CSE 484 / CSE M 584
Computer Security: Final Exam Review

TA: Adrian Sham

adrsham@cs

Using material of Franzi's slides

Reminders

- Lab #3 due tomorrow (6/5), 5pm
- Office hour tomorrow! 9:30am CSE 003D
- Final exam is 8:30-10:20am (6/9) in MGH 241
- Please do the Course Evaluations (section)!
  - AA: https://uw.iasystem.org/survey/146319
  - AB: https://uw.iasystem.org/survey/146316

Buffer Overflows

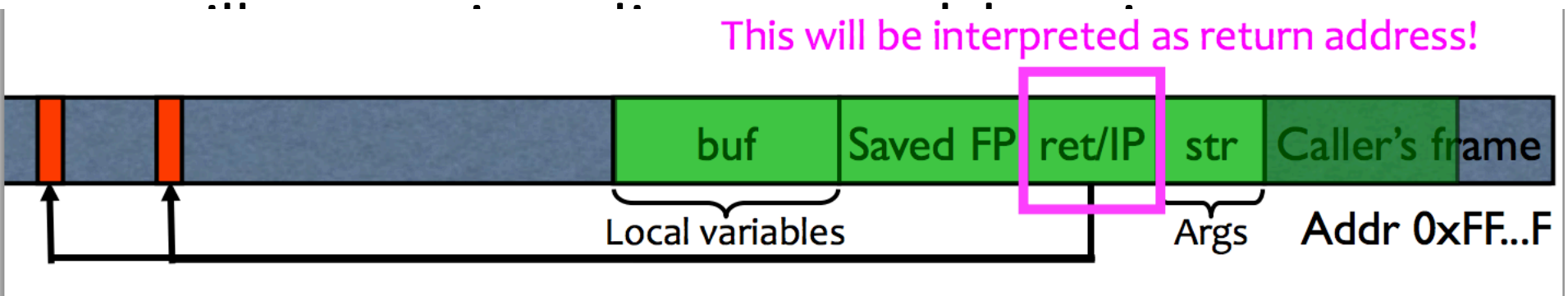
- What is a buffer overflow?
  - Occurs when a program writes data beyond the boundary of the buffer
- Main cause of problem
  - No/bad bounds checking
- Unsafe C library functions
  - strcpy(char *dest, const char *src)
  - strcat(char *dest, const char *src)
  - gets(char *s)
  - scanf(const char *format, …)
  - printf(const char *format, …)

Basic Buffer Overflows

- Memory pointed to by str is copied onto stack

```
void func (char *str) {
    char buf[126];
    strcpy (buf,str);
}
```

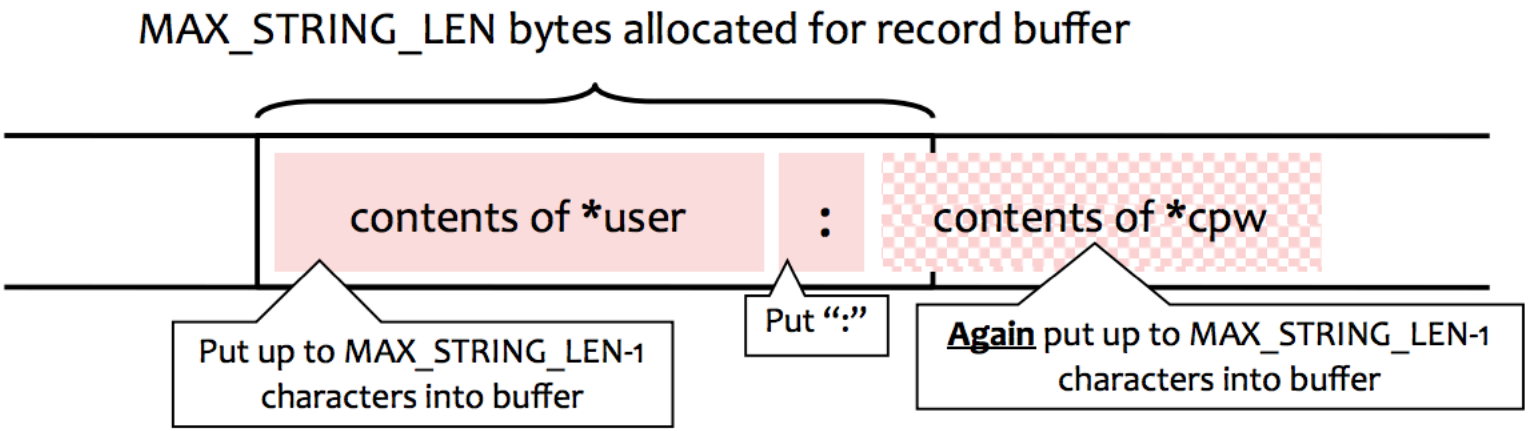
- If a string longer than 126 bytes is copied, it



