Defending Computer Networks
Lecture 20: More Encryption

Stuart Staniford

Adjunct Professor of Computer Science

Logistics

- HW5 out on website
  - Will need at least one correction
  - Due Weds Dec 2nd
- No lectures next week
- Quiz weighting options
  - A) Leave it as it is
  - B) Downweight quizzes
  - C) Have a third quiz
- Anonymous Piazza posts – Z request

Britain hit by massive cyber-attack as Islamic State hackers launch assault against Anonymous

12:57, 18 NOV 2015 | **UPDATED** 14:37, 18 NOV 2015 | **BY** JASPER HAMILL

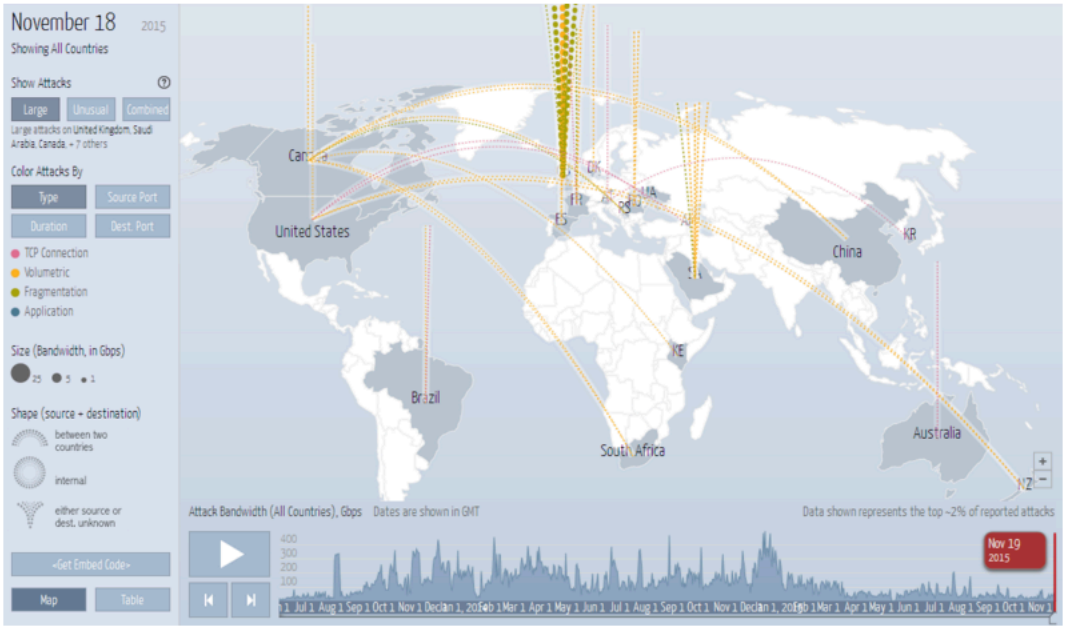
UK caught in crossfire of a global 'war' as terrorists lock horns with the masked hacking group

Britain has come under sustained cyber-attack as Islamist hackers launch revenge attacks against Anonymous, the shadowy group which has vowed to wipe "ISIS off the internet".

Unknown assailants launched their assault late last night and continued throughout the day.

Using a Twitter account set up to publicise its anti-ISIS operation, Anonymous said Britain had been "lit up" by the digital blitzkreig.

As of writing, a look at the Digital Attack Map shows an unprecedented amount of attack traffic aiming towards the UK. Most of the DDoS attacks use "fragmentation" which sends a flood of TCP or UDP fragments to a victim, overwhelming the victim's ability to re-assemble the streams and severely reducing performance.



