CSE543 - Introduction to Computer and Network Security

Module: Cryptography

Professor Trent Jaeger
A historical moment …

• Mary Queen of Scots is being held by Queen Elizabeth …
  ‣ … and accused of treason.
  ‣ All communication with co-conspirators encrypted.

• Walsingham needs to prove complicity.
Intuition

• Cryptography is the art (and sometimes science) of secret writing
  ▸ Less well known is that it is also used to guarantee other properties, e.g., authenticity of data
  ▸ This is an enormously deep and important field
  ▸ However, much of our trust in cryptographic systems is based on faith (particularly in efficient secret key algorithms)
  ▸ … ask Mary Queen of Scots how that worked out.

• This set of lectures will provide the intuition and some specifics of modern cryptography, seek others for additional details (Menezes et. al.).
Cryptography

• Cryptography (cryptographer)
  ‣ Creating ciphers

• Cryptanalysis (cryptanalyst)
  ‣ Breaking ciphers

• The history of cryptography is an arms race between cryptographers and cryptanalysts
Encryption algorithm

- Algorithm used to make content unreadable by all but the intended receivers

\[ E(\text{plaintext}, \text{key}) = \text{ciphertext} \]
\[ D(\text{ciphertext}, \text{key}) = \text{plaintext} \]

- Algorithm is public, key is private

- Block vs. Stream Ciphers
  - Block: input is fixed blocks of same length
  - Stream: stream of input
Hardness

• Inputs
  ‣ Plaintext \( P \)
  ‣ Ciphertext \( C \)
  ‣ Encryption key \( k_e \)
  ‣ Decryption key \( k_d \)

\[
D(E(P, k_e), k_d) = P
\]

• Computing \( P \) from \( C \) is hard, \( P \) from \( C \) with \( k_d \) is easy
  ‣ for all \( P \)s with more than negligible probability
  ‣ This is known as a TRAPDOOR function
  ‣ Devil is in the details ....
Example: Caesar Cipher

- Substitution cipher
- Every character is replaced with the character three slots to the right

Q: What is the key?

SECURITY AND PRIVACY
VHFXULWBDQGSULYDFB
Cryptanalyze this ....

“GUVFVF N TERNG PYNFF”
Cryptanalysis of ROTx

- **Goal:** to find plaintext of encoded message
- **Given:** ciphertext
- **How:** simply try all possible keys
  - Known as a brute force attack

\[1 \quad \text{T F D V S J U Z B M E Q S J W B D Z}
2 \quad \text{U G E W T K V A C N F R T H X C E A}
3 \quad \text{W H F X U L W B D Q G S U L Y D F B}
\]

SECURITY AND PRIVACY
Substitution Cipher

• A substitution cipher replaces one symbol for another in the alphabet
  ‣ Caesar cipher and rot13 are a specific kind (rotation)
  ‣ The most common is a *random permutation* cipher

\[
\begin{array}{cccccccccccccccc}
A & B & C & D & E & F & G & H & I & J & K & L & M \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
C & M & T & E & F & H & P & U & D & X & N & Z & L \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
N & O & P & Q & R & S & T & U & V & W & X & Y & Z \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
O & A & J & R & Y & I & G & W & V & B & S & Q & K \\
\end{array}
\]
Why are substitution ciphers breakable?

• Substitution ciphers are breakable because they don’t hide the underlying frequency of characters. You can use this information if you know the target language frequency count.

• For example, in English ...
  ‣ e, t, a, o, i, n, s, r, h, d, l, u, c, m, f, y, w, g, p, b, v, k, x, q, j, z

• Q: how do you exploit this?
Using frequency ..

- Vg gbbx n ybg bs oybbq, fjrng naq grnef gb trg gb jurer jr ner gbqnl, ohg jr unir whfg ortha. Gbqnl jr ortva va rnearfg gur jbex bs znxvat fher gung gur jbeyq jr yrnir bhe pvvyqera vf whfg n yvggyr ovg orggre guna gur bar jr vaunovg gbqnl.
Using frequency ..

• Vg gbbx n ybg bs oybbq, fjrng naq grnfr gb trg gb jurer jr ner gbqnl, ohg jr unir whfg ortha. Gbqnl jr ortva va rnearfg gur jbex bs znxvat fher gung gur jbeyq jr yrni r bhe puvyqera vf whfg n yvggyr ovg orgg re guna gur bar jr vaunovg gbqnl.

