****

Information System Security Course Pack [SOEN 321]

Created By Erik Stodola

September 8th to December 7th for Fall Term [2015-2016], SOEN 321 with Mohammad Mannan

****

Midterm Up To Page 33

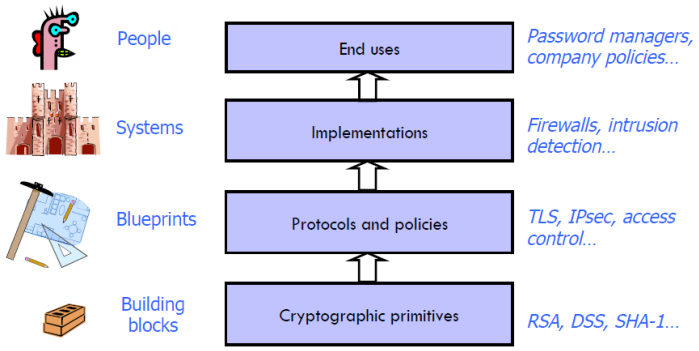
****

Lecture #1(A): Introduction to Security Context

Why Security is Difficult

- http://www.cs.cornell.edu/~shmat/

Security Stack



All defense mechanisms must work correctly and securely 4

Security vs. Correctness

- **System Correctness:** System satisfies specification.
  - For reasonable input, get reasonable output.
- **System Security**: System properties preserved in face of attack.
  - For unreasonable input, output not completely disastrous.
- **Main Difference:** Active interference from adversary.

Bad News for Security and Privacy

- Security often **not** a primary consideration
  - Performance and usability take precedence.
- "I've got nothing to hide"
- Feature-rich systems may be poorly understood.
- Implementations are buggy.
- Many attacks are **non-technical** in nature.
  - Phishing, social engineering, etc.
  - Understand human aspects, psychology.

What can we trust? What do we trust?

What code to trust?

- What code can we trust?

    - Consider “login” or “su” in Unix/Linux.
- Is Windows/OSX binary reliable?
    - Does it send your password to someone?
    - Does it have backdoor for a “special” remote user?
- Can’t trust the binary
    - Let’s use open-source: Ubuntu?
    - Check source code, or write your own (and compile).
- Who wrote the compiler?
- Consider a Trojaned compiler:
    - Compiler looks for source code that looks the login process, inserts backdoor into it.
- Ok, inspect the source code of the compiler… Looks good? Recompile the compiler!
    - Does this solve the problem?

Reflections on Trusting Trust

- Author Ken Thompson
- Turing Award Lecture
- Creator of the UNIX operating system
    - With Dennis Ritchie.

Ken Thompson

- Creator of the system program language **B**
    - A predecessor to the **C** language.
- 1983 – Joint Turing Award with Dennis Ritchie for their work on UNIX.
- 1999 – National Medal of Technology awarded by Bill Clinton.

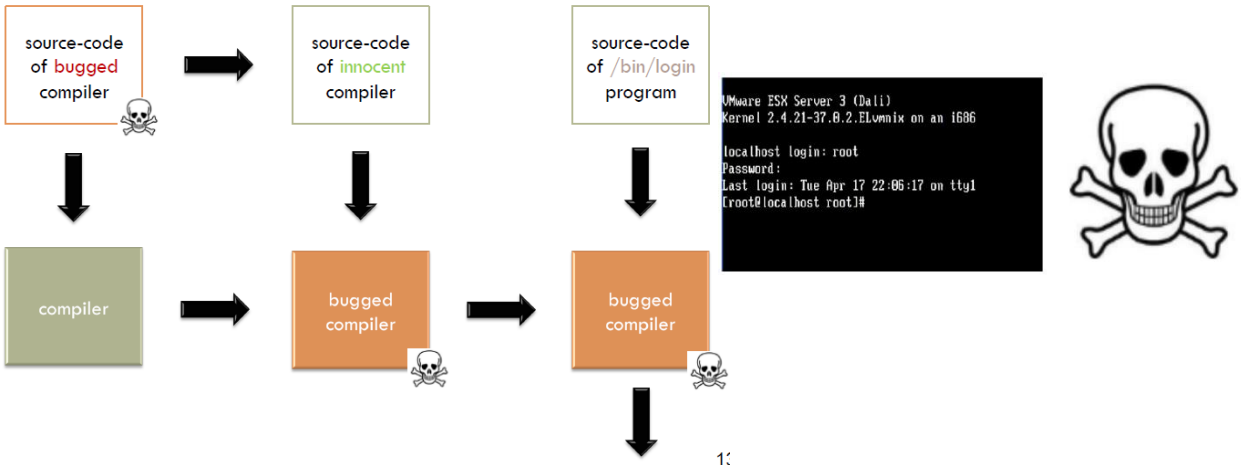
A Bugged Compiler

```
compile(s)
char *s;
{
        ...
}
```

```
compile(s)
char *s;
{
        if(match(s, "pattern")) {
                compile("bug");
                return;
        }
        ...
}
```

```
compile(s)
char *s;
{
        if(match(s, "pattern1")) {
                compile ("bug1");
                return;
        }
        if(match(s, "pattern 2")) {
                compile ("bug 2");
                return;
        }
        ...
}
```

What Happens



13

Moral

- "The moral is obvious. You can't trust code that you did not totally created yourself. (Especially code from companies that employ people like me.)"
- "No amount of source-level verification or scrutiny will protect you from using untrusted code."

Is this a real problem?

- Yes, it happened to a Delphi compiler in 2009!
  - Win32/Induc-A
- http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?name=Virus%3AWin32%2FInduc.A#tab=2

What Else Can Go Wrong?

