block cipher.

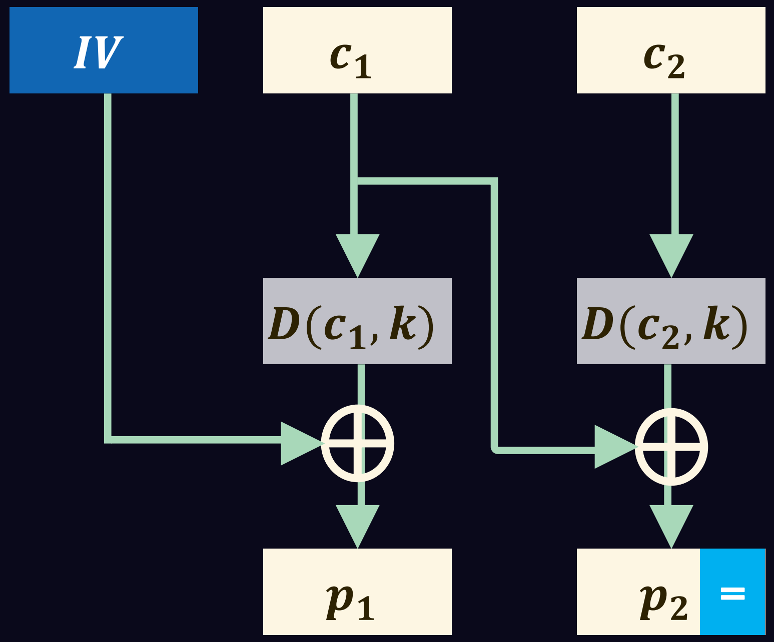
CBC – cipher block chaining

[SP 800-38a]

- Encryption



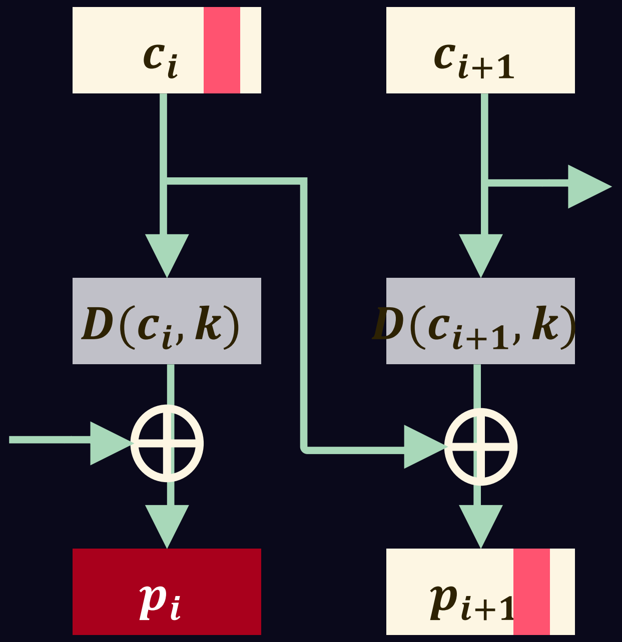
- Decryption



CBC is malleable

- An attacker who modifies a specific single bit of ciphertext c_i can flip a specific single bit of the next plaintext p_{i+1}
- Does not require knowledge of p, k
- Does require ability to **intercept** ciphertext
- Results in garbage decryption of c_i
- CBC does not guarantee **integrity**

- Decryption



Padding schemes

- Block size b bytes, e.g. DES: $b = 8$
- Plaintext length: $kb + r$
- PKCS#7 [**CMS**]: append $b - r$ bytes, each of value `0x(b-r)`
- ISO/IEC 7816-4: append all zero bytes, except leftmost `0x80`
- [**SSLv3.0**] append rightmost byte of value `0x(b-r-1)`
- Special case: $r = 0$

pp	pp	pp	pp	pp	03	03	03
pp	pp	pp	pp	pp	80	00	00
pp	pp	pp	pp	pp	??	??	02
??	??	??	??	??	??	??	15

Padding oracle attack

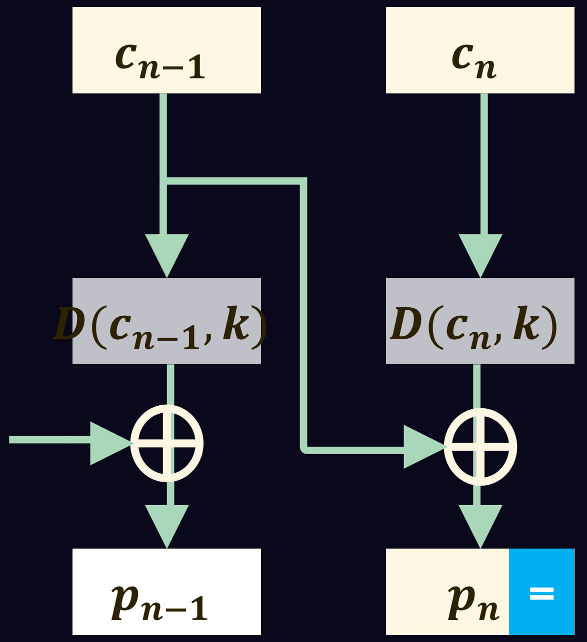
- Padding oracle: the server responds differently based on (in)correct padding.
- The oracle can be OK/KO codes and/or differences in timing.
- An attacker can modify the decrypted message padding until the response changes. This reveals information about the plaintext.
- Shown against CBC in
  - *SSL [**V02**]*
  - *SSL 3.0 [**POODLE**] [**CVE-2014-3566**]*
  - *TLS 1.2 [**Lucky13**]*

Padding oracle attack

[V02]

- Given a message with valid padding

- Decryption

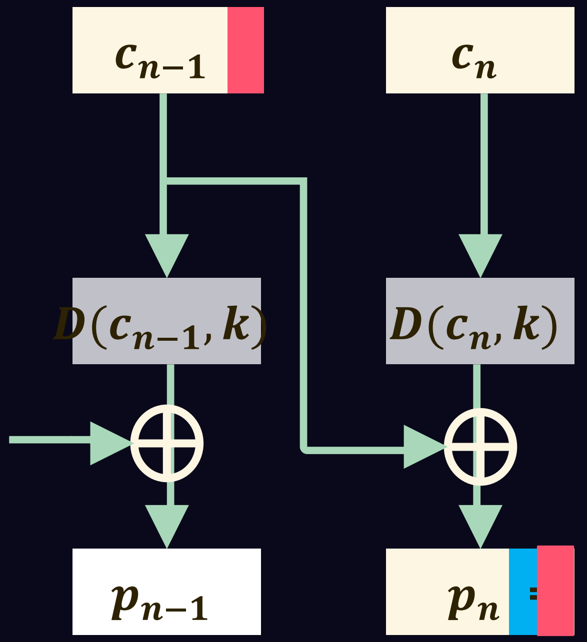


Padding oracle attack

[V02]

- Given a message with valid padding
- Modify the last byte of the second-to-last block: $(c_{n-1})_b \bigoplus_{j=0}^{255} j$