• It took a lot of blood, sweat and tears to get to where we are today, but we have just begun. Today we begin in earnest the work of making sure that the world we leave our children is just a little bit better than the one we inhabit today.

‘r’ appears very frequently so very likely is one of the top frequency letters.
Using frequency ..

• Vg gbbx n ybg bs oybbq, fjrng naq grnef gb trg gb jurer jr ner gbqnl, ohg jr unir whfg ortha. Gbqnl jr ortva va rnearfg gur jbex bs znxvat fher gung gur jbewyq jr yrni r bhe puvyqera vf whfg n yvggyr ovg orggre guna gur bar jr vaunovg gbqnl.

• It took a lot of blood, sweat and tears to get to where we are today, but we have just begun. Today we begin in earnest the work of making sure that the world we leave our children is just a little bit better than the one we inhabit today.

Repeat this process, picking out more letters, then common words, e.g., ‘the’ ... which gives (e to r), (g to t), and (u to h)
Shared key cryptography

- Traditional use of cryptography
- Symmetric keys, where a single key \((k)\) is used for encryption and decryption

\[
D(E(p, k), k)) = P
\]

- All (intended) receivers have access to key
- **Note:** Management of keys determines who has access to encrypted data
  - E.g., password encrypted email
- Also known as symmetric key cryptography
Key size and algorithm strength

• Key size is an oft-cited measure of the strength of an algorithm, but is strength strongly correlated (or perfectly correlated with key length)?
  ‣ Say we have two algorithms, A and B with key sizes of 128 and 160 bits (the common measure)
  ‣ Is A “less secure” than B?
  ‣ What if A=B (for variable key-length algorithms)?

*Implication:* references to key length in advertisements are often meaningless.
Is there an unbreakable cipher?

- As it turns out, yes ....
  ‣ (Claude Shannon proved it)
The one-time pad (OTP)

- Assume you have a secret bit string $s$ of length $n$ known only to two parties, Alice and Bob
  - Alice sends a message $m$ of length $n$ to Bob
  - Alice uses the following encryption function to generate ciphertext bits:
    \[ c_i = m_i \oplus k_i \]
    \[ \sum_{i=0}^{n} c_i = m_i \oplus k_i \]
  - E.g., XOR the data with the secret bit string
  - An adversary Mallory cannot retrieve any part of the data
- Simple version of the proof of security:
  - Assume for simplicity that value of each bit in $k$ is equally likely, then you have no information to work with.
Generic Block Encryption

• Break input into smaller chunks
• Apply \textit{substitution} on smaller chunks and \textit{permutation} on output of the substitution
• Achieves Shannon’s properties of \textit{confusion} and \textit{diffusion}
  ‣ \textbf{Confusion}: Relation between ciphertext and key as complex as possible
  ‣ \textbf{Diffusion}: Relation between ciphertext and plaintext as complex as possible
• Multiple \textit{rounds}
• Plaintext easily recovered
Data Encryption Standard

- Introduced by the US NBS (now NIST) in 1972
- Signaled the beginning of the modern area of cryptography
- Block cipher
  - Fixed sized input
- 8-byte input and a 8-byte key (56-bits+8 parity bits)
- Multiple rounds of substitution, initial and final permutation
Substitution Box (S-box)

• A substitution box (or S-box) is used to obscure the relationship between the key and the ciphertext
  ‣ Shannon's property of **confusion**: the relationship between key and ciphertext is as complex as possible.
  ‣ In DES S-boxes are carefully chosen to resist cryptanalysis.
  ‣ Thus, that is where part of the security comes from.

Example: Given a 6-bit input, the 4-bit output is found by selecting the row using the outer two bits, and the column using the inner four bits. For example, an input "011011" has outer bits "01" and inner bits "1101"; the corresponding output would be "1001".
Permutations Box (P-box)

• A permutations box (or P-box) is used to obscure the relationship between the plaintext and the ciphertext
  ‣ Shannon's property of diffusion: the relationship between plaintext and ciphertext is as complex as possible.
  ‣ DES uses a combination of diffusion and confusion to resist cryptanalysis
Cryptanalysis of DES

• DES has an effective 56-bit key length
• Wiener: $1,000,000 - 3.5 hours (never built)
• July 17, 1998, the EFF DES Cracker, which was built for less than $250,000 < 3 days
• January 19, 1999, Distributed.Net (w/EFF), 22 hours and 15 minutes (over many machines)
• We all assume that NSA and agencies like it around the world can crack (recover key) DES in milliseconds