- CPU
  - AES NI, random number generation
  - Other number theoretic attacks (e.g., bugged prime number generator)
- BIOS
  - Remote Management Tools (Intel AMT)
  - Firmware (From any device, E.g, Ethernet)
  - Crypto Engines

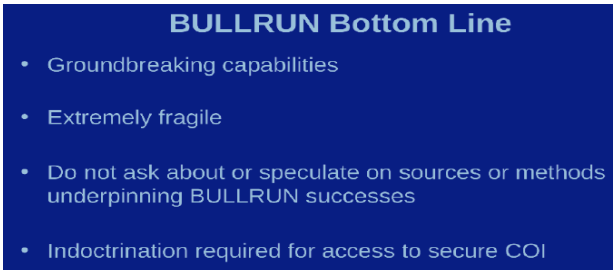
Optional Reading

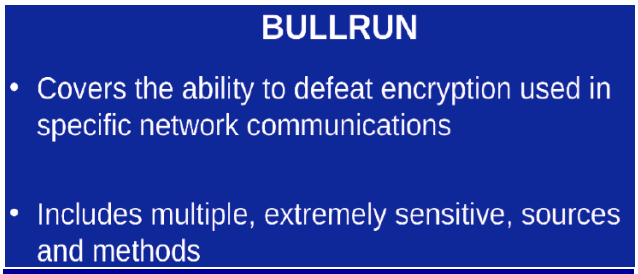
- Reflections on trusting trust
  - Communications of the ACM (August 1984)
  - https://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf
- Malicious Cryptography (book, 2004)
  - Adam Young and Moti Yung
- Designing and Implementing Malicious Hardware
  - USENIX LEET 2008
  - https://www.usenix.org/legacy/event/leet08/tech/full_papers/king/king.pdf
- Beware of Snake Oil (Blog Post, 1991)
  - http://www.philzimmermann.com/EN/essays/SnakeOil
- Detailed Time-Line
  - http://www.eff.org./nsa-spying/timeline
  - See also: The Open Crypto Audit Project: Our Story

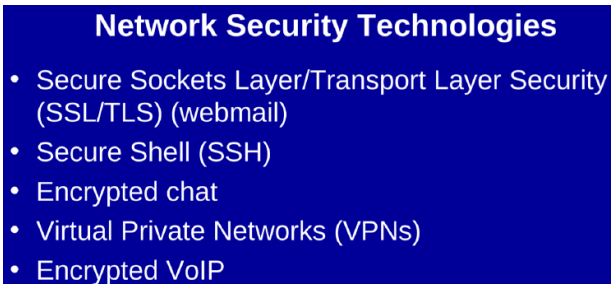
        - Kenneth White & Matthew Green
        - DEF CON 22
- NYT, Propublica, Guardian: NSA spend $250 M/yr to counter & undermine "**the use of ubiquitous encryption across the internet"**
- NIST standards "intentionally weakened"
- BULLRUN: NSA actively working to "Insert vulnerabilities into commercial encryption systems, IT systems, networks, and endpoint communication devices used by targets" NYT, 2013/09/05

The Bullrun Project

- http://www.spiegel.de/media/media-35532.pdf





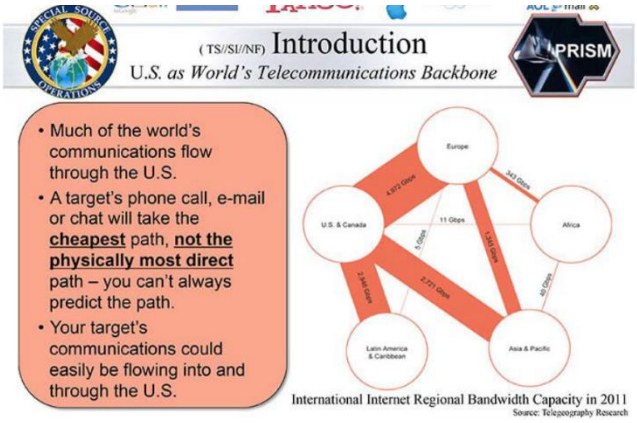


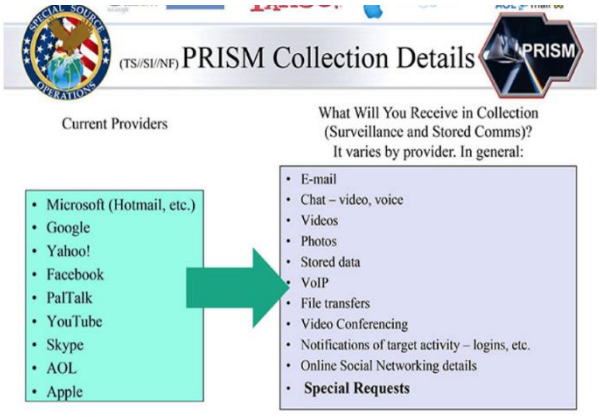
Attacking Randomness

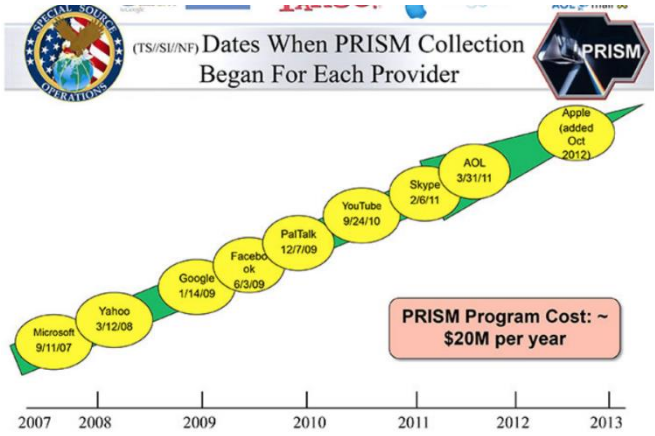
- Dual Elliptic Curve Deterministic Random Bit Generator (DUAL_EC_DRBG)
- Parts of Bullrun
- Implemented in RSA BSAFE crypto library (since 2004)
- Became known in 2013 (RSA was paid US $10 Million)
- Weaknesses are known to security community since 2007.
- But still was included in Windows Vista (2007); removed in Windows 10

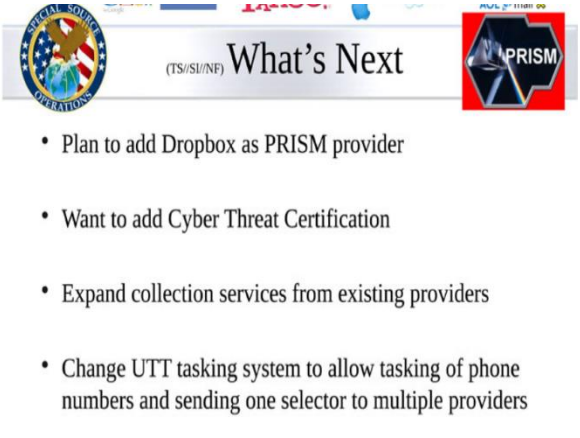
The PRISM Program

- https://nsa.gove1.info/dni/prism.html









What's next for you?