- Decryption

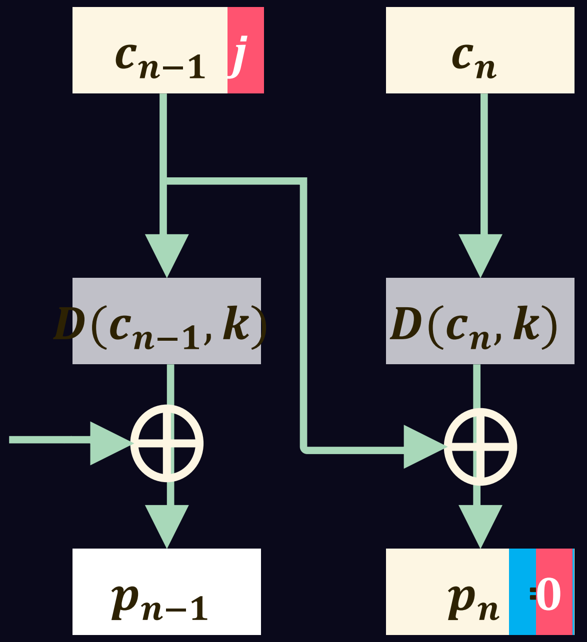


Padding oracle attack

[V02]

- Given a message with valid padding
- Modify the last byte of the second-to-last block: $(c_{n-1})_b \bigoplus_{j=0}^{255} j$
- In at least one case j, valid PKCS#7 padding:
  - $00 = D(c_n, k)_b \oplus (c_{n-1})_b \oplus j$

- Decryption

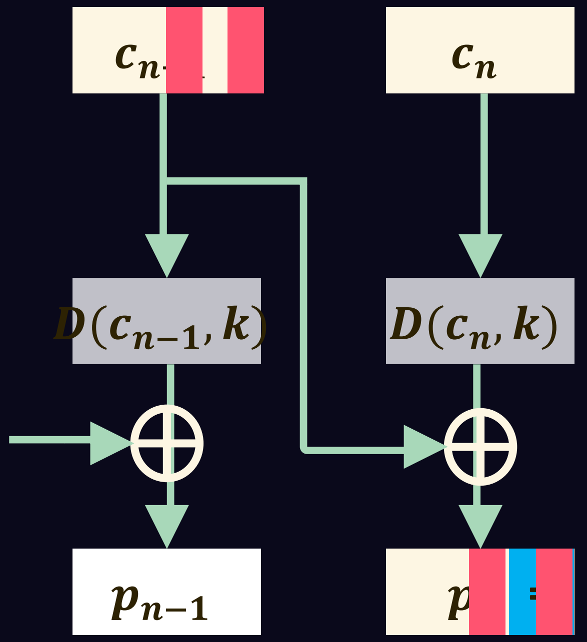


Padding oracle attack

[V02]

- Given a message with valid padding
- Modify the last byte of the second-to-last block: $(c_{n-1})_b \bigoplus_{j=0}^{255} j$
- In at least one case, valid PKCS#7 padding:
  - $00 = D(c_n, k)_b \oplus (c_{n-1})_b \oplus j$
- Rule out other cases by iterating from left, e.g. case `04 04 04 04` can be ruled out by modifying byte $b - 3$

- Decryption

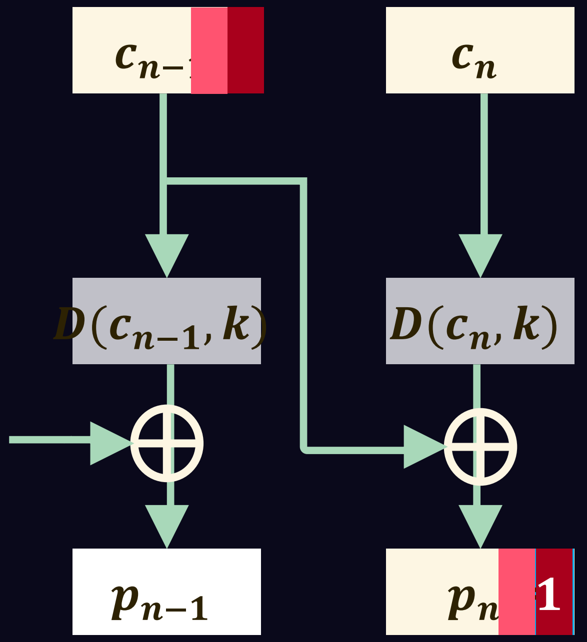


Padding oracle attack

[V02]

- Given a message with valid padding
- Modify the last byte of the second-to-last block: $(c_{n-1})_b \bigoplus_{j=0}^{255} j$
- In at least one case, valid PKCS#7 padding:
  - $00 = D(c_n, k)_b \oplus (c_{n-1})_b \oplus j$
- Once you know which j_b results in a padding of 00, add $j_b + 1$ to the last byte and repeat with byte $b - 1$, etc.

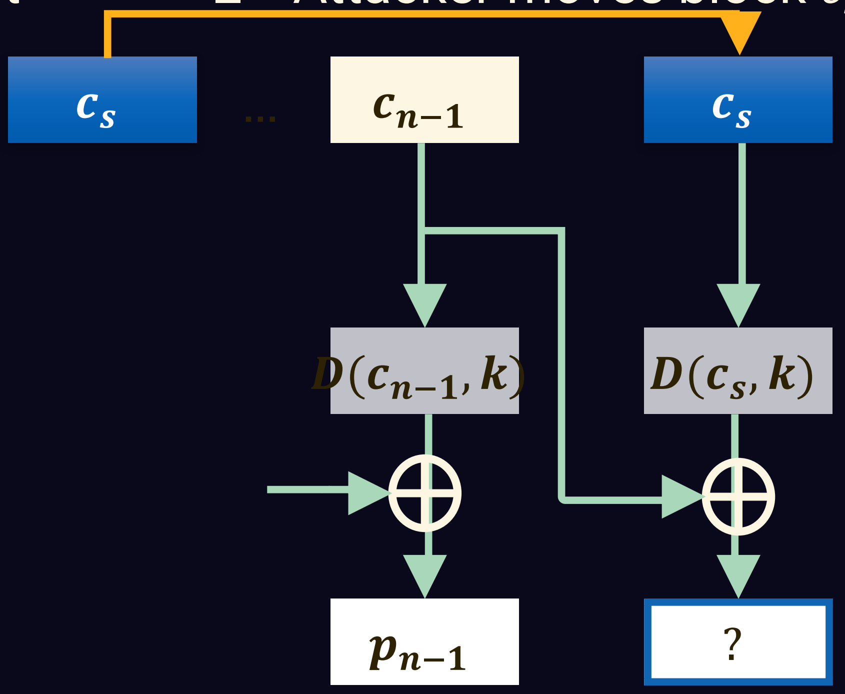
- Decryption



Padding oracle attack – last block?

