An Introduction to Cryptography
CMSC 23200/33250, Winter 2020, Lecture 3

David Cash & Blase Ur
University of Chicago
Your connection to this site is private.

Details

Permissions
Connection

Chrome verified that Symantec Class 3 Secure Server CA - G4 issued this website's certificate. The server did not supply any Certificate Transparency information.

Certificate Information

Your connection to www.amazon.com is encrypted using a modern cipher suite.

The connection uses TLS 1.2.

The connection is encrypted and authenticated using AES_128_GCM and uses ECDHE_RSA as the key exchange mechanism.

What do these mean?
The Wi-Fi network "Pat's wifi" requires a WPA2 password.

Password: [Enter password]

- [ ] Show password
- [x] Remember this network

[ Cancel ] [ Join ]
Can you please come over asap to help me move the couch?
I need to be out of here by 3pm
I guess you forgot your phone at home or something
What is Cryptography?

Cryptography involves algorithms with security goals.

Cryptography involves using math to stop adversaries.
Common Security Goal: Secure Channel

Confidentiality: Adversary does not learn anything about messages $m_1, m_2$

Authenticity: $m'_1 = m_1$ and $m'_2 = m_2$
WPA2 (Wi-Fi Protected Access 2): Secure WiFi

pw="fourwordsuppercase"

Secure channel

Physical medium (air)
GSM Cell Phone Encryption (A5/1, A5/3)

Secure channel

Physical medium (air)

K = b9842544

<table>
<thead>
<tr>
<th>User</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice Doe</td>
<td>340934c3</td>
</tr>
<tr>
<td>Betty Lee</td>
<td>b9842544</td>
</tr>
<tr>
<td>Cheryl Zang</td>
<td>93d94520</td>
</tr>
<tr>
<td>Pat Dobbs</td>
<td>2ea0f48d</td>
</tr>
</tbody>
</table>
Disk Encryption

K = b9842544

macOS

Hard Drive
Crypto in your browser: TLS (Transport Layer Security)

No pre-shared key, yet “guarantees” secret & authenticated communication with amazon.com.
Attacks on TLS

- **Crypto primitives**
  - RSA, DSA, ECDSA
  - Diffie–Hellman, ECDH
  - HMAC
  - MD5, SHA1, SHA-2
  - DES, 3DES, RC4, AES
  - Export grade

- **Data structures**
- **Encryption**
  - Modes, IVs
- **Padding**

- **Alerts & errors**
- **Certification / revocation**
- **Negotiation**
- **Renegotiation**
- **Session resumption**
- **Key reuse**
- **Compression**
- **State machine**

- **Library details**
  - OpenSSL
  - LibreSSL, BoringSSL
  - NSS
  - Gnu
  - SC
  - s2n

- **Applications**
  - Web browsers: Chrome, Firefox, IE/Edge, Safari
  - Web servers: Apache, IIS, nginx, node, ...

- **Certificates**
  - Protocols
    - HTTP, IMAP, ...

- **Protocols**
  - SSL 2.0
  - SSL 3.0
  - TLS
  - SSL 2.0 downgrade
  - FREAK, Logjam
  - POODLE
  - BEAST
  - Cross-protocol DH/ECDH attack
  - SSL 2.0 downgrade, FREAK, Logjam
  - Termination, Cookie Cutter
  - Heartbleed
  - Debian OpenSSL entropy bug
  - SLOTH

- **Attacks**
  - SLOTH
  - POODLE
  - BEAST
  - Lucky13
  - Cross-protocol DH/ECDH attack
  - SSL 2.0 downgrade, FREAK, Logjam
  - Lucky microseconds
  - Ray & Dispensa
  - MaloDroid
  - CCS injection
  - SMACK
  - BERserk
  - Jager et al.
  - DROWN
  - SLOTH
  - Bleschenbacher
  - CRIME, BREACH, HEIST

- **Virtual host confusion**
  - STARTTLS injection
  - SSL stripping
  - CA breaches
  - Frankencerts
  - MalloDroid
  - CCS injection
  - Lucky microseconds
  - Jager et al., DROWN
  - Ray & Dispensa, MaloDroid
  - BERserk
  - Most dangerous code...
  - SSL stripping
Rest of this lecture

- Syntax of a cipher
- Some historical ciphers and how they were broken
- The One-Time Pad cipher and its security/insecurity
- Towards practice: Begin stream ciphers and blockciphers
## Four settings for cryptography