- **Step 1:** We can at least be aware of what's going on.
- **Step 2:** Use crypto where possible, and reduce info leakage.
- **Step 3:** Improve current crypto/security tools; insert "security" to whatever your design/implement.

Security Mindset

- "Think like an attacker, but do not attack"
- https://www.schneier.com/blog/archives/2008/03/the_security_mi_1.html

Some Highlights

- Uncle Milton Industries has been selling ant farms to children since 1956. Some years ago, I remember opening one up with a friend. There were no actual ants included in the box. Instead, there was a card that you filled in with your address, and the company would mail you some ants. My friend expressed surprise that you could get ants sent to you in the mail.
- I replied: "What's really interesting is that these people will send a tube of live ants to anyone you tell them to."
- This kind of thinking is **not natural** for most people. It's not natural for Engineers. Good engineering involves thinking about how things can be made to work; the security mindset involves thinking about how things can be made to fail. It involves **thinking like an attacker, an**

adversary or a criminal. You don't have to exploit the vulnerabilities you find, but if you don't see the world that way, **you'll never notice most security problems**.

- Must never use any attack tools for "unethical" purposes.
- Never use them for personal gain.
- Also consider legal aspects.
- Make the world a better place.

****

Lecture #1(B): Introduction to Cryptography

What's in it?

- Basic tools of cryptography.
- A little bit of cryptanalysis.
- How and where to use crypto tools?
    - How to choose between different tools?
- What crypto cannot achieve?
    - More than you what cryptographers may concede?
- Few usage examples.

Do we use Cryptography?

- Yes we do.
    - Since when?
- Which of the following does not involve crypto?
    - OPUS Card
    - Bank Machines
    - Facebook Login
    - Cellphone Call

Things to Remember

- **Cryptography** and **security** are **NOT** the **same**.
- Always think from three perspective (at least)
    - Defenders' view point
    - Attackers' view point
        - They are as capable as you are (or more!)
        - Your opponents are humans, not machines.
    - How much it will cost, to protect & to attack.
- What is the threat model?
    - Assumptions about the system & operating environment.
    - Ex. Telephone banking PIN vs. Login Password
- Security is an **arms race**.
    - You just need to stay one step ahead.

Before We Begin

- Use slides as pointers to topics
    - Content on slides is (most likely) incomplete
    - Printing the slides may not be worthwhile.
- Class participation is expected.
    - Class discussion is used in exam questions.

- Sources of slide content.
    - Two previous course instances
    - HAC
    - Wikipedia
    - Stallings Book

Cryptography (Definition)

- "Cryptography is the study of **mathematical techniques** related to aspects of information security such as **confidentiality, data integrity, entity authentication, and data origin authentication.**
- Cryptography is not the only means of providing information security, but rather one **set of techniques**" (HAC, Definition 1.1, pg.4)

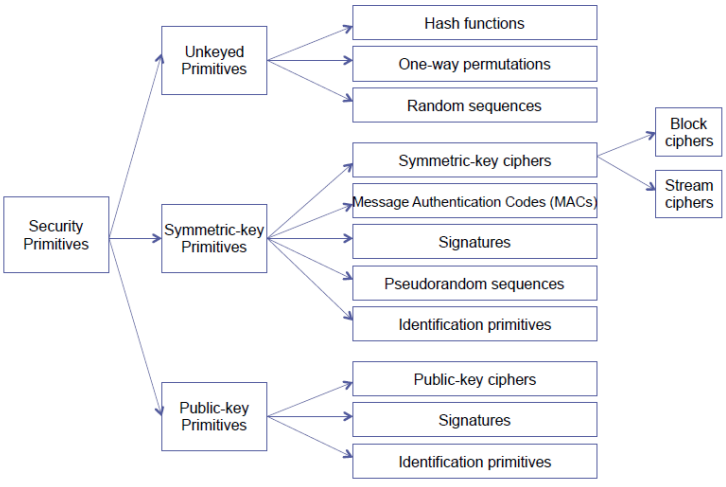
Crypto (Goals)

- **Confidentiality**: How do you prevent others from reading a message?
- **Integrity:** How do you know someone didn't replace or modify a message?
- **Data Origin Authentication:** How do you know message originated from a certain source?
- **Entity Authentication:** How do you know that a person is who they claim to be (passwords)?
- **Non-Repudiation:** How to prevent someone from denying past commitments or actions?

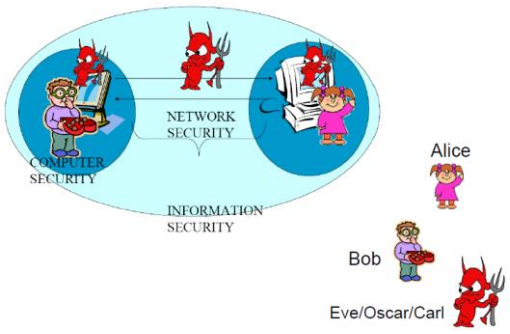
Cryptology

- Cryptography + cryptanalysis = cryptology.
- **Cryptanalysis**
    - Study of code breaking (breaking of crypto systems)
- Good cryptographers known cryptanalysis
    - Must know how systems are broken if you're going to create secure systems.

Crypto Tools (HAC, Fig 1.1; Simplified)



Communication Model

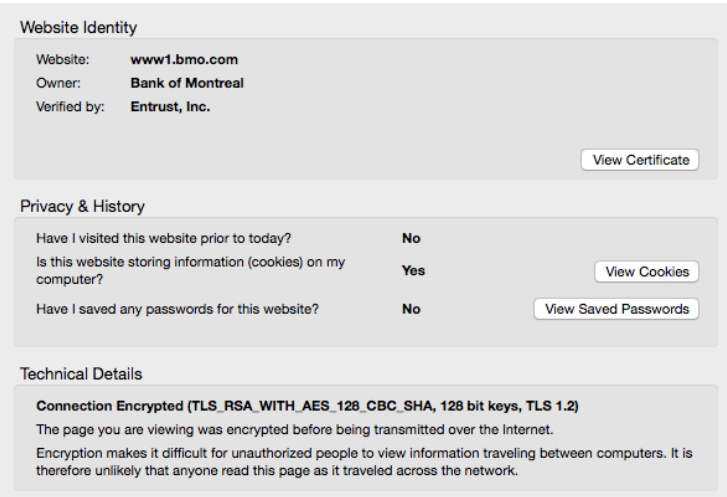


Attacker Model

- Passive vs. Active Attacker

- What computational resources the attacker has?
- What does the attacker known about a system?
    - Cryptosystems used, protocols
- What are the **assumptions?**
    - Encryption keys are shared via a secure channel.