Main Focus of Today

- AES (Advanced Encryption Standard)
- Public/Private Key Cryptography

Caesar Cipher

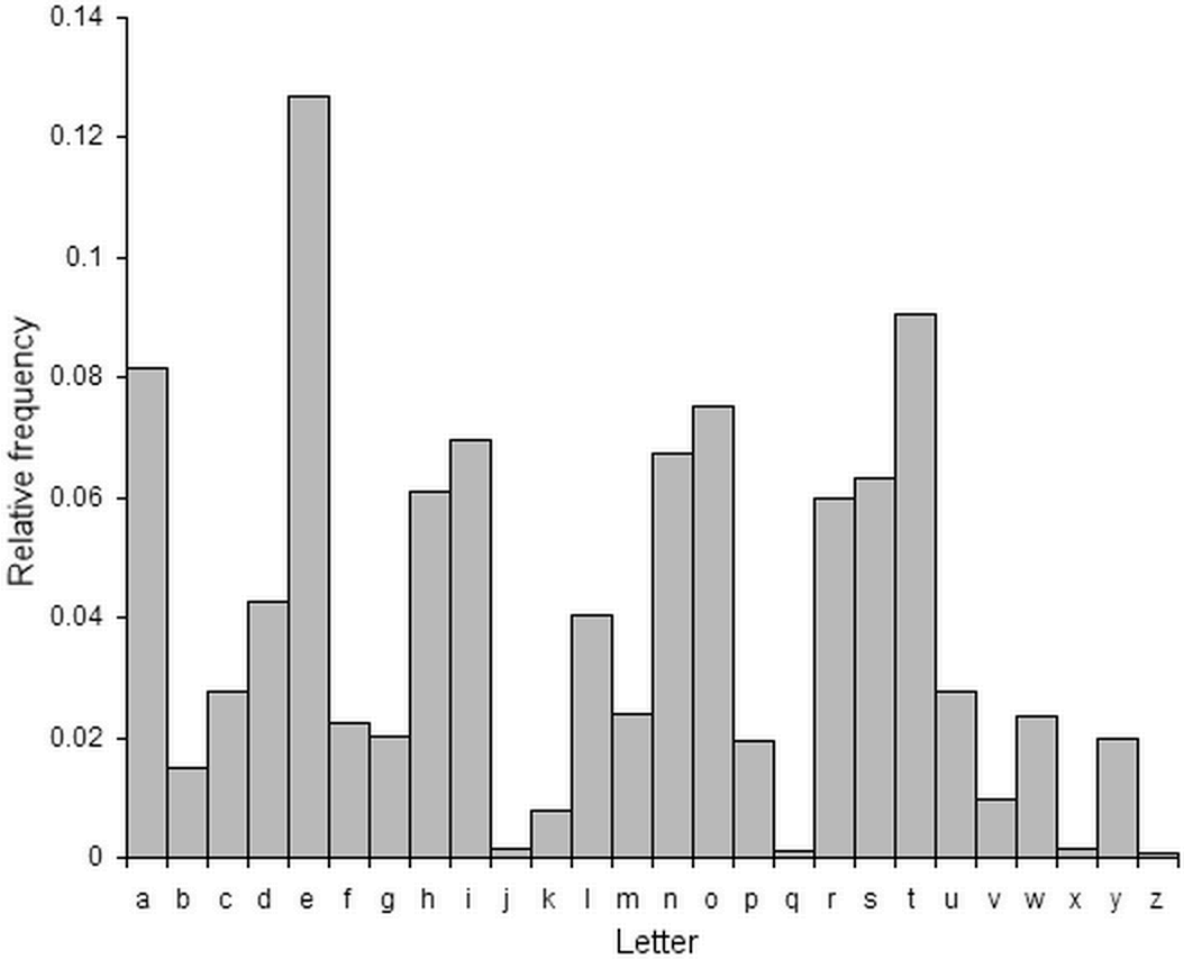
Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher: XYZABCDEFGHIJKLMNOPQRSTUVW

Plaintext: the quick brown fox jumps over the lazy dog
Ciphertext: QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

Here shift is key, -3 in this case

http://en.wikipedia.org/wiki/Caesar_cipher

Frequency Analysis



Example of a known-ciphertext analysis

http://en.wikipedia.org/wiki/Caesar_cipher

Entropy Example 3

- $H(X) = Sum[i..n, -P(x_i)*log(P(x_i)]$
- Suppose X is one byte and
  - Only two bytes over occurs, say ‘a’ and ‘b’
  - P(‘a’) = 1/2, P(‘b’) = 1/2 else $P(x_i) = 0$
  - log P(‘a’) = log P(‘b’) = -1,
  - H(X) = 1
  - One bit of information in each byte
    - Either ‘a’ or ‘b’ – code with 0 or 1.

Entropy of English

- Prior examples assume successive picks are independent
- Not true of natural languages, eg “qu”
- Have to account for these dependencies by making the state vector larger
- English has about 1 bit per character
- Highly relevant for cryptanalysis

Eg Brute Force Password Guessing

- Say over the network
- Say password is 8 ascii (7 bit) characters
  - How many possibilities to guess if people pick randomly
  - How many if people pick English words/phrases?