<table>
<thead>
<tr>
<th>Pre-shared key?</th>
<th>Security Goal</th>
<th>Confidentiality</th>
<th>Authenticity/Integrity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes (&quot;Symmetric&quot;)</td>
<td>Symmetric Encryption (aka Secret-key Encryption)</td>
<td>Message Authentication Code (MAC)</td>
<td></td>
</tr>
<tr>
<td>No (&quot;Asymmetric&quot;)</td>
<td>Public-Key Encryption</td>
<td>Digital Signatures</td>
<td></td>
</tr>
</tbody>
</table>
Ciphers (a.k.a. Symmetric Encryption)

A cipher is a pair of algorithms Encrypt, Decrypt:

\[ K \rightarrow \text{Encrypt} \rightarrow C \rightarrow \text{Decrypt} \rightarrow K \]

Require that decryption recovers the same message.
Historical Cipher: ROT13 ("Caesar cipher")

Encrypt(K,m): shift each letter of plaintext forward by K positions in alphabet (wrap from Z to A).

Plaintext:  DEFGH
Key (shift):  3
Ciphertext:  FGHKL

Plaintext:  ATTACKATDAWN
Key (shift):  13
Ciphertext:  NGGNPXNGQNJ
Historical Cipher: Substitution Cipher

Encrypt(K,m): Parse key K as a permutation $\pi$ on \{A,… Z\}. Apply $\pi$ to each character of m.

P: ATTACKATDAWN
K: $\pi$
C: ZKKZAMZKYZGT

How many keys?
26! $\approx 2^{88}$
9 million years to try all keys at rate of 1 trillion/sec
Cryptanalysis of Substitution Cipher

CELEBRITY CIPHER
by Luis Campos

Celebrity Cipher cryptograms are created from quotations by famous people, past and present. Each letter in the cipher stands for another.

"UPXEGTHWZHFXYLFHOSLNPFXHM
TPJSXEPXMLVPAGAVGPOSLEEBXOAPMLNPFX,TPJ'CXZPWEBXVBZXEBHOS." — V.W. YXGVHO

Previous Solution: “Time is the cruelest teacher; first she gives the test, then teaches the lesson.” — Leonard Bernstein

TODAY'S CLUE: .espnbas

© 2012 by NEA, Inc., dist. by Universal Uclick 9-20
Quick recall: Bitwise-XOR operation

We will use bit-wise XOR:

\[
\begin{array}{c}
0101 \\
\oplus 1100 \\
\hline
1001
\end{array}
\]

Some Properties:
- \( X \oplus Y = Y \oplus X \)
- \( X \oplus X = 000...0 \)
- \( X \oplus Y \oplus X = Y \)
Cipher Example: One-Time Pad

Key $K$: Bitstring of length $L$

Plaintext $M$: Bitstring of length $L$

Encrypt($K,M$): Output $K \oplus M$

Decrypt($K,C$): Output $K \oplus C$

Correctly decrypts because

$$K \oplus C = K \oplus (K \oplus M) = (K \oplus K) \oplus M = M$$

Q: Is the one-time pad secure?

Bigger Q: What does “secure” even mean?
Evaluating Security of Crypto

Kerckhoff’s Principle: Assume adversary knows your algorithms and implementation. The only thing it doesn’t know is the key.

1. Quantify adversary goals
   - Learn something about plaintext? Spoof a message?
2. Quantify adversary capabilities
   - View ciphertexts? Probe system with chosen inputs?
3. Quantify computational resources available to adversary
   - Compute cycles? Memory?
Breaking Encryption - A Basic Game

**Ciphertext-only attack:** The adversary sees ciphertexts and attempts to recover some useful information about plaintexts.

More attack settings later.
Recovering Partial Information; Partial Knowledge

- Recovering entire messages is useful
- But recovering **partial information** is also be useful

A lot of information is missing here.

But can we say who this is?

- Attacker may know large parts of plaintext already (e.g. formatting strings or application content). The attacker tries to obtain something it doesn’t already know.

\[ M = \text{http://site.com?password=********} \]
“Attacks” versus “Security”

An **attack** is successful as long as it recovers some info about plaintext that is useful to adversary.

Encryption should hide all possible partial information about plaintexts, since what is useful is situation-dependent.
Attacks can succeed without recovering the key

Full break: Adversary recovers $K$, decrypts all ciphertexts.

However: Clever attacker may compromise encryption without recovering the key.
Security of One-Time Pad

**Claim:** If adversary sees only one ciphertext under a random key, then any plaintext is equally likely, so it cannot recover any partial information besides plaintext length.