- Suppose there is a valuable secret S hidden in an earlier block c_s
- **Attacker capability**: intercept the message and overwrite c_n with c_s

- Attacker moves block c_s to end



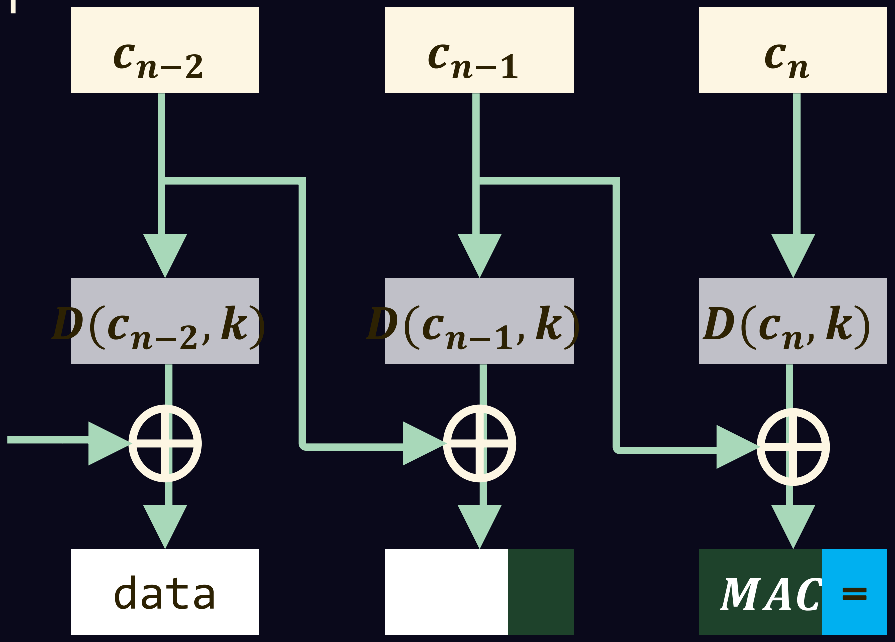
MAC, padding, and CBC

- MAC: A family of cryptographic functions parameterized by a symmetric key k.
  - *can act on input data of variable length to produce an output value of a specified length. The output value is called the MAC of the input message.*
  - *Without knowledge of k, it must be computationally infeasible to predict the (as-yet-unseen) value of $MAC(k, x)$ even if one has already seen $MAC(k, x_j)$ for other messages $x_j \neq x$*
  - *[**NIST-CSRC-G**]/term/message_authentication_code_algorithm*
- SSL integrity protection: MAC-then-encrypt, then send.

MAC, padding, and CBC

- Upon reception: first decrypt, then check if padding is correct, then check MAC:
  - *If padding incorrect, raise error*
  - *Else if MAC incorrect, raise (possibly different) error*
  - *Else respond with success, interpret data*

- Decryption



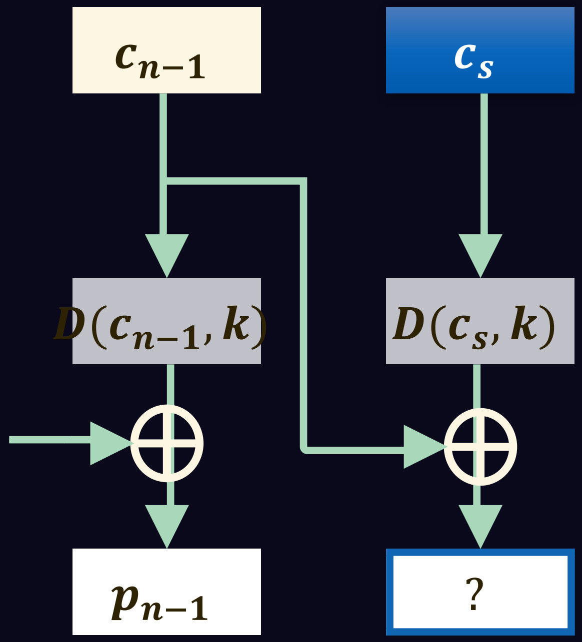
Padding oracle attack

[POODLE]

- Suppose a genuine encrypted message is sent, such that:
  - *The last block c_n is entirely SSL 3.0 padding*
  - *there is a valuable secret S hidden in block c_s*
- **Attacker capability 1**: intercept the message and swap c_s with c_n
- The last decrypted block will be

$$D(c_s, k) \oplus c_{n-1}$$
$$= S \oplus c_{n-1}$$

- Attacker moves block c_s to end

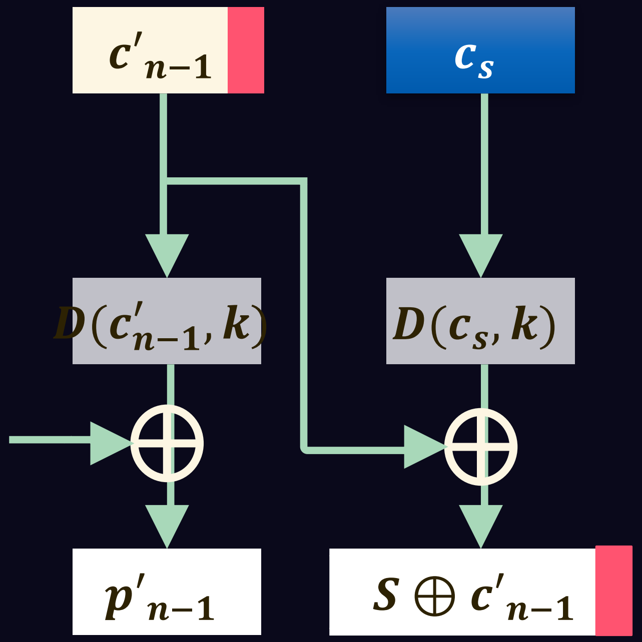


Padding oracle attack

[POODLE]

- If the last byte of $S \oplus c_{n-1}$ is equal to `07` exactly, the server follows the success protocol conditions; otherwise it fails (somehow)
- The attacker retries such that the last byte of c_{n-1} is different.
- Recovering 1 byte costs on average 128 attempts

- Attacker repeats



Padding oracle attack

[POODLE]

- Suppose **attacker capability 2**: modify some of the original plaintext, before and after the secret (request path and payload) – without knowing the secret (cookie)
- Next byte: change URL path to shift cookie forward by 1 byte, so the last byte of c_s is the second-to-last byte of the cookie

`GET / HT`	`TP/1.1\r\n`	`Cookie: `	`abcdefgh`	`\r\n\r\nXXXX`	`MAC data`	`???????7`
`GET /a H`	`TTP/1.1\r`	`\n Cookie:`	` abcdefg`	`h\r\n\r\nXXX`	`MAC data`	`???????7`
			c_s			

- Ideal difficulty of guessing an m-byte secret ~ $2^{8 \cdot m}$
- Difficulty by padding oracle ~ $m \cdot 2^8$

Example by Adam Langley
https://www.imperialviolet.org/2014/10/14/poodle.html

SSL 3.0 example

[RFC 6101]

- Simplified https example:

```
POST /path HTTP/2.0
Host: www.example.org
Cookie: name=value
Payload
MAC
padding
```

- After padding up to block size, encrypt

- Suppose someone is using a browser to visit a genuine website.
- If the login is successful, they get a cookie.
- The browser keeps the cookie secret, but every time it contacts the website it will include the cookie in the request, without telling you.
- If someone steals that cookie, they can pretend to be you.