One Time Pad

- Like Caesar Cipher
  - Modulo addition on characters, but
  - Instead of a single constant shift
  - Key is random, different on each character
  - Requires a key that is as long as the plaintext
  - The “pad” – keystream – is shared in advance
  - Pad has full entropy of the alphabet

One Time Pad Example

Supposing we number letters 0-25, and make space 26

Key:	3	15	22	11	19	1	8	25	4	22	7	13	26	12	9	3
Plaintext:	T	H	E		Q	U	I	C	K		B	R	O	W	N	
Ciphertext:	W	W		K	I	V	Q	A	O	V	I	D	N	H	W	C

What about frequency analysis now?

One Time Pad Properties

- Shown by Shannon (1949) that
  - **If** the key is genuinely random/completely unpredictable
  - Then ciphertext contains no information about the plain text.
- Practical difficulties
  - Distributing full length one time pad
    - If you reuse pad, after a while, cryptanalysis possible
  - Alternatively, use an algorithmic RNG
    - Potentially analyzable if algorithm can be discovered

AES

- AES = Advanced Encryption Standard
- Published by US NIST (2001)
  - Following an open competition
  - Including public crypto-analysis attempts
- Replaced earlier (1977) DES standard
- Widely used in practice
  - Fairly fast
- Currently believed secure
  - Some slight uncertainty following Snowden
    - https://www.schneier.com/blog/archives/2012/03/can_the_nsa_bre.html

AES Design

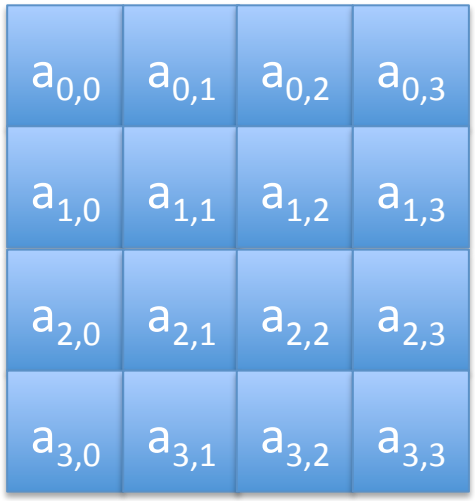
- Block oriented cipher
  - Takes blocks of plaintext 128 bits at a time
    - 16 bytes
  - Other sizes must be padded
- Key sizes of 128, 192, and 256 bits
- What's known as a substitution-permutation network
- Repeated numerous rounds
- Will sketch how it works while skipping worst of the math

AES Rounds

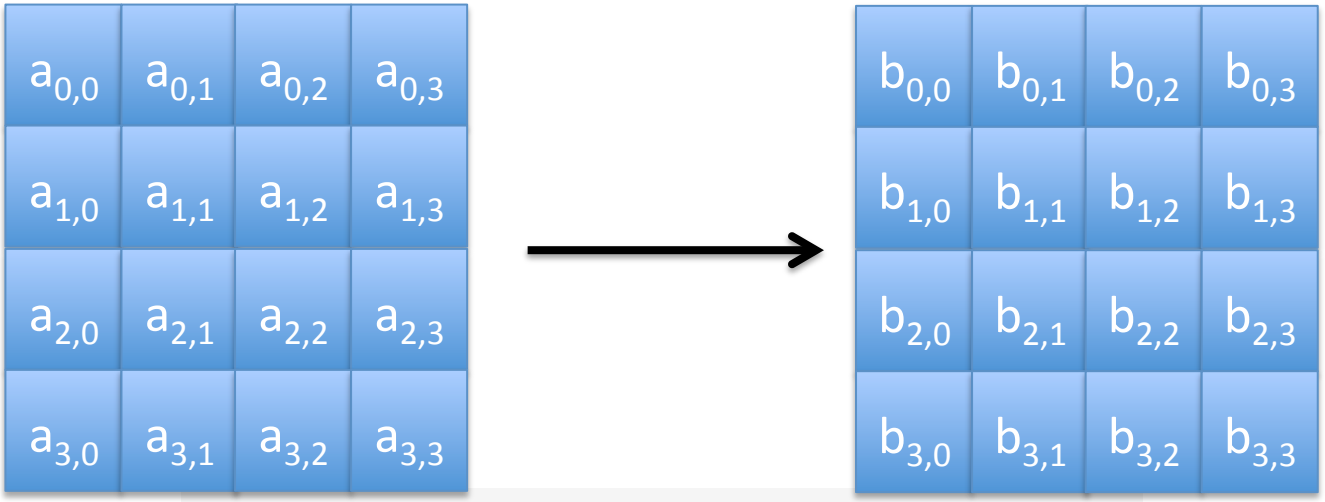
- The basic cipher operations are performed repeatedly on each block
  - 128 bit keys, 10 rounds
  - 192 bit keys, 12 rounds
  - 256 bit keys, 14 rounds
- Each round uses a different key
  - Derived from the master key using a set of complex algebraic operations
  - Depending on the algebra of finite (Galois) fields.