• Not viable alone - can use Triple DES
Variants of DES

- DESX (XOR with separate keys ≈ 60-bits)
  - Linear cryptanalysis
- Triple DES (three keys ≈ 112-bits)
  - keys $k_1, k_2, k_3$

\[ C = E(D(E(p, k_1), k_2, k_3)) \]
Advanced Encryption Standard (AES)

• International NIST bakeoff between cryptographers
  ‣ Rijndael (pronounced “Rhine-dall”)
• Replacement for DES/accepted symmetric key cipher
  ‣ Substitution-permutation network, not a Feistel network
  ‣ Variable key lengths
  ‣ Fast implementation in hardware and software
  ‣ Small code and memory footprint
Attacking a Cipher

• The attack mounted will depend on what information is available to the adversary
  ‣ **Ciphertext-only attack**: adversary only has the ciphertext available and wants to determine the plaintext
  ‣ **Known-plaintext attack**: adversary learns one or more pairs of ciphertext/plaintext encrypted under the same key, tries to determine plaintext based on a different ciphertext
  ‣ **Chosen-plaintext attack**: adversary can obtain the encryption of any plaintext, tries to determine the plaintext for a different ciphertext
  ‣ **Chosen-ciphertext attack**: adversary can obtain the plaintext of any ciphertext except the one the adversary wants to decrypt
Known-Plaintext Attack

• **Known-plaintext attack**: adversary learns one or more pairs of ciphertext/plaintext encrypted under the same key, tries to determine plaintext based on a different ciphertext
  ‣ Suppose that the adversary knows common messages
    • “Calling all cars”
  ‣ When these messages are encrypted the adversary may use them to extract the key material
    • “Xwggdib wgg xwmn”

• As a result, we will see that adversaries design cryptographic modes to prevent such detection
Symmetric Ciphers and Attacks

• Another Problem: Same plaintext encrypts to same cipher text
  ‣ E(d, k) = c for each d and k
  ‣ What can you do?
Symmetric Ciphers and Attacks

‣ What can you do?
  ‣ Add a salt to the encryption process (like for passwords)
  ‣ Initialization vector

‣ Cipher modes

Electronic Codebook (ECB) mode encryption

Cipher Block Chaining (CBC) mode encryption
Hash Algorithms

• Hash algorithm
  ‣ Compression of data into a hash value
  ‣ E.g., \( h(d) = \text{parity}(d) \)
  ‣ Such algorithms are generally useful in algorithms (speed/space optimization)

• … as used in cryptosystems
  ‣ One-way - (computationally) hard to invert \( h() \), i.e., compute \( h^{-1}(y) \), where \( y=h(d) \)
  ‣ Collision resistant hard to find two data \( x_1 \) and \( x_2 \) such that \( h(x_1) == h(x_2) \)

• Q: What can you do with these constructs?
Hash Functions

• MD4, MD5
  ‣ Substitution on complex functions in multiple passes

• SHA-1
  ‣ 160-bit hash
  ‣ “Complicated function”

• SHA-2
  ‣ 256 to 512 bit hash

• SHA-3
  ‣ Alternative to SHA-2

• Limited formal basis
  ‣ Practical attacks on SHA-1, MD5
Using hashes as authenticators

• Consider the following scenario
  ‣ Prof. Alice has not decided if she will cancel the next lecture.
  ‣ When she does decide, she communicates to Bob the student through Mallory, her evil TA.
  ‣ She does not care if Bob shows up to a cancelled class
  ‣ She wants Bob to show for all classes held

• She and Bob use the following protocol:
  1. Alice invents a secret $t$
  2. Alice gives Bob $h(t)$, where $h()$ is a crypto hash function
  3. If she cancels class, she gives $t$ to Mallory to give to Bob
     – If does not cancel class, she does nothing
     – If Bob receives the token $t$, he knows that Alice sent it
Hash Authenticators

• Why is this protocol secure?
  – \( t \) acts as an authenticated value (authenticator) because Mallory could not have produced \( t \) without inverting \( h() \)
  – Note: Mallory can convince Bob that class is occurring when it is not by simply not delivering \( t \) (but we assume Bob is smart enough to come to that conclusion when the room is empty)