SSL 3.0 example

[RFC 6101]

- Simplified https example:

```
POST /path HTTP/2.0
Host: www.example.org
Cookie: name=value
Payload
MAC
padding
```

- After padding up to block size, encrypt

- Visiting a malicious website can force your browser to make repeated requests to the genuine one: the attacker controls this malicious request, except the **secret**, so the attacker can add garbage to the path and the payload.

adaptive chosen plaintext

Padding Oracle On Downgraded Legacy Encryption

[POODLE]

- [**CVE-2014-3566**] "The SSL protocol 3.0, as used in OpenSSL through 1.0.1i and other products, uses nondeterministic CBC padding, which makes it easier for man-in-the-middle attackers to obtain cleartext data via a padding-oracle attack, aka the "POODLE" issue."
- is a **protocol** vulnerability on SSL v3, but many related ones on TLS are due to specific implementations - 2014-8730, 2015-8313, 2015-3642, 2015-4078
- Uses techniques from the [**BEAST**] attack [**CVE-2011-3389**].
- [**POODLE**]: "run a JavaScript agent on evil.com (or on http://example.com) to get the victim's browser to send cookie-
bearing HTTPS requests to https://example.com, and
intercept and modify the SSL records sent by the browser in such a way that there's a nonnegligible chance that example.com will accept the modified record."

References

- [**BEAST**] "Here come the ⊕ ninjas." T. Duong, J. Rizzo, Unpublished manuscript, 2011. url: https://scholar.google.com/scholar?oi=bibs&hl=en&cluster=178124520586567 9782
- [**CMS**] https://tools.ietf.org/html/rfc5652#section-6.3
- [**CVE-2011-3389**] nvd.nist.gov/vuln/detail/CVE-2011-3389
- [**CVE-2014-3566**] url: nvd.nist.gov/vuln/detail/CVE-2014-3566
- [**FIPS 197**] "Advanced Encryption Standard (AES)" doi: 10.6028/NIST.FIPS.197. url: csrc.nist.gov/publications/detail/fips/197/final

References

- [**Lucky13**] "Lucky Thirteen: Breaking the TLS and DTLS Record Protocols". N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. N. Schuldt. USENIX Security 2013.
- [**NIST-CSRC-G**] Computer Security Resource Center - Glossary csrc.nist.gov/glossary
- [**POODLE**] url: www.openssl.org/~bodo/ssl-poodle.pdf

References

- [**SP 800-38a**] "Recommendation for Block Cipher Modes of Operation: Methods and Techniques". doi: 10.6028/NIST.SP.800-38A. url: csrc.nist.gov/publications/detail/sp/800-38a/final
- [**SSLv3.0**] Secure Sockets Layer version 3 - RFC 6101 tools.ietf.org/html/rfc6101
- [**V02**] [*Security Flaws Induced by CBC Padding Applications to SSL, IPSEC, WTLS...*]() S. Vaudenay, EUROCRYPT 2002. doi: [10.1007/3-540-46035-7_35](10.1007/3-540-46035-7_35). url: https://www.iacr.org/cryptodb/data/paper.php?pubkey=2850

Cryptographic Hash functions

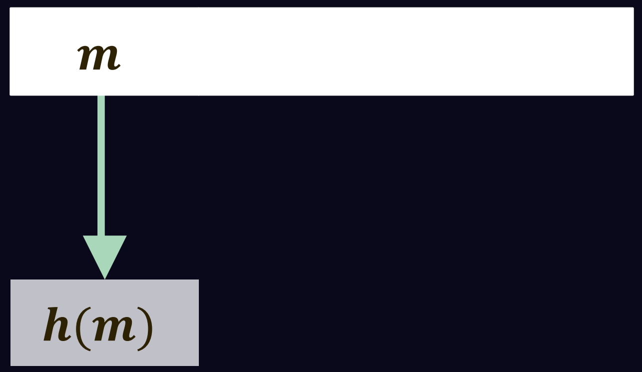
FONDAZIONE BRUNO KESSLER
CENTER FOR CYBERSECURITY
ALEPH
APPLIED CRYPTOGRAPHY

Summary

- Merkle-Damgård construction, MD5
- Cryptographic properties – resistance to attacks
- Birthday problem and collisions
- Message authentication and length extension attack
- Applications

Hash functions

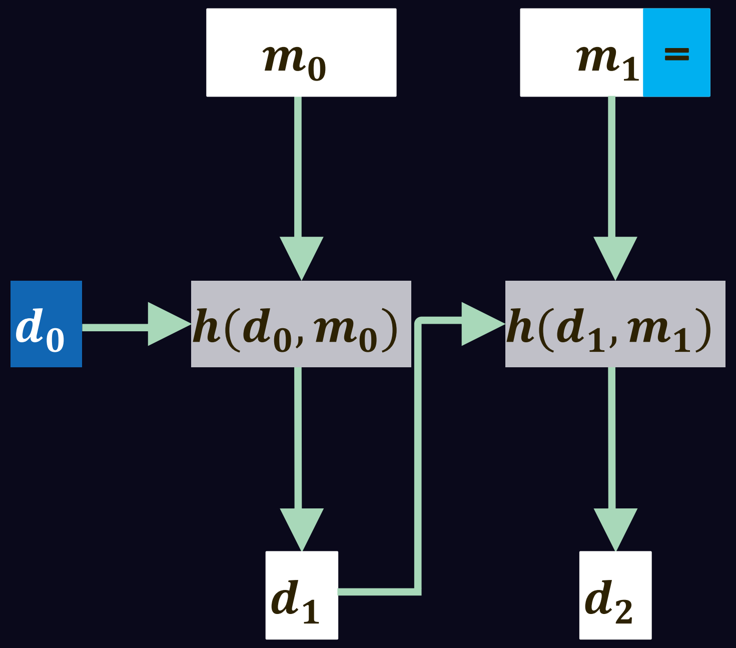
- The output of a hash function is known as digest: $d = h(m)$, a.k.a. "fingerprint" or "thumbprint".
- Used to guarantee **integrity** of a message or file, e.g. OS distributions help.ubuntu.com/community/UbuntuHashes
  - *Input message **usually** much longer than digest*
  - *Smallest change in input leads to large change in output*
  - *Digests are very quick to compute in software.*
- Cryptographic hash functions are designed to give output computationally indistinguishable from random.