AES State Matrix

- Each 16 byte block is arranged in a 4x4 state matrix
- Used to keep track of what block will turn into



First Step: S-box (substitution)



	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

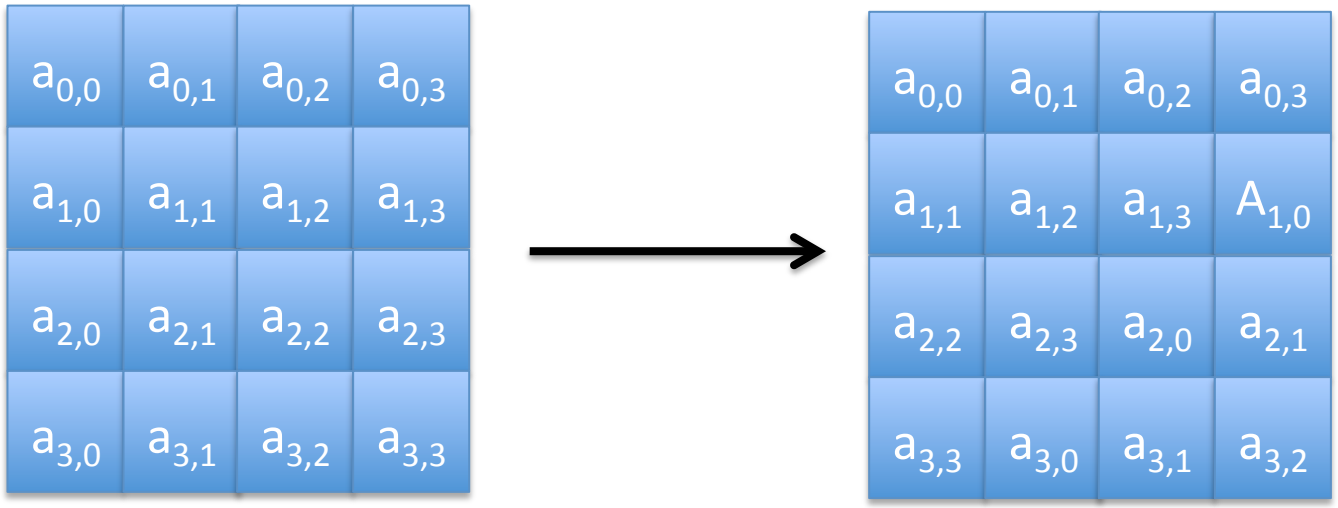
Shift Rows Step

No change

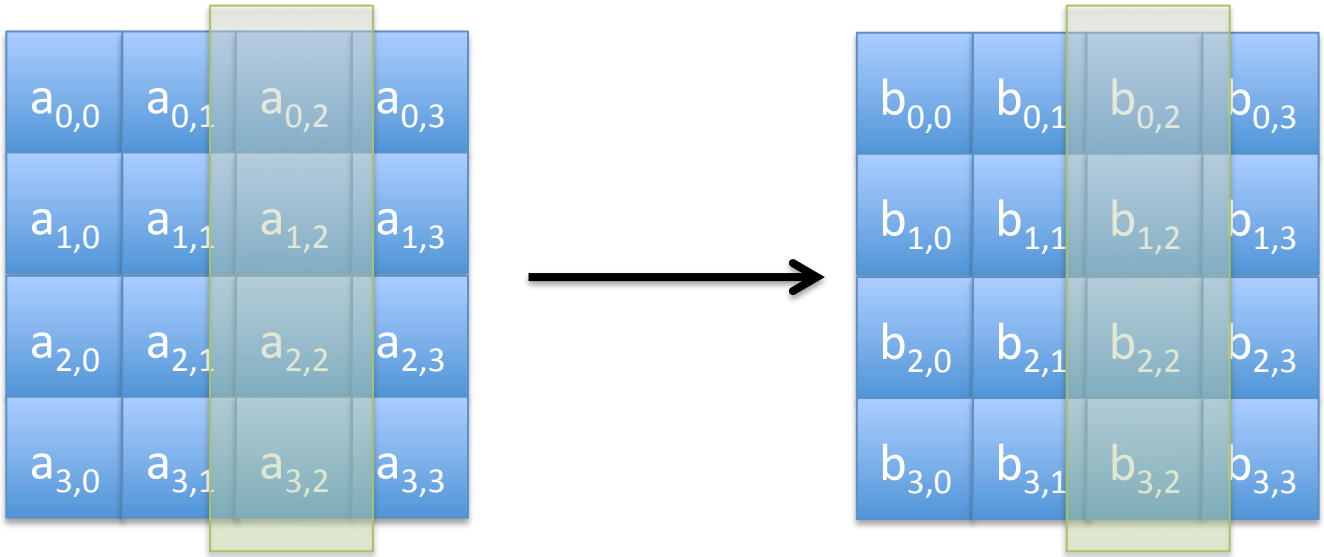
One left

Two left

Three left