Encryption: Use



How long will it take to brute force an (assume 2^{55}AES / sec):
AES 256-bit key: **509 x 10^{38}** trillion years
AES 128-bit key: **149** trillion years

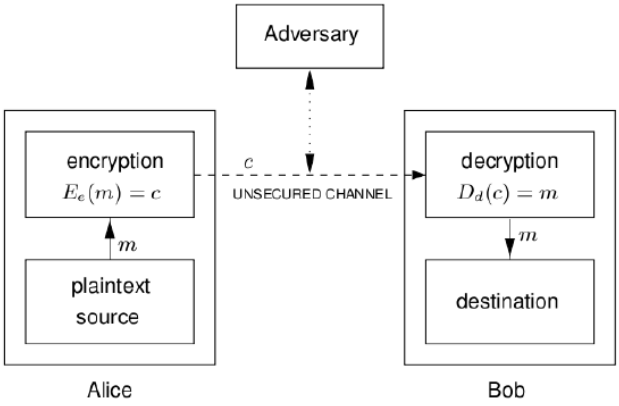
On Key Size

- "Virtual Matrix Encryption is a symmetric encryption algorithm with 1 Mega Bit Key. It is incomparable to AES with only 256 Bit or any other symmetric cypher in existence. Furthermore, its huge key size assures that this technology is future proof – it is unfeasible to do a brute force attack on a Mega Bit Key."
- **Source:** http://www.meganet.com/meganet-products-encryptionsoftware.html
- **See Also:** https://www.schneier.com/blog/archives/2007/06/perpetual_dogho.html

Encryption: Goals

- Data confidentiality
- Protect a large amount of data with "short" secrets.
    - Similar to physical lock-and-key?
- Make it "difficult" for those without the secret key
    - Efficient retrieval with the correct key.

Encryption: Simple Model



What to hide: **key + plaintext**
Symmetric key encryption: *d* is **easily** derived from *e*
Public key encryption: deriving *d* from *e* is **infeasible**
Cipher = *E* & *D*

Encryption: Terminologies

- **Plaintext**: The original message.
- **Ciphertext:** The encoded message.
- **Cipher:** Algorithms for transforming plaintext to ciphertext and vice-versa
  - Block and stream ciphers
- **Key:** Info used in cipher known only to sender/receiver.
- **Encipher (Encrypt):** Converting plaintext to cipher text.
- **Decipher (Decrypt):** Recovering ciphertext from plaintext.
- **Substitution:** Each element is mapped into another.
- **Transposition:** Rearrangement of elements.

Kerckhoff's Principle (1883)

- **Security should depend only on the key**
  - Don't assume enemy won't know algorithm.
  - Can capture machines, disassemble programs, etc.
  - Too expensive to invent new algorithm if it is compromised.
- **Security by obscurity doesn't work**
  - Look at history of examples; Content Scramble System (CSS), A5/1
  - Better to have open scrutiny by experts

How to Attack Encryption

- **Cryptanalysis**
  - Search for weaknesses in the algorithms
  - (Partial) knowledge about plaintext and ciphertext.
- **Brute-Force**
  - Try all possible keys (N), on average N/2 trials are required.
- **Many Others**
  - Malware, key/screen logger, physical and side-channel attacks, implementation bugs, key backdoor, legal means.

Attacks on Encryption Schemes

- **Ciphertext-Only Attack**: Deduce the decryption key or plaintext by only observing ciphertext.
- **Known-Plaintext Attack:** Using a quantity of plaintext and corresponding ciphertext.
- **Chosen-Plaintext Attack:** Choose plaintext and is then given corresponding ciphertext.
- **Adaptive Chosen-Plaintext Attack:** Chosen plaintext attack where the choice of plaintext may depend on the ciphertext received from previous requests.
- **Chosen-Ciphertext Attack:** Select the ciphertext, then the corresponding plaintext is given.
- **Adaptive Chosen-Ciphertext Attack:** Chosen ciphertext attack where the choice of ciphertext may depend on the plaintext received from previous requests.

Encryption – History

- Classic Ciphers
  - Pen & Paper, simple mechanical machines.
  - Egyptians, Greeks, Romans, Hebrew Scholars
- World War I
  - Vernam Cipher (one time pad)
- World War II
  - Complex electro-mechanical machines
  - Engima
- Modern Crypto
  - Depends on, computers and math.
  - Data Encryption Standard (DES) [1975]
  - Public Key Systems [1976]
  - Common People Use

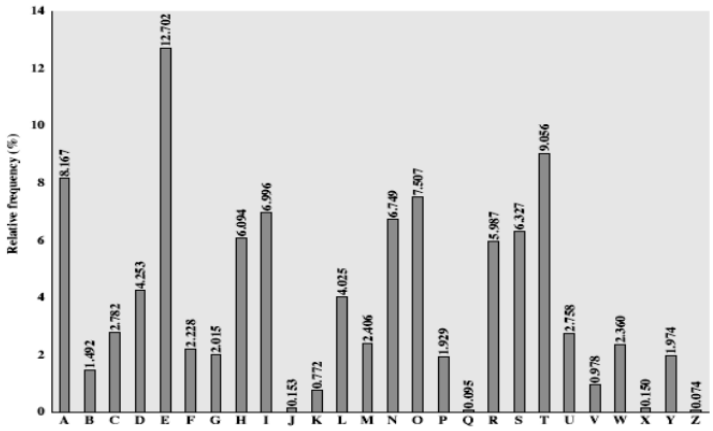
Substitution Ciphers

- Simple substitution (monoalphabetic)
  - Each symbol in plaintext is replaced by another symbol according to some fixed permutation.
- Let A be the set of English characters, A = {a, b, c, … , z}
- Key k (e=d) in K maps plaintext to ciphertext, using character in A
- Example Below
  - Plain: abcefghijklmnopqrstuvwxyz
  - Cipher: DKVQFIBJWPESCXHTMYAUOLRGZN
- Key space is also possible mapping, $|K| = 26! = 4*10^{26}$
- Does a large key space mean it is secure? **No**!