Merkle-Damgård construction

[M79]

- Concatenate use of compression function h on message blocks
- MD5, SHA-1, SHA-2
- d_0 pre-defined constant



MD5

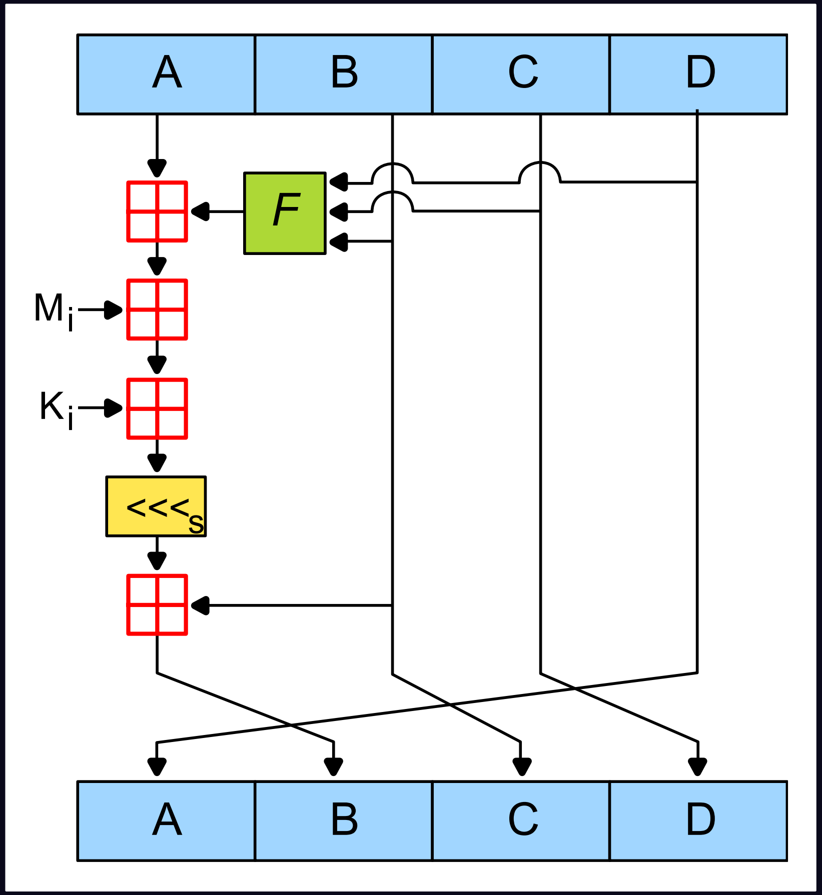
[RFC 1321]

- Block size: 512 (16 32-byte words)
- Digest size: 128 (4 32-byte words)
- Last block padding: append bit
  $$p = 10 \dots 0$$
  such that
  $$len(m||p) \ mod \ 512 = 448$$
  and finally append $len(m)$
- Special case: $len(m) \ mod \ 512 = 0$
  The entire last block is padding

MD5

[RFC 1321]

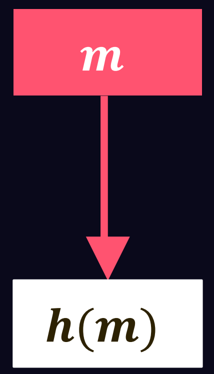
- Initialize 4 digest word registers A:D, of 32 bits each
- for 4 rounds:
  - *for each message block word M[1:16], of 32 bits each:*
    - F nonlinear function
    - K standard constants
    - $<<<_s$ s-bit left-ROT
    - $\boxplus$ addition modulo 2^{32}



MD5 by Surachit CC BY-SA 3.0, commons.wikimedia.org/w/index.php?curid=2652649

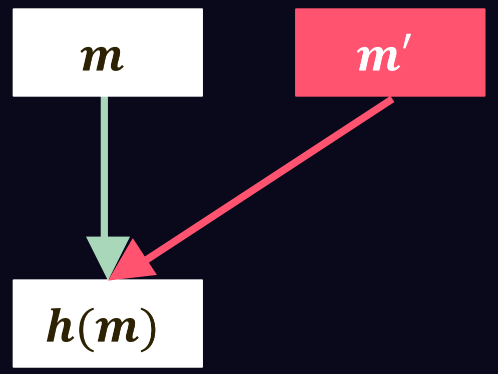
Cryptographic hash functions

- **Preimage**: given d, find **any** m such that $h(m) = d$



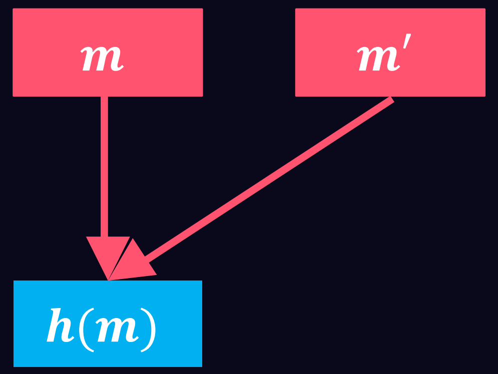
Cryptographic hash functions

- **Preimage**: given d, find **any** m such that $h(m) = d$
- **2nd preimage**: given m, find **any** $m' \neq m$ such that $h(m) = h(m')$



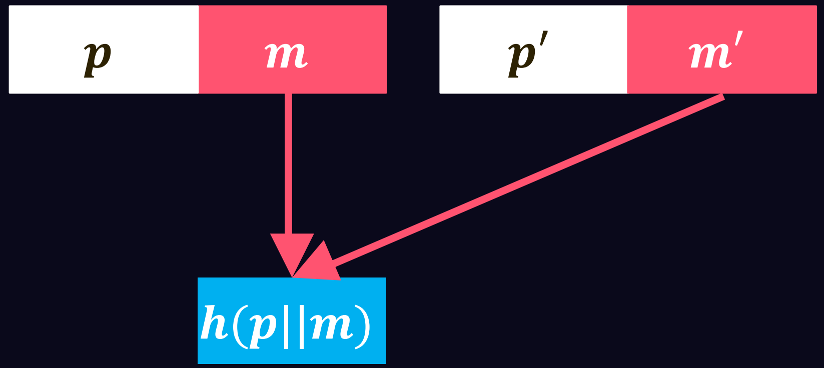
Cryptographic hash functions

- **Preimage**: given d, find **any** m such that $h(m) = d$
- **2nd preimage**: given m, find **any** $m' \neq m$ such that $h(m) = h(m')$
- **Collision**: find **any two** $m, m' \neq m$ such that $h(m) = h(m')$ (not known)



Cryptographic hash functions

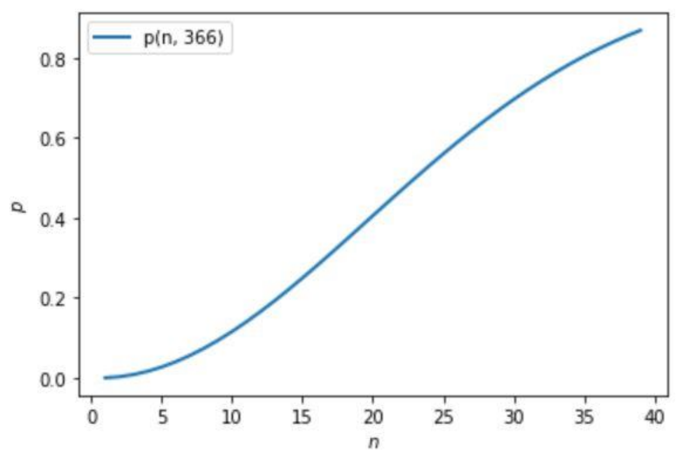
- **Preimage**: given d, find **any** m such that $h(m) = d$
- **2nd preimage**: given m, find **any** $m' \neq m$ such that $h(m) = h(m')$
- **Collision**: find **any two** $m, m' \neq m$ such that $h(m) = h(m')$
- **Chosen prefix collision**: given prefixes p, p', find m, m' such that $h(p||m) = h(p'||m')$