Bounds checking

- Make sure that the **right** value is being supplied
- strncpy (char *dest, const char *src, size_t n)

```
strncpy(record,user,MAX_STRING_LEN-1);
strcat(record,":")
strncat(record,cpw,MAX_STRING_LEN-1);
```



Off-By-One Overflow

```
void mycopy(char *input) {
     char buffer[512]; int i;

     for (i=0; i<=512; i++)
         buffer[i] = input[i];
}
void main(int argc, char *argv[]) {
     if (argc==2)
        mycopy(argv[1]);
}
```

This will copy **513** characters into buffer. Oops!

Writing Stack with Format Strings

- %n format symbol tells printf to write the number of characters that have been printed
  - printf (“Overflow this! %n”, &myVar);
  - Argument of printf is interpreted as destination address
  - This writes 14 into myVar
- This includes other related functions, including sprintf, fprintf etc.

Writing Stack with Format Strings

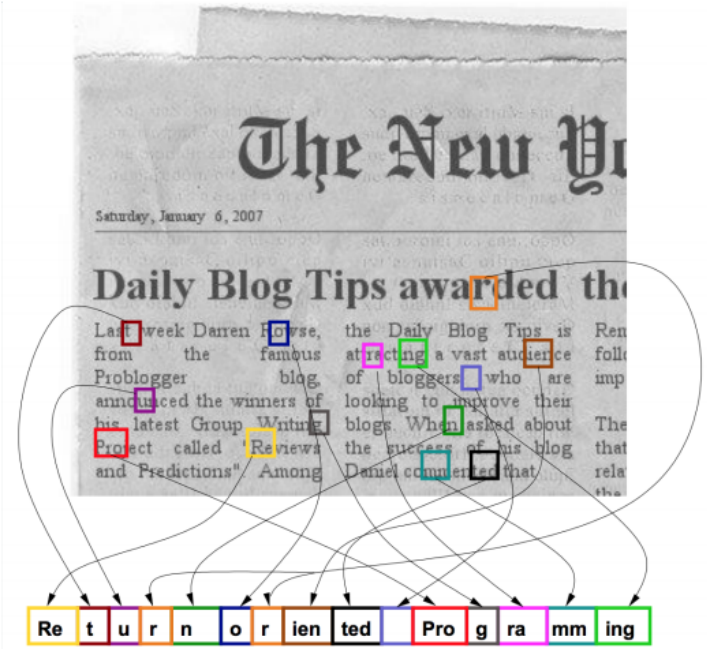
- What if printf does **not** have an argument?

```
char buf[16]="Overflow this!%n";
printf(buf);
```

- Stack location pointed to by printf's internal stack pointer will be interpreted as address into which the number of characters will be written.
- What if attacker controls `buf`?

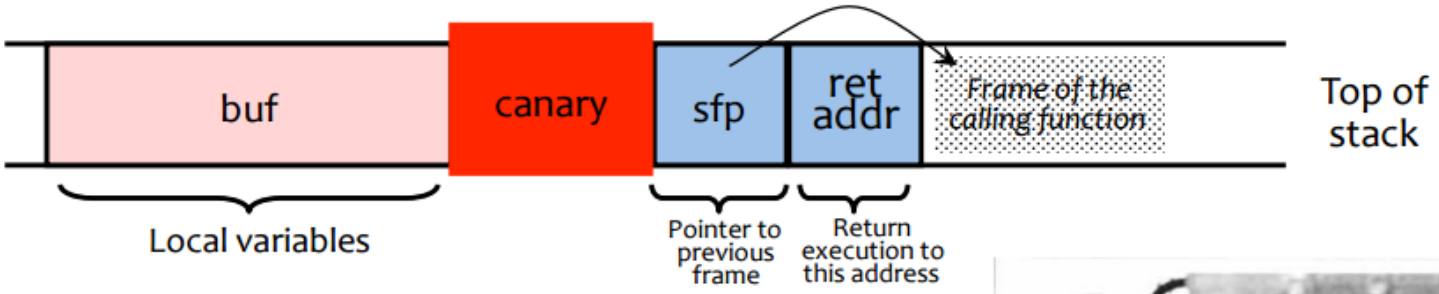
Defenses

- Mark all writeable memory locations as non-executable
  - Does not prevent return-to-libc exploits
  - Does not prevent return-oriented programming