Brute Force Key Search

- Given: ciphertext c, and the cipher used.
- For each key k in K
  - Calculate m' = $D_k(c)$
  - Does m' look like a real message?
    - Yes -> Done!
    - No -> Continue
- Expect to try $|K|/2$ keys before correct match.

English Letter Frequencies



Language Redundancy & Cryptanalysis

- Monoalphabetic ciphers retain their relative letter frequencies!
- Count relative letter frequencies.
- Make guesses based on plaintext language frequencies.
- Frequency analysis, history
    - Text analysis of the Quran
    - Al-Kindi (~800)

Shift/Caesar Cipher

- Simple substitution cipher
- Each letter is replaced by another
- For alphabet A, |A| = s.
    - $e(m_i) = c_i = (m_i + k) \bmod s$, o <= I <= s-1
    - $d(c_i) = (c_i - k) \bmod s$
    - Caesar Cipher : k =3
- **How to break this?**
    - How many keys are there?
    - Plaintext language

Meaning of Mod (1/3)

- Can be used as a binary operator or congruence relation.
- As a binary operator:
    - **a mod n = a – (⌊a/n⌋ * n), for n ≠ 0**
    - n is called the **modulus**
    - E.g. 7 mod 3
        - = 7 - (⌊7/3⌋ ∗3)
        - = 7 – (2 * 3)
        - = 1
- If **r = a mod n**
    - then **a = k * n + r**
    - **k, r**: integers
- As a congruence relation:
    - Mod expresses that two arguments have the same remainder with respect to a given modulus.

        - $a \equiv b \pmod{n}$
        - n divides (a – b)
        - implies: a mod n = b mod n
- $7 \equiv 4 \pmod{3}$ expresses that both 7 and 4 have a remainder of 1 when divided by 3
- $-11 \equiv 17 \pmod{7}$
- More properties:
    - If $a \equiv b \pmod{n}$ then $b \equiv a \pmod{n}$
    - $a \equiv a \pmod{n}$
    - If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$ ***Chain Rule**

Polyalphabetic Substitution

- Multiple substitution alphabets
    - Bellaso / Vigenere Cipher (1553/1586)
    - Enigma (1920's)
- More resistant to frequency analysis.

Transposition Cipher

- For a simple transposition cipher
    - With a fixed period t,
    - Encryption involves grouping the plaintext into block of t characters
    - Applying to each block a single permutation e on the numbers 1 to t.
- Example: t =6, e = (6 4 1 3 5 2)
    - m = CAESAR
    - c = RSCEAA
    - d = (3 6 5 2 5 1)
- Transposition ciphers that operate on characters have the same letter frequencies as original plaintext.
- Transposition ciphers can be made much more secure by performing more than one stage of transposition.
    - Eliminates the ease of simply guessing the column order (the structure is diffused)

Block & Stream Ciphers

- Block Cipher
    - Plaintext message is broken into fixed-length blocks before encryption.
    - One block is processed at time.
    - Most modern ciphers are block ciphers.
- Stream Cipher
    - Block length is one.
    - Requires only limited buffering of data.

HAC Pointers

- Chapter 1
    - Sections: 1.2, 1.3.1, 1.4, 1.5.1, 1.5.2, 1.5.4, 1.8.1, 1.8.2, 1.8.4, 1.13
- Chapter 7
    - Sections 7.2, 7.4

Other Overviews

- Public key encryption
- Hash
- Message Authentication Code (MAC)
- Digital Signatures

****

Lecture #2: Stream Ciphers

Stream Ciphers (Simple Classification)

- One-Time Pad
- Synchronous Stream Ciphers
  - Keystream is generated **independently** of plaintext and of ciphertext.
- Self-synchronizing stream ciphers
  - Key-stream is generated as a function of the key and a fixed number of previous ciphertext digits.

Vernam Cipher

- Definition 1.39 (HAC):
- The Vernam Cipher is a stream cipher defined on the alphabet A = {0,1}
- A binary message m_1, m_2 … m_t is operated on by a binary key string $k_1k_2…k_t$ of the same length to produce a ciphertext string $c_1c_2…c_t$ where: $\mathbf{c_i = m_i \oplus k_i}$, $1 \le i \le t$

One-Time Pad

- Vernam cipher "can" use any bit source as a keystream:
  - E.g, music, picture, or video files.
- But for **real** security it should be:
  - Random
  - Never used again
- Vernam Cipher with such a keystream is called a **one-time pad**.
- The one-time pad is provably secure.

Unconditional Security

- One-time pad is unconditionally secure against a ciphertext-only attack.
  - This means that the uncertainty in the plaintext, after observing the ciphertext, must be equal to the a priori uncertainty about the plaintext.
  - Observation of the ciphertext provides no information to an adversary.

One-Time Pad: Keystream

- Very important that keystream is:
  - **Not Re-Used**: Security is gone as soon as second encryption performed with same key.
  - **Random:** If attacker can guess the key or limit the possibilities, then they could discover the plaintext.
- **Challenge:** How do you keep getting new and random keys that are as long as your message?
  - If easy, we can communicate securely!
  - Could pre-send them (e.g., a DVD containing only random bits), practical? **No**

One-Time Pad: Provably Secure

- The probability of a key bit being 1 or 0 is exactly equal to ½ (Assumed).

- The plaintext bits are not balanced. Let the probability of 0 be x and then the probability of 1 turn out to be 1 – x.
- The probability of ciphertext bits:
- The probability of a ciphertext bit being 1 or 0 is equal to (1/2)x + (1/2)(1-x) = ½
    - No better than random guess.

m_i	prob	k_i	prob.	c_i	prob.
0	x	0	½	0	½ x
0	x	1	½	1	½ x
1	1-x	0	½	1	½ (1-x)
1	1-x	1	½	0	½ (1-x)

Generating Keystreams

- Most stream ciphers substitute a keyed "cryptographically-secure pseudo-random" keystream for a random keystream.
- Function that takes a **fixed-length key (seed)** and produces an **unbounded** bit stream.
- If the function is a pseudo-random number generator, it produces "**statistically random"** numbers, meaning it passes standard tests for randomness.