Birthday problem

- How many random strings would you have to hash, until you found a collision?
- This result is an upper bound against brute-force collision for any hash function – to the extent that digests are indistinguishable from random numbers.
- A sequence of integers $I = \{i\}_{j=1}^{n}$ are drawn uniformly at random with replacement in the range $[1, d]$. Let the probability that any two of these integers are equal be

$$\mathbb{P}(i_j = i_k) \approx 1 - e^{\frac{-n(n-1)}{2d}} =: p(n, d)$$



- The number of random integers required to obtain a given p is

$$n(p, d) = \left[2d \ln\left(\frac{1}{1-p}\right)\right]^{1/2}$$

- Example: in a class of $n = 25$ people with $d = 366$ possible birthdays, the probability of at least two students having the same birthday is $\approx .57$ (check on WolframAlpha).

Collision

- [**WY05**] collision: an algorithm that can find **two different sequences** of 128 bytes (two 512-bit blocks) with the same MD5 hash, given any starting state d_0

```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
```

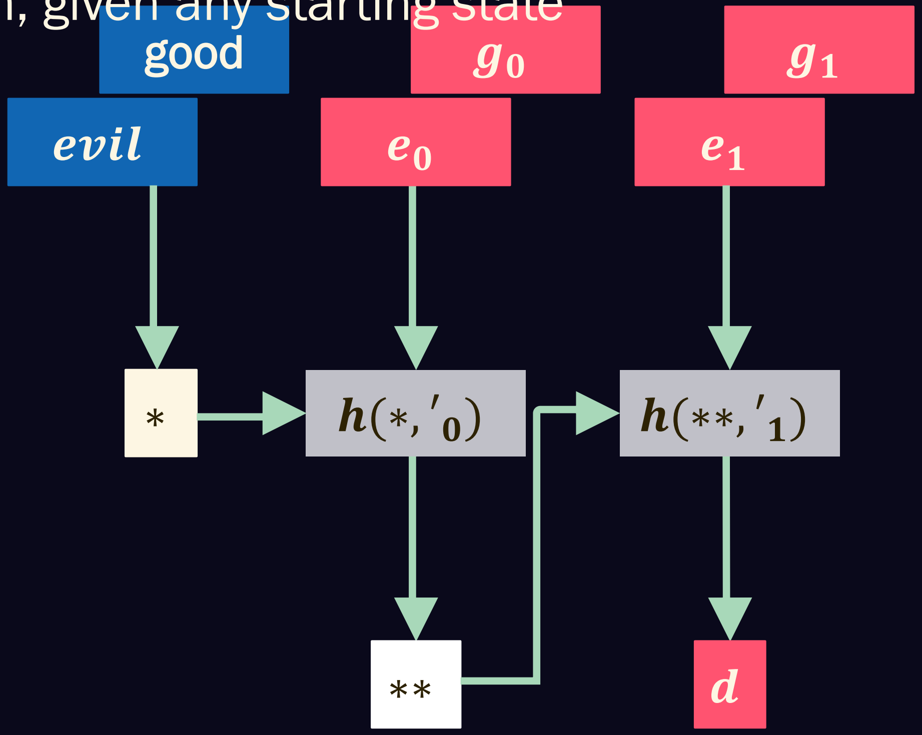
```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70
```

FONDAZIONE BRUNO KESSLER CENTER FOR CYBERSECURITY

ALEPH APPLIED CRYPTOGRAPHY

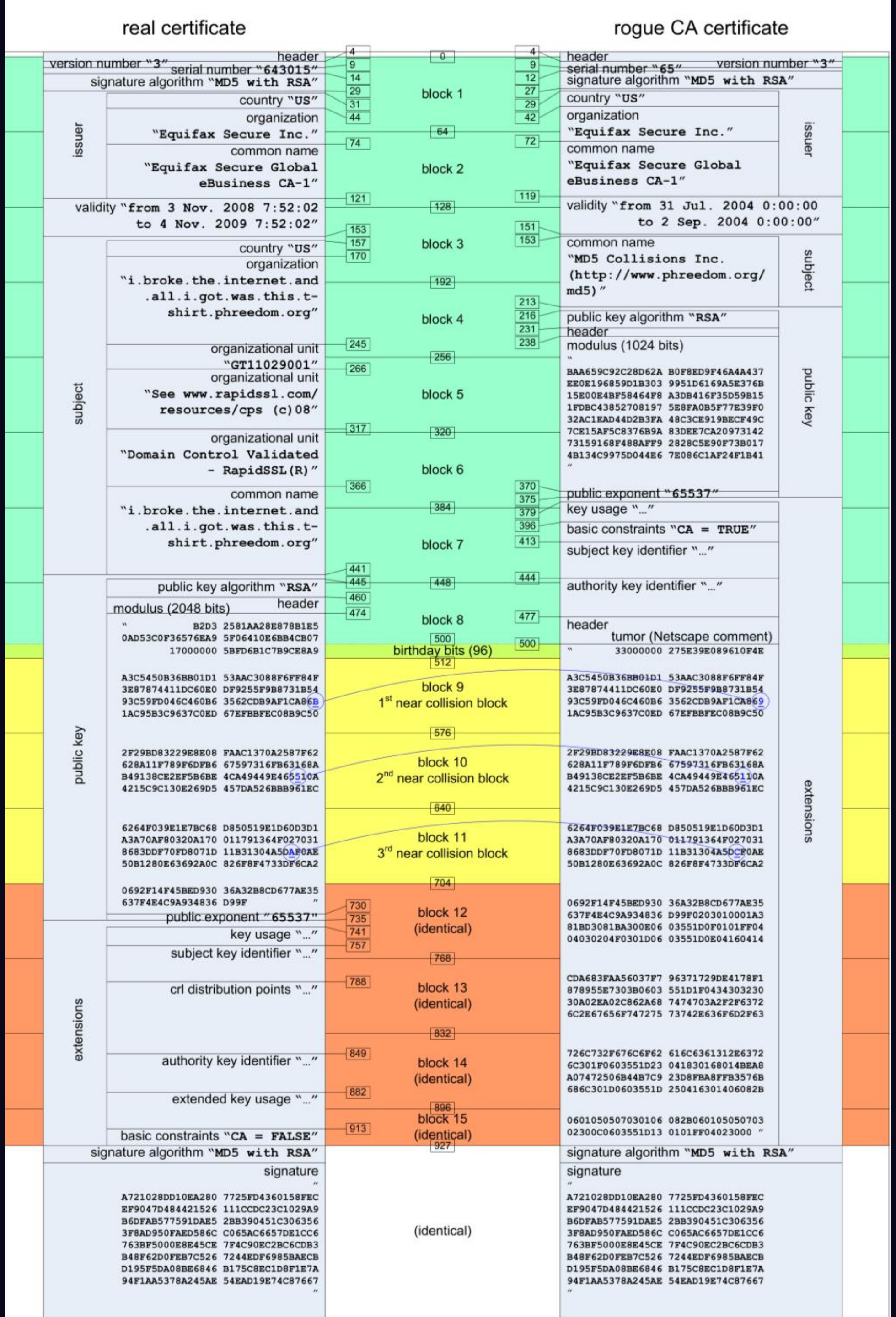
Chosen prefix collision

- [WY05] collision: an algorithm that can find **two different sequences** of 128 bytes (two 512-bit blocks) with the same MD5 hash, given any starting state
- [SLdW07] **chosen prefix** collision allows **completely arbitrary files** to have the **same MD5 hash**, by appending a few thousand bytes at the end of each file (≥2 blocks).
- MD5 collision demo: https://www.mscs.dal.ca/~selinger/md5collision/