Defenses

- Stack canaries
  - Embed "canaries" (stack cookies) in stack frames and verify their integrity prior to function return
  - Any overflow of local variables will damage the



Defenses

- ASLR: Address Space Randomization
  - Map shared libraries to a random location in process memory
    - Attacker does not know addresses of executable code
- Issues
  - NOP slides and heap spraying to increase likelihood for custom code execution
  - Brute force or memory disclosures to map out memory on the fly

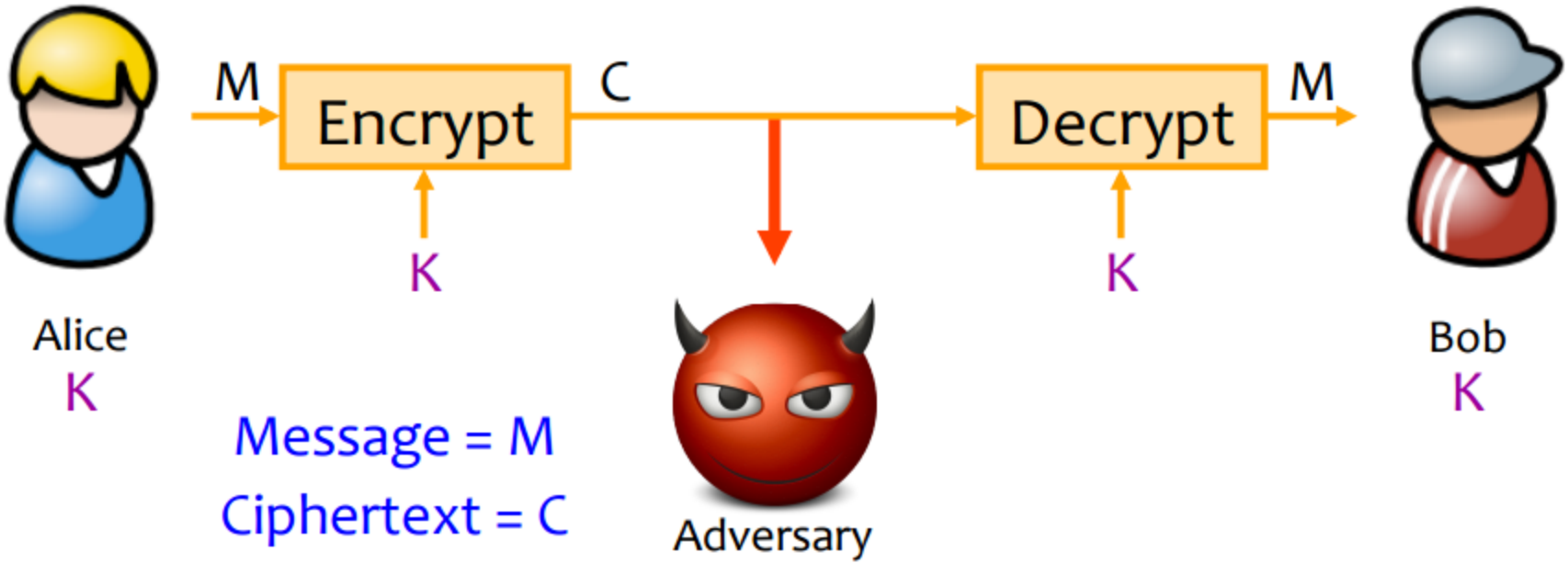
Fuzz Testing (Fuzzing)

- Generate “random” inputs to a program
  - Sometimes conforming to input structures (XML file structure etc.)
- Try lots of different inputs, and see if program crashes
  - If crashes, a bug was found
  - And the bug may be exploitable
- Sanitize your inputs