Ciphertext observed: 10111
Possible plaintext: 00101
→ Possible key: 10010

1. Adversary goal: Learn partial information from plaintext
2. Adversary capability: Observe a single ciphertext
3. Adversary compute resources: Unlimited time/memory (!)
Issues with One-Time Pad

1. Reusing a pad is insecure
2. One-Time Pad is *malleable*
3. One-Time Pad has a long key
Issue #1: Reusing a One-Time Pad is Insecure

HELLOALICE ⊕ Pad = C₁

PWDHAMSTER ⊕ Pad = C₂

HELLOALICE ⊕ Pad = Pad

PWDHAMSTER ⊕ Pad = PWDHAMSTER
Issue #1: Reusing a One-Time Pad is Insecure

Has led to real attacks:
- Project Venona (1940s) attack by US on Soviet encryption
- MS Windows NT protocol PPTP
- WEP (old WiFi encryption protocol)
- Secure routers caught doing this last fall! [link]

\[ C_1 \oplus C_2 = S3CR3T1234 \oplus 3L33THXRRR \]
Issue #2: One-Time Pad is *Malleable*

\[
\text{PAYALICE}\$1 \oplus \text{Pad} = \text{C} \oplus \text{000ALICE00} \oplus \text{000DAVID00} = \text{C'}
\]

\[
\text{Decrypt}(\text{Pad}, \text{C'}) = \text{PAYDAVID}\$1
\]
Issue #3: One-Time Pad Needs a Long Key

**Can prove:** Any cipher as secure as the OTP must have:
Key-length \(\geq\) Plaintext-length

**In practice:** (covered here and next lecture):
- Use *stream cipher*: Encrypt(\(K, m\)) = \(G(K) \oplus m\)
- Add *authentication tag*
- Use *nonces* to encrypt multiple messages
Tool to address key-length of OTP: Stream Ciphers

Stream cipher syntax: Algorithm G that takes one input and produces an very long bit-string as output.

Usually very, very large (petabytes if needed)  
Key/Seed k: 1100..11  
Typically 16 or 32 bytes.  

G(k): 111110100010001110101000101100100111100...

\[ \oplus \text{DONUTSDONUTSDONUTSDONUTSDONUTSDONUTSDONUTSDONUTSDON} \]

Use G(seed) in place of pad.  
Still malleable and still one-time, but key is shorter.
Security goal: When $k$ is random and unknown, $G(k)$ should “look” random.

... even to an adversary spending a lot of computation.

Much stronger requirement that “passes statistical tests”.

**Brute force attack:** Given $y = G(k)$, try all possible $k$ and see if you get the string $y$.

**Clarified goal:** When $k$ is random and unknown, $G(k)$ should “look” random to anyone with less computational power needed for a brute force attack.

(keylength = 256 is considered strong now)
Aside: Fundamental Physical Property of the Universe*

There exist functions (say on bitstrings) that are:
1) Very fast to evaluate
2) Computationally infeasible to reverse

The disparity can be almost arbitrarily large!

Evaluating $y = f(x)$ may only take a few cycles….

… and finding $x$ from $y$ within the lifetime of the universe may not be possible, even with a computer made up of every particle in the universe.

*conjectured, but unproven property
## Computational Strength

<table>
<thead>
<tr>
<th># Steps</th>
<th>Who can do that many?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{56}$</td>
<td>Strong computer with GPUs</td>
</tr>
<tr>
<td>$2^{80}$</td>
<td>All computers on Bitcoin network in 4.5 hours</td>
</tr>
<tr>
<td>$2^{128}$</td>
<td>Very large quantum computer? (Ask Fred+Bill)*</td>
</tr>
<tr>
<td>$2^{192}$</td>
<td>Nobody?</td>
</tr>
<tr>
<td>$2^{256}$</td>
<td>Nobody?</td>
</tr>
</tbody>
</table>

*Not directly comparable but this is an estimate of equivalent power. Quantum computers are most effective against public-key crypto, but they also speed up attacks on symmetric-key crypto. (More next week.*)
Example Stream Cipher: RC4

**Internal state:** Array $s$ of 256 bytes and pointers $i$, $j$

To compute next output byte:

- $i = i + 1 \pmod{256}$
- $j = j + S[i] \pmod{256}$
- swap $S[i]$ and $S[j]$

**Warning:** Broken

Then:

- Output bits are biased in easily detectable ways.... but only retired by major websites in 2016.

**Replacement:** Salsa20/ChaCha, or AES-based methods to be discussed
Pad reuse can still happen with stream ciphers

\[ m_1 \oplus G(k) \rightarrow \text{ciphertext} \]

\[ m_2 \oplus G(k) \rightarrow \text{ciphertext} \]

\[ \ldots \]