Cryptographically (Secure)

- Means it should be computationally infeasible to predict the next bit given a complete history of past bits (**next-bit test**).
- The number of 1's and 0's should be equal (**balanced**).

Some Other Types of Security

- **Unconditional Security**
    - The uncertainty in the plaintext, after observing the ciphertext, must be equal to the a priori uncertainty about the plaintext. Observation of the ciphertext provides no information to an adversary.
- **Computational Security**
    - Level of computation required to defeat it (using beast attack known) >> the computational resources of the adversary.

Pseudo-Random Number Generators

- A given PRNG is a **deterministic function** that takes a fixed-length key (seed) and produces an unbounded bit stream.
    - **Same Seed, Same Sequence**
- Produces "statistically random" numbers, meaning it passes standard tests for randomness.
- Cryptographically secure, if
    - Unpredictable ("next-bit" test)
    - Balanced

RC4

- Proposed by Ron Rivest in 1987
- Initially a trade secret, but was posted on Internet anonymously in 1994.
- Very simple algorithm, byte-oriented, fast in s/w
- Variable-length key: 40 – 2048 bits
- Used with **SSL, WPA, WEP, Skype, Opera Min**

RC4 – Initialization **RC4 – Key Stream**

```
for i from 0 to 255
  S[i] := i
  T[i] := key[i mod keylen]
endfor
j := 0
for i from 0 to 255
  j := (j + S[i] + T[i]) mod 256
  swap(S[i], S[j])
endfor
```

```
i := 0
j := 0
while GeneratingOutput:
  i := (i + 1) mod 256
  j := (j + S[i]) mod 256
  swap(S[i], S[j])
  k := S[(S[i] + S[j]) mod 256]
  output k
endwhile
```

RC4 – Cryptanalysis

- WEP Attacks
  - Aircrack-ptw can break 104-bit RC4 keys of WEP-128 in under a minute
- More Recent Attacks
  - "All Your Biases Belong to Us: Breaking RC4 in WPA-TKIP and TLS"
    - https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/vanhoef
  - "Attacks Only Get Better: Password Recovery Attacks Against RC4 in TLS"
    - https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/garman

Performance

- Can be easily tested via OpenSSL:
  - Openssl Speed
  - https://www.openssl.org/docs/manmaster/apps/speed.html
  - Use –evp switch from h/w acceleration
- See also
  - https://calomel.org/aesni_ssl_performance.html

Entropy Sources

- Hardware vs. Software Sources
- Sound/video input, disk drives
- Elapsed time between emissions of particles during radioactive decay.
- Linux (/dev/random)
  - Mouse, keyboard activities
  - Disk I/O
  - Interrupts
- Embedded Devices / IOTS (Input/output Transition System)
- CPU Support
  - Intel RdRand:
    - https://software.intel.com/en-us/articles/intel-digital-random-number-generator-drng-software-implementation-guide
  - Can be backdoored?
    - https://sharps.org/wp-content/uploads/BECKER-CHES.pdf
- Online Services
  - Random.org (Support HTTPS)
  - Should not be trusted for any security applications

****

Lecture #3-4: Block Ciphers

Block Ciphers (Symmetric Key)

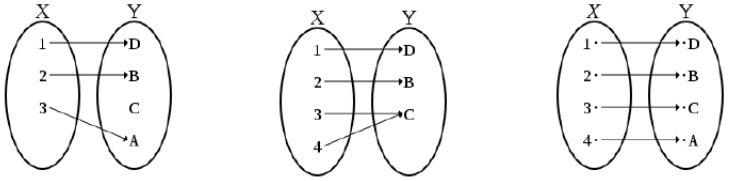
- **Most Used Crypto Tool**
- Used for:
  - **Data confidentiality**
  - MACs
  - PRNGS (Pseudorandom Number Generators)
  - Authentication
- Very efficient software and hardware implementations
  - Can be used in low-resource devices

Definition

- A function that maps n-bit plaintext blocks to n-bit ciphertext blocks
  - N is called the block length (DES: 64 bits, AES: 128 bits)
- Parametrized by k-bit key K (chosen at **random**)
- Encryption: C = E(M,K), Decryption: M = D(C,K)
- If K is k bits long, the number of keys is 2^k
  - How many bits are required to represent a 3-digits PIN? ($\log_2 1000$)
  - 2^{10}
- The encryption function must be one-to-one.
  - Why is this important?
- The encryption function is a bijection, defining a permutation on n-bit vectors.
- Each key potentially defines a different bijection.
- If each k-bit key is **equi-probable,** each defines a different bijection, the entropy of key space is k.

Bijection

- E_k must be a **bijection** as the process must be reversed (decrypted)
  - **Bijection:** on-to-one and onto.
  - **One-To-One**: Each element in X maps to at most one element in Y (X = plaintext, Y = ciphertext)
  - **Onto:** Each element in Y maps to at least one element in X.



Practical Security

- A block cipher:
  - **Totally broken** if key is recoverable
  - **Partially broken** if (some) plaintext is recoverable.
- Standard assumptions (in addition to **Kerckhoff's)**:
  - Attacker has access to all transmitted ciphertext.

Attack Analysis

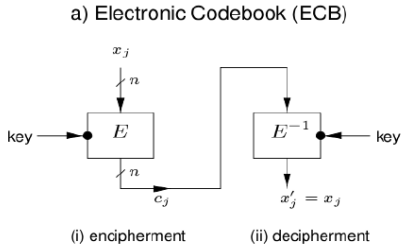
- Complexity of an attack
  - **Data Complexity**, expected number of required input data units (ciphertext blocks)
  - **Storage Complexity**, expected number of required storage units.
  - **Processing Complexity**, expected number of required operations on data.
    - Possible Parallelization?
  - Computationally secure if
    - Data complexity is 2^n : depends on **block size**
    - Processing Complexity is 2^k : depends on **key size**

Modes of Operation

- How to employ block ciphers for large messages
  - Dividing messages
  - Padding the last block
- Basic modes: ECB, CBC, OFB, CFB
  - Standardized for DES operations (1981)
  - More modes defined for AES
  - Can also be used with public key encryption
- Initialization Vector (IV):
  - A block of data used in addition to the input message.
  - Randomize the encryption process.