Example: rogue CA certificate

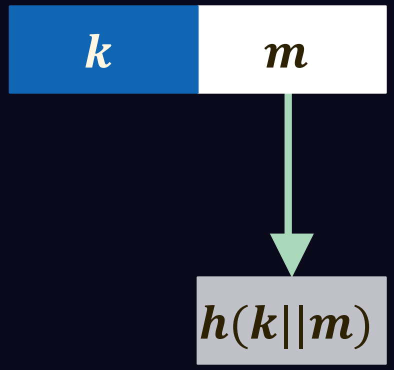
- X.509 certificates tie digital identities to public/private key pairs
- Issuer digital signature over the **hash of** Certificate Signing Request
- In 2008, 6 CAs issued certificates signed with MD5.
- [**hashclash**], 2009: craft two CSR – one legitimate, one rogue CA – with same MD5; get a CA to sign the legitimate; copy the signature to the rogue. win.tue.nl/hashclash/rogue-ca/
- “basic constraints” include a bit indicating whether it is a CA or a user certificate (“CA = TRUE” or “FALSE”)
- SHA-1 chosen prefix has been demonstrated. [**LP20**]



FONDAZIONE BRUNO KESSLER · CENTER FOR CYBERSECURITY · ALEPH APPLIED CRYPTOGRAPHY

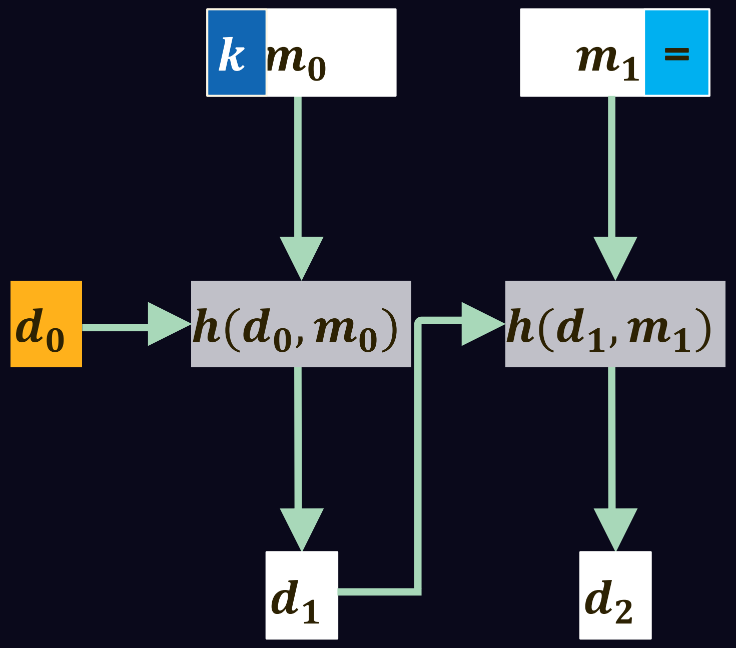
Hash-based Message Authentication Codes

- Intuition (not secure):
- Suppose two have the same pre-shared secret key (symmetric)



Length extension attack

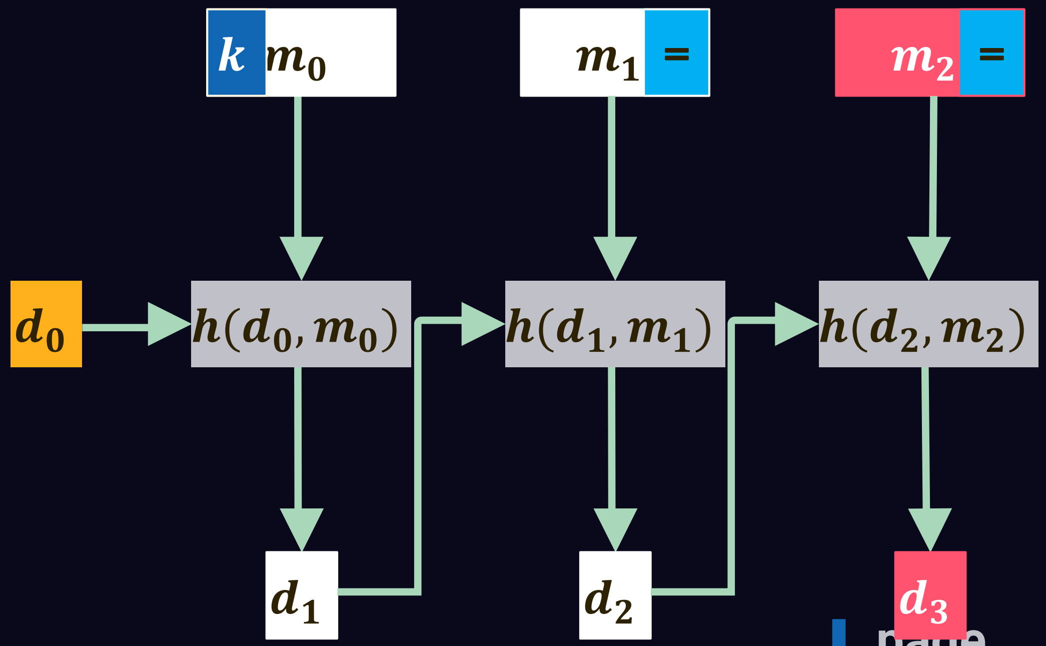
- Recall Merkle-Damgård



FONDAZIONE BRUNO KESSLER CENTER FOR CYBERSECURITY
ALEPH APPLIED CRYPTOGRAPHY

Length extension attack

- Recall Merkle-Damgård
- Given $h(m)$ and $len(m)$, an attacker is able to compute
  $$h(m \,||\, = \,||\, m' \,||\, = ')$$
- for any m' even without knowing the original message m – including any secret k in m.
- The attacker can craft a valid code for a message of their choosing by extending a valid code, without knowing the key.



Example: [Flickr] API

- Flickr allowed users to share data with third party apps without divulging the user's credentials. App providers ask the user to follow a link like this:

```
http://flickr.com/services/auth/?api_key=[api_key]&perms=[perms]&api_sig=[sig]
```

- All API calls using an authentication `token` must be signed:

```
args = {key=value}
api_sig = MD5(token || args)
request = args || api_sig
```

- With length extension, an attacker doesn't need to know the `token` to craft a valid request and signature.
- The original arguments and padding can further be eliminated thanks to a separate parsing vulnerability.