• What is important here is that hash preimages are good as (single bit) authenticators.
• Note that it is important that Bob got the original value \( h(t) \) from Alice directly (was provably authentic)
Hash chain

• Now, consider the case where Alice wants to do the same protocol, only for all 26 classes (the semester)
• Alice and Bob use the following protocol:
  1. Alice invents a secret $t$
  2. Alice gives Bob $h^{26}(t)$, where $h^{26}()$ is 26 repeated uses of $h()$.  
  3. If she cancels class on day $d$, she gives $h^{(26-D)}(t)$ to Mallory, e.g.,
     - If cancels on day 1, she gives Mallory $h^{25}(t)$
     - If cancels on day 2, she gives Mallory $h^{24}(t)$
     .......
     - If cancels on day 25, she gives Mallory $h^{1}(t)$
     - If cancels on day 26, she gives Mallory $t$
  4. If does not cancel class, she does nothing
     – If Bob receives the token $t$, he knows that Alice sent it
Hash Chain (cont.)

• Why is this protocol secure?
  ‣ On day \(d\), \(h^{(26-d)}(t)\) acts as an authenticated value (authenticator) because Mallory could not create \(h^{(26-d)}(t)\) without inverting \(h^{(26-d-1)}(t)\) because for any \(h^k(t)\) she has \(h^j(t)\) where \(26 > j > k\)
  ‣ That is, Mallory potentially has access to the hash values for all days prior to today, but that provides no information on today’s value, as they are all post-images of today’s value
  ‣ Note: Mallory can again convince Bob that class is occurring by not delivering \(h^{(26-d)}(t)\)
  ‣ Chain of hash values are ordered authenticators
• Important that Bob got the original value \(H^{26}(t)\) from Alice directly (was provably authentic)
A (simplified) sample token device

- A one-time password system that essentially uses a hash chain as authenticators.
  - For seed \((S)\) and chain length \((l)\), epoch length \((x)\)
  - Tamperproof token encodes \(S\) in firmware
  - Device display shows password for epoch \(i\)
  - Time synchronization allows authentication server to know what \(i\) is expected, and authenticate the user.

- **Note:** somebody can see your token display at some time but learn nothing useful for later periods.
A question?

• Is there going to come a day where all passwords are useless?
  ‣ Suppose I can remember 16 bytes of entropy (possible?)
    • That is, 16 pseudorandom characters
  ‣ Won’t there come a day when adversaries could still crack?
    • Moore’s law and its corollaries?
Answer: no

• Nope, you just need to make the process of checking passwords more expensive. For example, you can repeat the salted hash many times ...
  ‣ Linear cost speedup?

\[ salt_i, h^{100}(salt_i, pw_i) \]
• MAC
  ‣ Used in protocols to authenticate content, authenticates integrity for data \( d \)
  ‣ To simplify, hash function \( h() \), key \( k \), data \( d \)

\[
MAC(k, d) = h(k \oplus d)
\]
  ‣ E.g., XOR the key with the data and hash the result

• Q: Why does this provide integrity?
  ‣ Cannot produce \( MAC(k,d) \) unless you know \( k \)
  ‣ If you could, then can invert \( h() \)

• Exercise for class: prove the previous statement
A simple proof

• Setup: you know d and have an polynomial-time algorithm X(d) that produces MAC(k,d) without k (assume d is known).

• Suppose X() exists:

\[ d = 0 \]
\[ \text{then, } X(d) = h(k \oplus 0) = h(k) \]

• There are two possible explanations
  ‣ k is constant (which it is not)
  ‣ X(d) knows or receives k from input (which by definition it does not)
  ‣ ... a contradiction.
HMAC

• MAC that meets the following properties
  ‣ Collision-resistant
  ‣ Attacker cannot compute proper digest without knowing $K$
    • Even if attacker can see an arbitrary number of digests $H(k+x)$

• Simple MAC has a flaw
  ‣ Block hash algorithms mean that new content can be added
  ‣ Turn $H(K+m)$ to $H(K+m+m')$ where $m'$ is controlled by an attacker

• $\text{HMAC}(K, d) = H(K + H(K + d))$
  ‣ Attacker cannot extend MAC as above
  ‣ Prove it to yourself
Birthday Paradox

- Birthday paradox: the probability that two or more people in a group of 23 share the same birthday is >than 50%