1) Electronic Code Book (ECB)

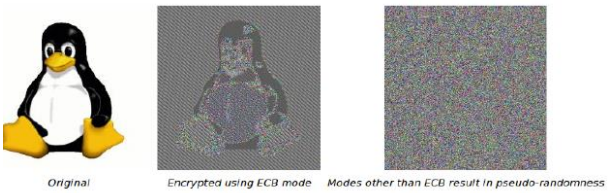
- Encryption: $c_j \leftarrow E_K(x_j)$
- Decryption: $x_j \leftarrow E^{-1}_K(c_j)$



- Identical plaintext (under the same key) result in identical ciphertext.
- Blocks are enciphered independently of other blocks.
- Bit errors in a single ciphertext affect decipherment of that block only

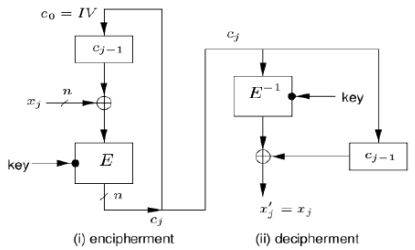
Weakness of ECB

- Does not hide data patterns.
- Malicious substation of ciphertext is possible.
- When can we use this mode?
- When not to use it?
  - Multi-block messages
  - Keys are reused for more than a single block.



2) Cipher Block Chaining (CBC)

b) Cipher-block Chaining (CBC)

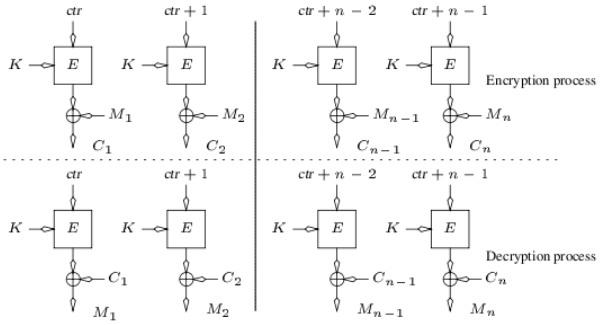


- Widely used, but has known weaknesses...
  - See: dergln.de/uploads/articles/11_09/beast_part2/ssl_jun21.pdf
- Encryption: $c_o \leftarrow IV$, $c_j \leftarrow E_K(c_{j-1} \oplus x_j)$
- Decryption: $c_o \leftarrow IV$, $x_j \leftarrow c_{j-1} \oplus E^{-1}{}_K(c_j)$

CBC – Properties

- Chaining causes ciphertext c_j to depend on all preceding plaintext.
  - But the same key, IV, plaintext results identical ciphertext
  - Random access to encrypted data is not possible.
  - IV must be integrity-protected
    - Otherwise, attackers may make predictable bit changes to the 1st block
- A single bit error in c_j affects decryption of blocks c_j, c_{j+1}
- Self-synchronizing
  - Error (including loss of blocks) in c_j but not in c_{j+1}, c_{j+2}
  - X_{j+2} is correctly decrypted

3) Counter Mode (CTR)



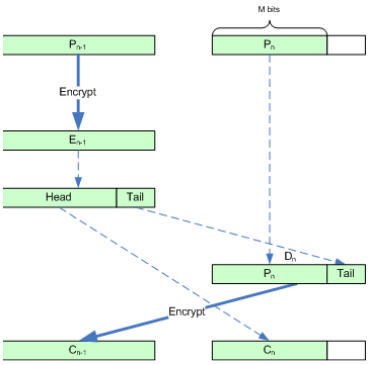
- Included with AES in 2001
- Proposed: Diffie & Hellman in 1979
- CTR must be different for each block.
- An IV/nonce value is also used with the counter for uniqueness (concat/addition/xor)
- **Software and Hardware Efficiency**, different blocks can be encrypted in parallel (multi-processor CPUs)
- **Preprocessing:** The encryption part can be done offline and when the message is known, just do the XOR
- **Random Access:** Decryption of a block can be done in a random order, very useful for hard-disk encryption.

XTS-AES

- **XTS:** XEX-based tweaked CodeBook mode (TCB) with **Cipher Text Stealing** (CTS)
- CTS provides support for sectors with size note divisible by block size: 520-byte sectors, 16-byte blocks
- XEX (xor-encrpyt-xor): Efficient processing of consecutive blocks within a data unit.
  - E.g, A Disk Sector
- Can be used with other modes

    - How last two block are processed
- Used by: TrueCrypt, dm-crypt (Linux), FileValue
- For more: en.wikipedia.org/wiki/XTS_mode

ECB-CTS



A) Substitution Cipher

- Recall that a substitution cipher is one in which the plaintext characters are replaced by other characters.
- Key is the mapping between plaintext and ciphertext.
- Example
    - **Plain:** abcdefghijklmnopqrstuvwxyz
    - **Cipher:** xkfvhijelanoqsrpcugwyzmbt

B) Transposition Cipher

- Recall: For a simple transposition cipher
    - With a fixed period t
    - Encryption involves grouping the plaintext into blocks of t characters
    - Applying to each block a single permutation e on the numbers 1 to t.
- **Transposition** is said to add diffusion (spreading out bits so redundancy in plaintext is spread over ciphertext)
- **Substitution** is said to add confusion (makes relationship between key and ciphertext as complex as possible)

C) Product Cipher

- Feistal proposed that we can approximate the ideal block cipher by using the concept of the product cipher.
    - Execution of two or more simple ciphers in sequence (round of operations), such that the final product is cryptographically stronger than its components.
    - In particular, alternating substitutions and transpositions.

Block Cipher Design Approaches

- A round function is used
    - Strong avalanche effect
    - Bijective
- Repeated multiple times (n rounds)
    - Round 1 input: plaintext block

- Round n output: ciphertext block
- Round key: derived from the key K, for each round.