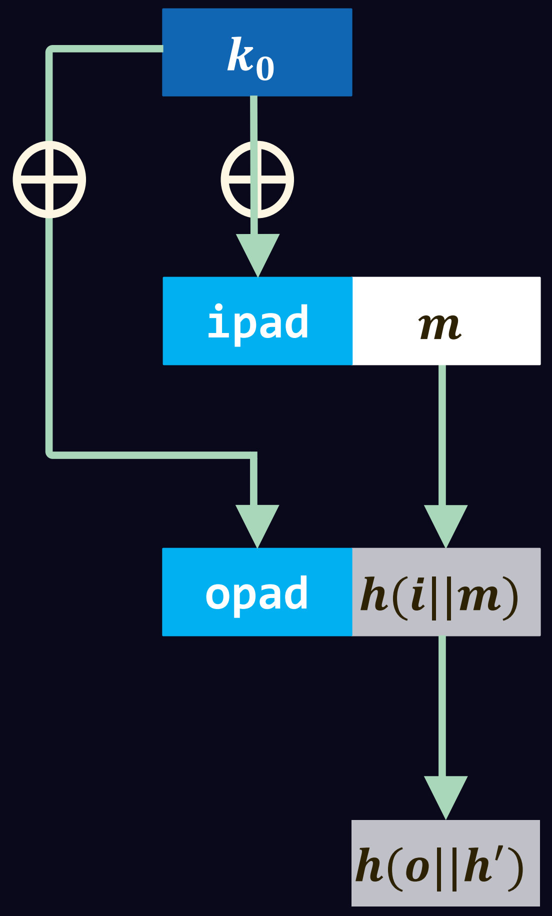
Cryptographic strength

	digest	Collision	Chosen Prefix	Preimage	Length extension
MD5	128	18	39	123.4	0
SHA-1	160	61.2	63.4	160	0
SHA-256	256	128		256	0
SHA3-256	256	128		256	512

https://en.wikipedia.org/wiki/Hash_function_security_summary

HMAC - Keyed-Hash Message Authentication Code

- Definition [**FIPS 198-1**]:
- $MAC = h(k \oplus o || h(k \oplus i || m))$
- h an approved hash function of input block size b
- k_0 the key k after any pre-processing to be of size b
- `opad: \x5c` repeated b times
- `ipad: \x36` repeated b times



Passwords

- Secrets must be stored in a way that prevents offline attacks (e.g. stolen db).
- Intuitively, if we store $h(p)$ it should not be possible to reconstruct p. It is, however, easy to precompute.
- If we store $h(s||p)$ with some random salt s for each p, we might force attackers to re-start each attempt.
- [**SP 800-63b**] recommends a 32-bit salt, 10000 iterations of a PBKDF (e.g. HMAC), and optionally one more iteration with a secret salt ("pepper") stored separately in an HSM. OWASP recommend a 32- to 64-bit salt.
- Beware common passwords – rankings are maintained, see wiki, github.

Linux passwords

- The linux function used to hash passwords is crypt(3). You can check the man page on your distribution, or see here.
- Salting algorithms are specified by an $id variable, and salts are encoded as strings of up to 16 characters, taken from a set of 64 possible values each, namely [a-zA-Z0-9./]. To quote:

ID	Algorithm
1	MD5
2a	Blowfish (not in mainline glibc; added in some Linux distributions)
5	SHA-256 (since glibc 2.7)
6	SHA-512 (since glibc 2.7)

Random number whitening

- Cryptographic hash functions are designed to give output computationally indistinguishable from random.
- They are an approved processing for random numbers. [**SP 800-90c**] A sequence is considered as having "full entropy" if the hashed sequence contains at least $2|d|$ bits of entropy (twice the digest size).
- There is a NIST suite designed to test random **bit** sequences [**SP 800-22**]. A sequence of SHA-256(b) will pass the test even if each b only contains just 8 bits of entropy.
- Low entropy messages are also an issue for message authentication. Suppose you only ever send two messages, e.g. "open" and "close". To avoid replay, you need a counter – which must be synchronized.

Proof of work – spam filter

[B02]

- Everyone is free to send me an email, but I will only read emails if the sender put some effort into them.

$$h(r\ e\ s\ @\ ) < d$$

- Each message to addressee @ has an attached statement:
  - *I am sender* s
  - *My email message to you is* e
  - *I have put in "*d*" amounts of work for the right to send; and*
  - *you can verify that I have done this by using the number* r *that I have found.*
- In other words, the PoW is an r solving the problem $h(r||m) < d$

References

- [**B02**] Back, A.: "Hashcash - A Denial of Service Counter-Measure". 2002. http://www.hashcash.org/papers/hashcash.pdf
- [**FIPS 198-1**] "Advanced Encryption Standard (AES)" doi: doi.org/10.6028/NIST.FIPS.198-1. url: csrc.nist.gov/publications/detail/fips/198/1/final
- [**Flickr**] "Flickr's API Signature Forgery Vulnerability" T. Duong, J. Rizzo, 2009. url: dl.packetstormsecurity.net/0909-advisories/flickr_api_signature_forgery.pdf
- [**hashclash**] "Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate". Stevens, Sotirov, Appelbaum, Lenstra, Molnar, Osvik, de Weger, Crypto 2009. url: www.win.tue.nl/hashclash/rogue-ca/. doi: 10.1007/978-3-642-03356-8_4

References

- [**LP20**] "SHA-1 is a Shambles: First Chosen-Prefix Collision on SHA-1 and Application to the PGP Web of Trust". Leurent, Peyrin 2020. www.usenix.org/conference/usenixsecurity20/presentation/leurent
- [**M79**] *"Secrecy, authentication, and public key systems."* Merkle, R. C. 1979. url: www.merkle.com/papers/Thesis1979.pdf
- [**RFC 1321**] "The MD5 Message-Digest Algorithm". url: https://tools.ietf.org/html/rfc1321
- [**SLdW07**] "Vulnerability of software integrity and code signing applications to chosen-prefix collisions for MD5". Marc Stevens, Arjen K. Lenstra, Benne de Weger. https://www.win.tue.nl/hashclash/SoftIntCodeSign/

References

- [**SP 800-22**] "A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications". url: csrc.nist.gov/projects/random-bit-generation/documentation-and-software
- [**SP 800-38b**] "Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication". doi: 10.6028/NIST.SP.800-38B. url: csrc.nist.gov/publications/detail/sp/800-38b/final
- [**SP 800-63b**] "Digital Identity Guidelines: Authentication and Lifecycle Management". doi: 10.6028/NIST.SP.800-63b. url: csrc.nist.gov/publications/detail/sp/800-63b/final
- [**SP 800-90c**] "Recommendation for Random Bit Generator (RBG) Constructions", second draft. url: csrc.nist.gov/publications/detail/sp/800-90c/draft
- [**WY05**] "How to Break MD5 and Other Hash Functions". Xiaoyun Wang, Hongbo Yu, Eurocrypt 2005. Doi: 10.1007/11426639_2