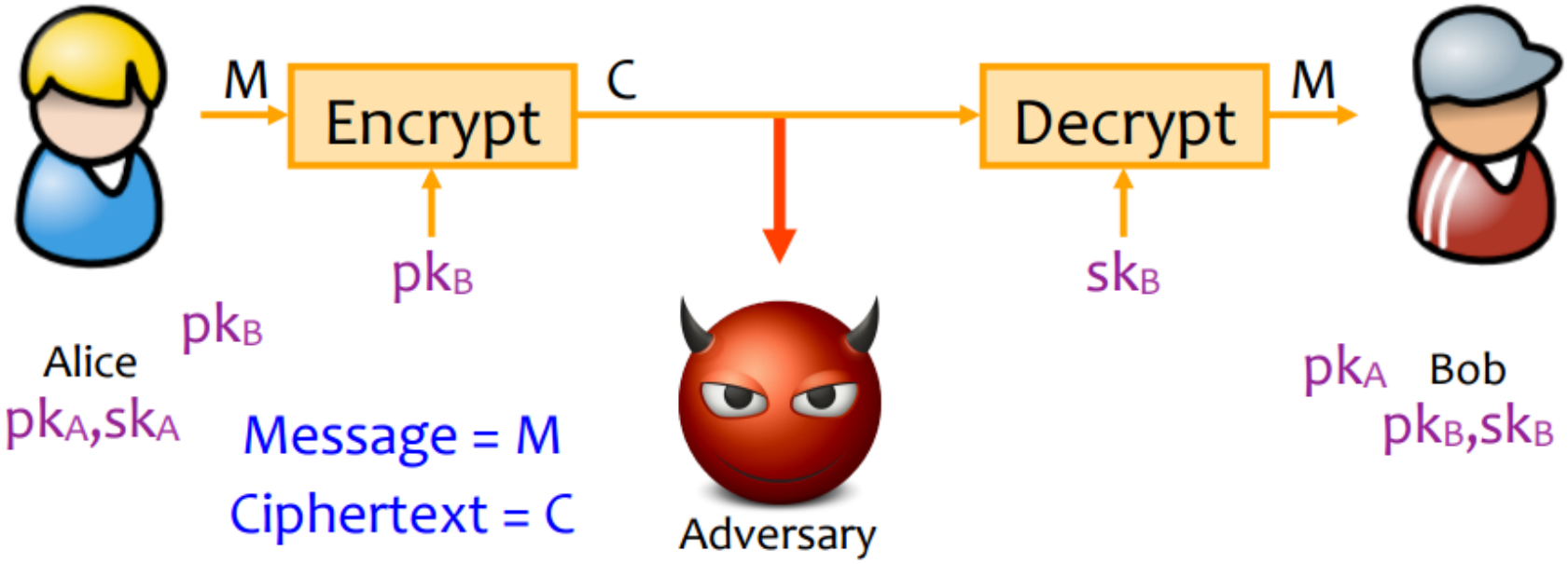
Cryptography

- 2 Flavors
  - Symmetric crypto
    - Both communicating parties have access to a shared random string K, called the key.
  - Asymmetric crypto
    - Each party creates a public key **pk** and a secret key **sk**
- Pros and cons?

Achieving Privacy (Symmetric)



Achieving Privacy (Asymmetric)