- General formulation
  - function f() whose output is uniformly distributed
  - On repeated random inputs \( n = \{ n_1, n_2, \ldots, n_k \} \)
    - \( \Pr(n_i = n_j) = 1.2k^{1/2} \), for some \( 1 \leq i,j \leq k, 1 \leq j < k, i \neq j \)
    - E.g., \( 1.2(365^{1/2}) \sim 23 \)

- Q: Why is the birthday paradox important to hash functions?
Using Crypto

• Suppose you (Alice) want to send a document securely to another party (Bob)
  • You have each obtained a secret key
  • Obtained in some secure fashion (key distribution, later)
• How do you send the document such that only Bob can read it?
• How do you send the document such that Bob knows it is from Alice?
Basic truths of cryptography …

- Cryptography is not frequently the source of security problems
  - Algorithms are well known and widely studied
    - Use of crypto commonly is … (e.g., WEP)
  - Vetted through crypto community
  - Avoid any “proprietary” encryption
  - Claims of “new technology” or “perfect security” are almost assuredly snake oil
Why Cryptosystems Fail

• In practice, what are the causes of cryptosystem failures
  ‣ Not crypto algorithms typically
Case Study

• ATM Systems
  ‣ Some public data
  ‣ High value information
  ‣ Of commercial enterprises, banks have most interest in security

• How do they work?
  ‣ Card: with account number
  ‣ User: provides PIN
  ‣ ATM: Verifies that PIN corresponds to encryption of account number with PIN key (offset can be used)

• Foundation of security
  ‣ PIN key (can obtain PIN and forge cards)
Simple Fraud

• Insiders
  ‣ Make an extra card; special ops allow debit of any acct

• Outsiders
  ‣ Shoulder surfing; fake ATMs; replay pay response

• PINs
  ‣ Weak entropy of PIN keys; limit user PIN choices; same PIN for everyone

• User-chosen PINs
  ‣ Bad; Store encrypted in a file (find match); Encrypted on card

• Italy
  ‣ Fake ATMs; Offline ATMs (make several copies of card)
More Complex Issues

• PIN key derivation
  ‣ Set terminal key from two shares
  ‣ Download PIN key encrypted under terminal key

• Other banks’ PIN keys
  ‣ Encrypt ‘working keys’ under a zone key
  ‣ Re-encrypt under ATM bank’s working key

• Must keep all these keys secret
Products Have Problems

- Despite well understood crypto foundations, products don’t always work securely
  - Lose secrets due to encryption in software
  - Incompatibilities (borrow my terminal)
  - Poor product design
    - Back doors enabled, non-standard crypto, lack of entropy, etc.
  - Sloppy operations
    - Ignore attack attempts, share keys, procedures are not defined or followed
- Cryptanalysis sometimes
  - Home-grown algorithms!, improper parameters, cracking DES
Problems

• Systems may work in general, but
  ‣ Are difficult to use in practice
  ‣ Counter-intuitive
  ‣ Rewards aren’t clear
  ‣ Correct usage is not clear
  ‣ Too many secrets ultimately

• Fundamentally, two problems
  ‣ Too complex to use
  ‣ No way to determine if use if correct
What Can We Do?

• Anderson suggests
  ‣ Determine exactly what can go wrong
    • Find all possible failure modes
  ‣ Put in safeguards
    • Describe how preventions protect system
  ‣ Correct implementation of safeguards
    • Implementation of preventions meets requirements
  ‣ Decisions left to people are small in number and clearly understood
    • People know what to do

• Problems of security in general
Important principles

• Don’t design your own crypto algorithm
  ‣ Use standards whenever possible
• Make sure you understand parameter choices
• Make sure you understand algorithm interactions
  ‣ E.g. the order of encryption and authentication
    • Turns out that authenticate then encrypt is risky
• Be open with your design
  ‣ Solicit feedback
  ‣ Use open algorithms and protocols
  ‣ Open code? (jury is still out)
Building systems with cryptography

• Use quality libraries
  ‣ SSLeay, lim (from Lenstra), Victor Shoup’s library, RSAREF, cryptolib
  ‣ Find out what cryptographers think of a package before using it

• Code review like crazy

• Educate yourself on how to use libraries
  ‣ Caveats by original designer and programmer
Common issues that lead to pitfalls

- Generating randomness
- Storage of secret keys
- Virtual memory (pages secrets onto disk)
- Protocol interactions
- Poor user interface
- Poor choice of key length, prime length, using parameters from one algorithm in another