Mix Columns Step



Each column is transformed by a complex (but fixed) algebraic equation that takes input from all four bytes and incorporates them into each output byte

Mix Columns Detail

Multiply block by this matrix:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

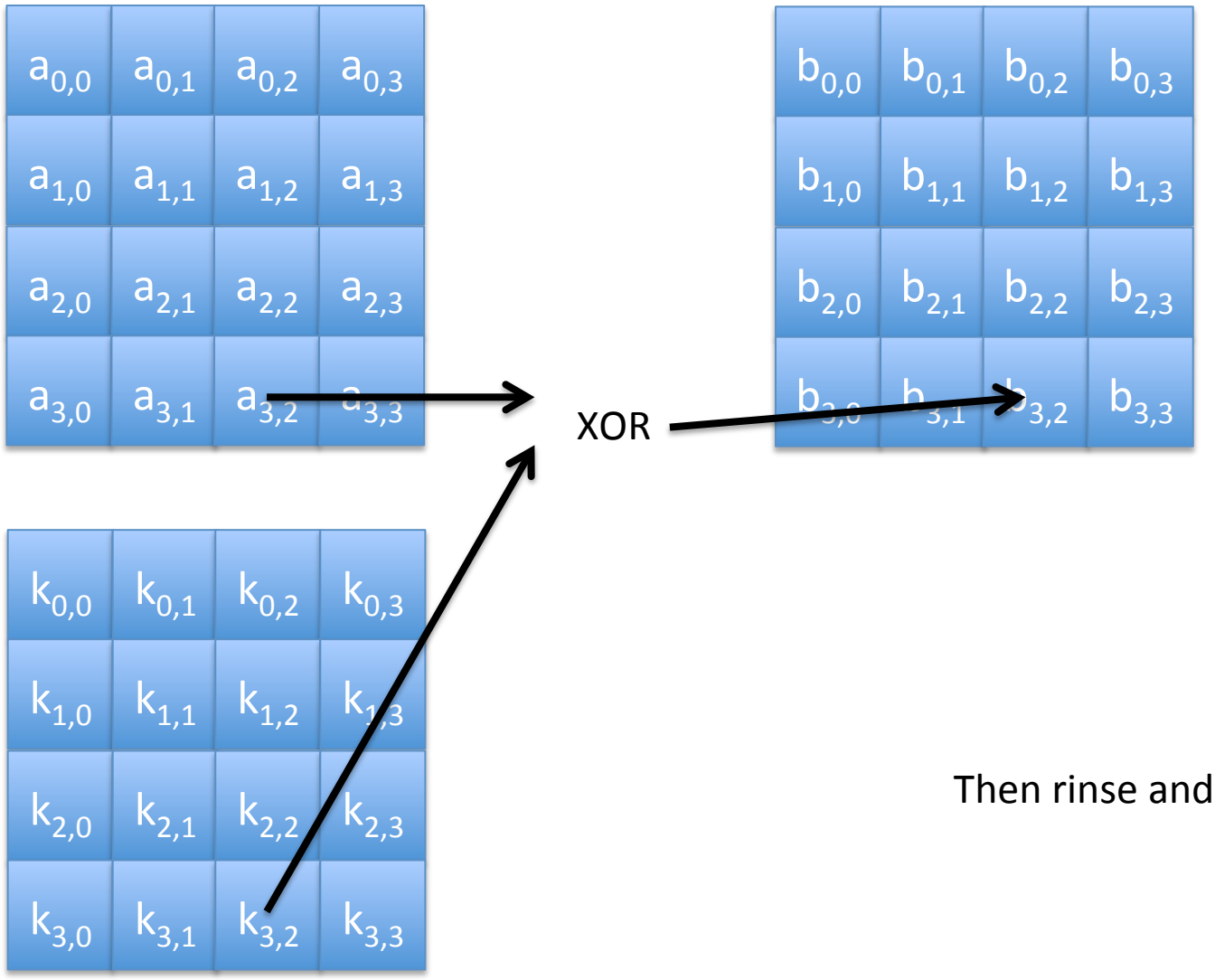
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