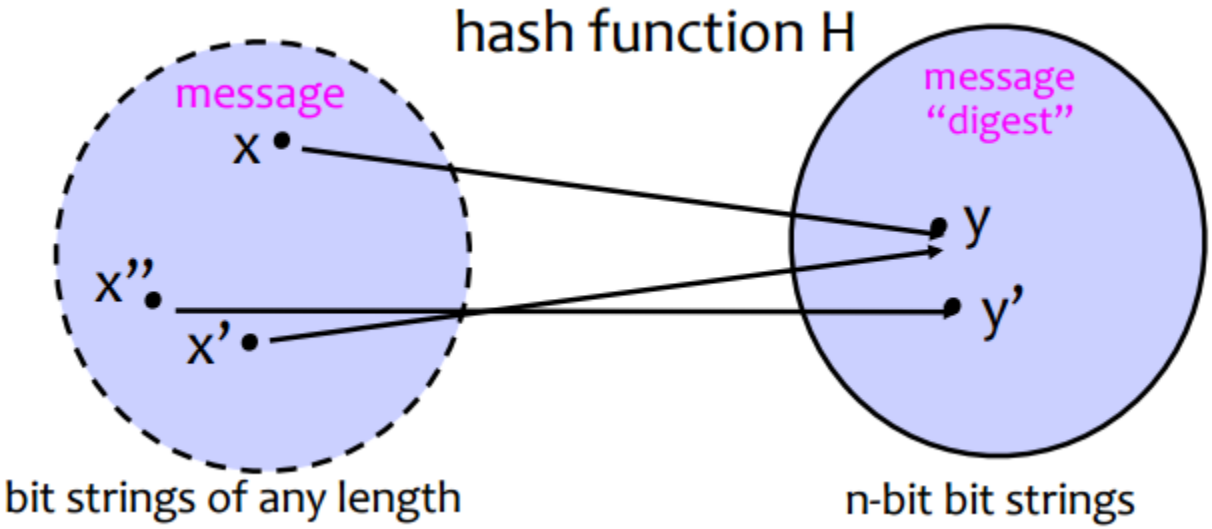
Encrypting a large message

- Block cipher (AES, DES) cannot encrypt stuff larger than 128-bit
- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Counter Mode (CTR)
- Remember the pros and cons of each mode, weaknesses?

How can a cipher be attacked?

- Attackers know ciphertext and encryption algorithm
  - What else does the attacker know?
- Ciphertext-only attack
- KPA: Known-plaintext attack
  - Knows some plaintext-ciphertext pairs
- CPA: Chosen-plaintext attack
  - Can obtain ciphertext for any plaintext of his choice
- CCA: Chosen-ciphertext attack
  - Can decrypt any ciphertext except the target

Hash Functions

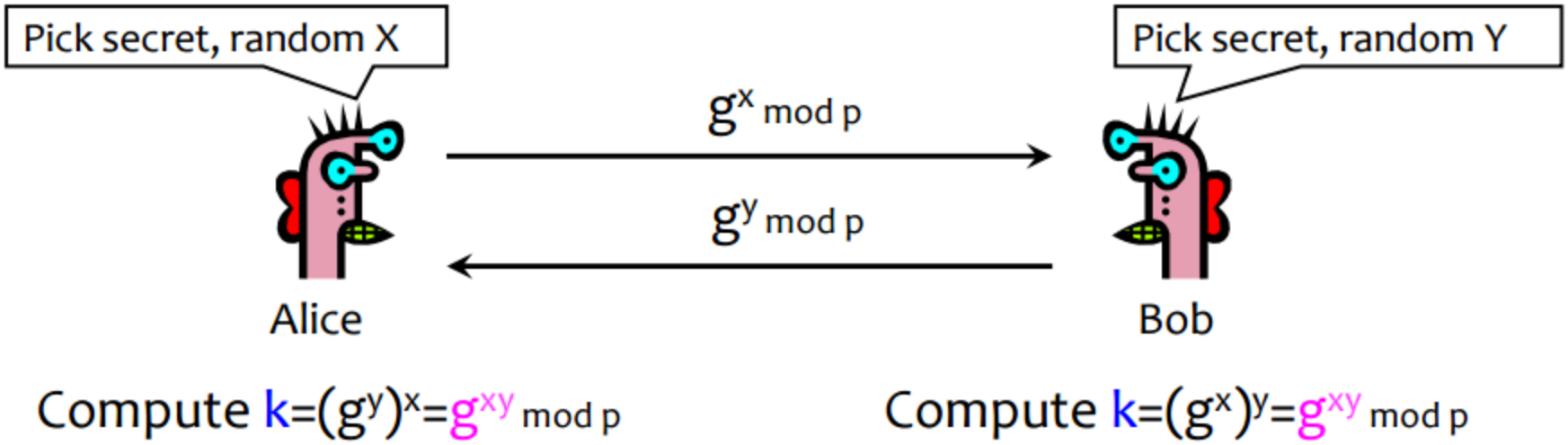


Cryptographic hash function

- What properties do we want from cryptographic hash functions?
  - One-way
  - Collision Resistance
  - Weak Collision Resistance
- Uses of hash functions?
  - Hashing passwords
    - Why?
- Common Hash functions
  - MD5, SHA-1, SHA-256 etc.

