SY110
Digital Cryptography Tools and Applications

Major Brian Hawkins, USMC

U.S. Naval Academy

Fall AY 2018

Outline

Admin

Reminder: Mandatory lecture in Alumni @ 1200 on Tuesday.

Review

Review

- Symmetric Encryption
- Assymetric Encryption
- Hashing

`http://rona.academy.usna.edu/~sy110/lec/cryptDig/cryptosummary.html`

(Re)Hashing Tools - md5

- Download file name "foo" from
  `http://faculty.cs.usna.edu/~bhawkins/courses/sy110/calendar.php?type=class&event=26`

From the command line:

- md5 foo
- md5 -d"let it be"
- md5 -d"let it ba"

Suppose you have a password file that looks like this:

username	salt	md5hash(salt+password)
...		
bjones	k%W3?A1	c8d0c0fec386b9cd6a625b3c8e57c988
...		

Which is the correct password: StartMeUp81 -or- LetItBl33d

Modern Day Symmetric Encryption - AES

Why could we break Caesar & Vigenre cipher?

- Frequency Analysis works because the possible value space is very small: a single english character only has 26 possible values.
- Digitally speaking: a single byte only has 256 possible, also very small in the scheme of things.
- Vigenre cipher is breakable as soon as we know the length of the keyword (except for the OTP)

How can we fix it?

- Answer: increase the possible value space to eliminate the possibility of frequency analysis
- Advanced Encryption Standard (AES):
  Encrypt 16-byte (128-bit) blocks at a time.
- $2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456$

Modern Day Symmetric Encryption - AES

Why could we break Caesar & Vigenre cipher?

- Frequency Analysis works because the possible value space is very small: a single english character only has 26 possible values.
- Digitally speaking: a single byte only has 256 possible, also very small in the scheme of things.
- Vigenre cipher is breakable as soon as we know the length of the keyword (except for the OTP)

How can we fix it?

- Answer: increase the possible value space to eliminate the possibility of frequency analysis
- Advanced Encryption Standard (AES):
  Encrypt 16-byte (128-bit) blocks at a time.
- $2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456$

Modern Day Symmetric Encryption - AES

Secuity through complexity

- AES is crackable through a brute force attack, but with ...
- 340,282,366,920,938,463,463,374,607,431,768,211,456 possible keys ...
- we will be long dead before decrypting this has any value.

From the command line:

- `aes -k` returns a *key*
- `aes -e` [*key*] [*PT*] returns *CT*
- `aes -d` [*key*] [*CT*] returns *PT*
- `aes -h` for more information

Let our powers combine!

Hashing as keygen for AES

- You can't remember a 128-bit encryption key for AES, but you can remember a passphrase...
- so use md5 on your passphrase to generate the AES key, then use AES to encrypt/decrypt.

Modern Day Asymmetric Encryption - RSA

From the command line:

- `openssl genrsa -out keypair.pem 2048`
  Generates a 2048-bit RSA keypair and store in file keypair.pem
- `openssl rsa -text -in keypair.pem`
  View the RSA keypair in file keypair.pem
- `openssl rsa -in keypair.pem -pubout -out pubkey.pem`
  Extract the public key from keypair.pem and save in file pubkey.pem
- `openssl rsautl -encrypt -pubin -inkey pubkey.pem -in plain -out cipher`
  Encrypt the file plain using the public key in file pubkey.pem, store the result in file cipher
- `openssl rsautl -decrypt -inkey keypair.pem -in cipher -out plain1`
  Decrypt the file cipher using the private key in file keypair.pem, store the result in file plain1

Let our powers combine...again!

Hashing + RSA for Digital Signatures

- RSA is SLOW!!!
- So go ahead and send the whole file as is.
  (Anybody could decrypt it anway with your public key)
- But hash the file and encrypt the hash with your private key...
  this becomes the Digital Signature.
- The recipient can hash the file also and compare it to the result of decrypting the Digital Signature with your public key
- If they match, then the original file was sent by you.

Activity

- Right-click and save to Desktop:
  `http://rona.academy.usna.edu/~sy110/lec/cryptDig/URLA.txt`
  `http://rona.academy.usna.edu/~sy110/lec/cryptDig/URLB.txt`
  `http://rona.academy.usna.edu/~sy110/lec/cryptDig/URL.sig`
- Go to the sy110 homepage and save to your desktop pubkey.pem, the official sy110 public key.

Which of these really came from the sy110 Course Coordinator?

Open a command shell, cd to your Desktop directory, and try the following commands:

- `openssl dgst -sha1 -verify pubkey.pem -signature URL.sig URLA.txt`
- `openssl dgst -sha1 -verify pubkey.pem -signature URL.sig URLB.txt`

So which is officially sanctioned?

Questions

Questions?