where

multiplication by 1 means no change, multiplication by 2 means shifting to the left, and multiplication by 3 means shifting to the left and then performing XOR with the initial unshifted value. After shifting, a conditional XOR with 0x1B should be performed if the shifted value is larger than 0xFF. Addition is simply XOR.

http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Add Round Key



Then rinse and repeat…

Public Key Cryptography

- Also known as Asymmetric Cryptography
  - To distinguish from symmetric cryptography
    - shared secret
- Instead of shared key, keys come in pairs
  - Public key used to encrypt data
  - Private key used to decrypt data
  - Not feasible to infer private key from public key

Main Advantage

- Enormously simplifies key management
  - Imagine large group that need to communicate by shared key schemes
    - Share one key amongst many and risk losing everything
    - Or keep track of n^2 keys
- With public key crypto, I give out my public key, everyone can know it.
  - Anyone can send me a message
  - Only I can read those messages
  - I only have to worry about one secret key (mine)
  - Don't have to share my secret (private) key with anyone

History

- Invented in the seventies
- 1973, secretly, by GCHQ (British intel)
- 1976 Diffie Hellmann Key exchange
- 1977 RSA public/private key exchange
  - Rivest, Shamir, Adleman
- Many more schemes since then
- RSA still in wide practical use
  - 1024 bit keys considered weak, but longer ok

RSA

- Underlying base is difficulty of factoring very large numbers
- Will sketch algorithm while again skipping worst of math details
- We choose two large primes *p*, *q*
  - hundreds of digits each
  - Modulus, $n = pq$
  - Size of *n* in bits is the key length
  - Then choose an exponent, *e* that
    - Has no common factors with $(p-1)(q-1)$
- Public key is *n* and *e*
- Private key can be computed from *p* & *q*

Encryption

- Take the text and turn blocks of text into numbers. Say M is number for some block
- Then take M^e mod n
- Toy example
  - http://www.woodmann.com/crackz/Tutorials/Rsa.htm
  - p = 43, q = 59 (both prime)
  - n = 43*59 = 2537
  - e = 13 (no common factor with 42 or 58)

Encryption Example

- Message is ST OP
- Encode as 1819 1415
  - $1819^{13} \bmod 2537 = 2081$
  - $1415^{13} \bmod 2537 = 2182$
  - Ciphertext is 2081 2182

Decryption

- Find decryption exponent d such that
- $de \bmod (p-1)(q-1) = 1$
- Ie *d* is the multiplicative inverse of *e*,
  - modulo $(p-1)(q-1)$
  - Tractable algorithms for this are known
- Then plaintext P can be computed from C
  - $P = C^d \bmod n$

Decryption Example

- Suppose ciphertext is 0981 0461
- d = 937 for our previous example
- 937*13 mod 2436 = 12181 mod 2436 = 1
- 0981^{937} mod 2537 = 0704
- 0461^{937} mod 2537 = 1115
- Message was HE LP

Applicability of Public Key

- Typically public key algorithms are computationally expensive.
- Not practical to apply to long messages
- Therefore generally used in the process of establishing a temporary symmetric key
  - Session key
    - Encrypted by public key crypto for transfer
    - Then used to encrypt lengthy communication session
    - Then thrown away

Let’s Try It

- openssl genrsa -out private_key.pem 1024
- more private_key.pem
- openssl rsa -in private_key.pem -text -noout
- openssl rsa -pubout -in private_key.pem -out public_key.pem
- more public_key.pem