Exchanging keys

- Diffie-Hellman Protocol
  - A method for securely exchanging cryptographic keys over a public channel
  - Public info: p and g



RSA Cryptosystem

- Key generation:
  - Generate large primes p, q
    - Say, 1024 bits each (need primality testing, too)
  - Compute $n=pq$ and $\varphi(n)=(p-1)(q-1)$
  - Choose small e, relatively prime to $\varphi(n)$
    - Typically, $e=3$ (can be vulnerable) or $e=2^{16}+1=65537$
  - Compute unique d such that $ed = 1 \bmod \varphi(n)$
    - Modular inverse: $d = e^{-1} \bmod \varphi(n)$
  - Public key = (e,n); private key = (d,n)
- Encryption of m: $c = m^e \bmod n$
- Decryption of c: $c^d \bmod n = (m^e)^d \bmod n = m$

Key Distribution

- How to distribute public keys while preventing forgery and tampering?
  - Public-key certificate
    - Signed statement specifying the key and identity
- Common approach: certificate authority (CA)
- How to revoke a bad certificate?
  - Certificate revocation lists (CRL)
  - Issues with CRL?
- Convergence
  - Observe unexpected changes from existing certificates

Crypto summary

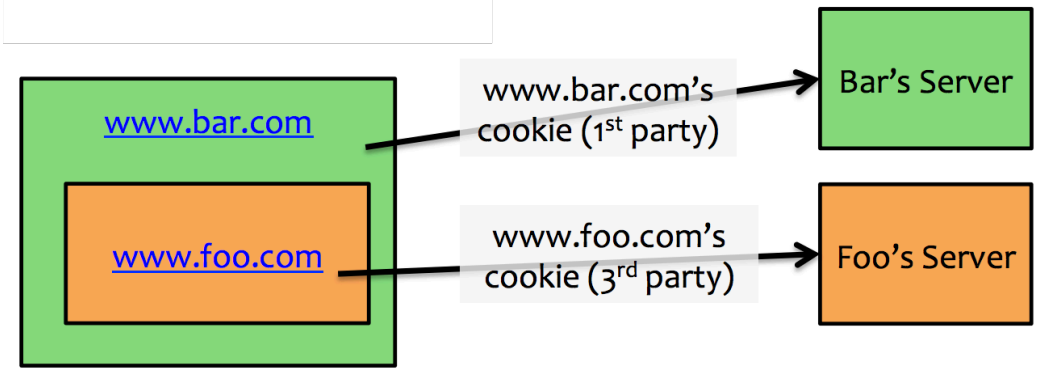
- Goal: Privacy
  - Symmetric keys
    - Onetime pad,
    - Block ciphers (DES, AES) -> modes: EBC,CBC,CTR
  - Public key crypto (Diffie-Hellman, RSA)
- Goal: Integrity
  - MACs, often using hash functions (e.g. MD5, SHA-256)
- Goal: Privacy and Integrity
  - Encrypt-then-MAC
- Goal: Authenticity (and Integrity)
  - Digital signatures (e.g. RSA)
- Kerckhoff's Principle
  - Security of a cryptographic object should depend only on the secrecy of the secret key

Web security

- Web browser
- Web applications
- Same origin policy
  - Can only access properties of documents and windows from the same domain, protocol, and port
  - Applies to cookies also
    - Only code from the same origin can read/write cookies associated with an origin

Cookies

- Browsers automatically include cookies with HTTP requests
- First-party cookie: belongs to the top-level domain
- Third-party cookie: Belongs to domain of embedded content



Other topics on web security

- Cross-Site Request Forgery (CSRF/XSRF)
- Cross-Site Scripting
  - Reflected XSS
  - Stored XSS
- Preventing XSS
  - Any user input and client-side data must be preprocessed before it is used inside HTML
  - Remove / encode HTML special characters
- Evading XSS Filters
- SQL Injections
- Third Party Tracking
  - How do third parties track your browsing?
  - Defenses?

Authentication and Passwords

- Password security
  - How to store passwords?
- Multi-Factor Authentication
  - How do they work?
- Different types of authentication
  - Graphical passwords
  - Biometrics

Mobile Platform security

- Differences from traditional OSes
- Android security
  - Based on Linux
  - Application sandboxes
- IOS security
- Differences between Android and iOS security model?
- Differences between new and old Android security model?

Other topics

- Usable Security
  - Phishing
  - SSL warnings
  - Password managers
- Anonymity on public networks
  - Onion Routing
    - Tor

Thanks for a great quarter ☺

- Hope you learned a lot about security
- Remember to do course evaluations!
- See you tomorrow and during final exams