- Examples
  - SP Networks (Used in AES, CAST-128)
  - Feistel Structures (used in DES, RC5, Blowfish)

Substitution-Permutation (SP) Network

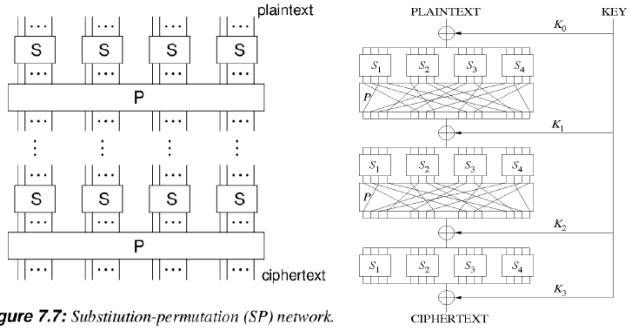


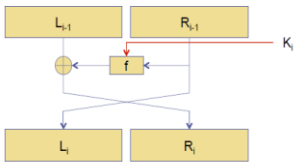
Figure 7.7: *Substitution-permutation (SP) network.*

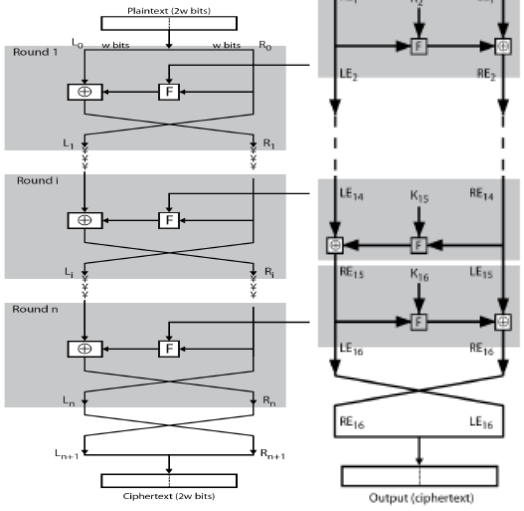
- A permutation is essentially a simple transposition
- In multi-processor CPUs: **SP is faster than Feistel**

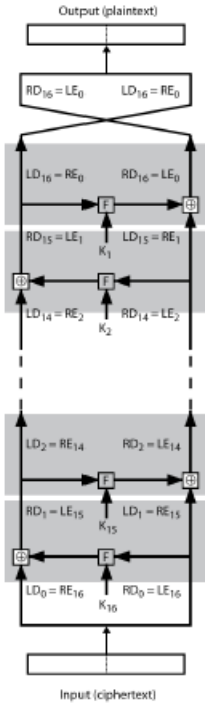
Feistel Structure

- Consists of a number of identical rounds of processing.
- **In each round, a substitution is performed on ½ of plaintext block, followed by a permutation that interchanges two halves**.
- The original key is expanded so a different key is used in each round.
- The Data Encryption Standard (DES) exhibits this structure.

- Iterated cipher mapping 2w-bit plaintext (L_0, R_0) to ciphertext (R_r, L_r) through r-round process, (L_{i-1}, R_{i-1}) $\rightarrow_{Ki}$ (L_i, R_i) as follows:
  - $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$,
  - subkey K_i is derived from cipher key K







Feistel Structure Parameters

- Block Size & Key Size
  - Larger means greater security, but may slow down encryption/decryption speed.

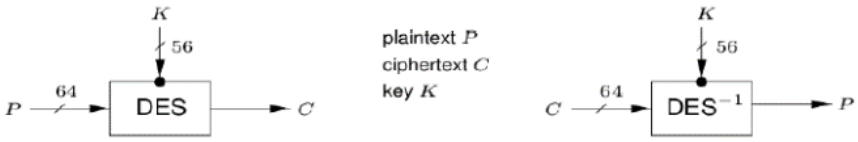
    - Block size is how much data needs to process.
- Number of Rounds:
    - More rounds means greater strength.
    - Typically **16** rounds.
- Subkey generation algorithm & round function:
    - Greater complexity should leader to greater cryptanalysis resistance.

Avalanche Effect

- Slight change in input (e.g., flipping a single bit) significantly changes the output (e.g., half the output bits are flipped)
- Strongly desired property of:
    - Block Ciphers
    - Cryptographic Hash Functions

Data Encryption Standard (DES) *Fixed Key Length*

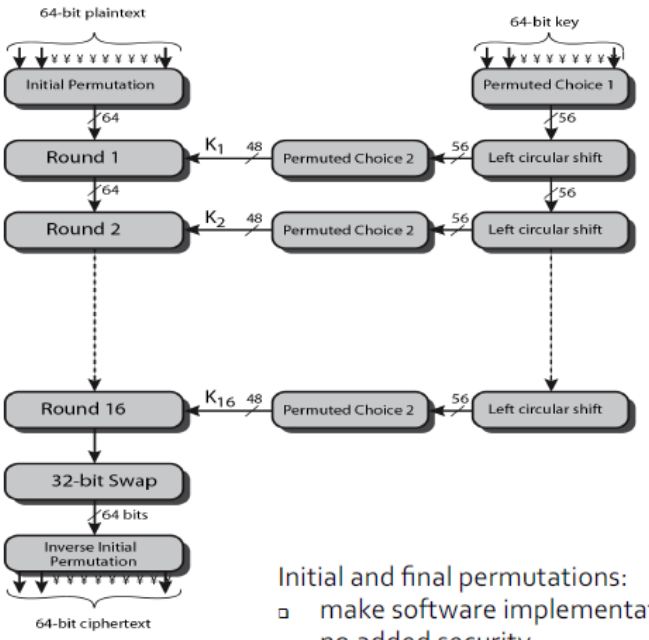
- The most widely used symmetric cipher, and most studies/analyzed.
- Uses the Feistel Structure
    - **Block length n = 64, key size k = 56**
    - **Number of rounds = 16**
- Developed by IBM, NSA, and other consultants
- Adopted in 1979 as standard



Design Controversy

- Relatively short key length at the time (now 56 bits is far too short)
    - Original IBM proposal had longer keys (128 bits)
- Design criteria for internal structure of DES, the S-boxes, were classified
- But it has withstood the tests of time, and public analysis found no major flaws.
    - No trapdoor was found.

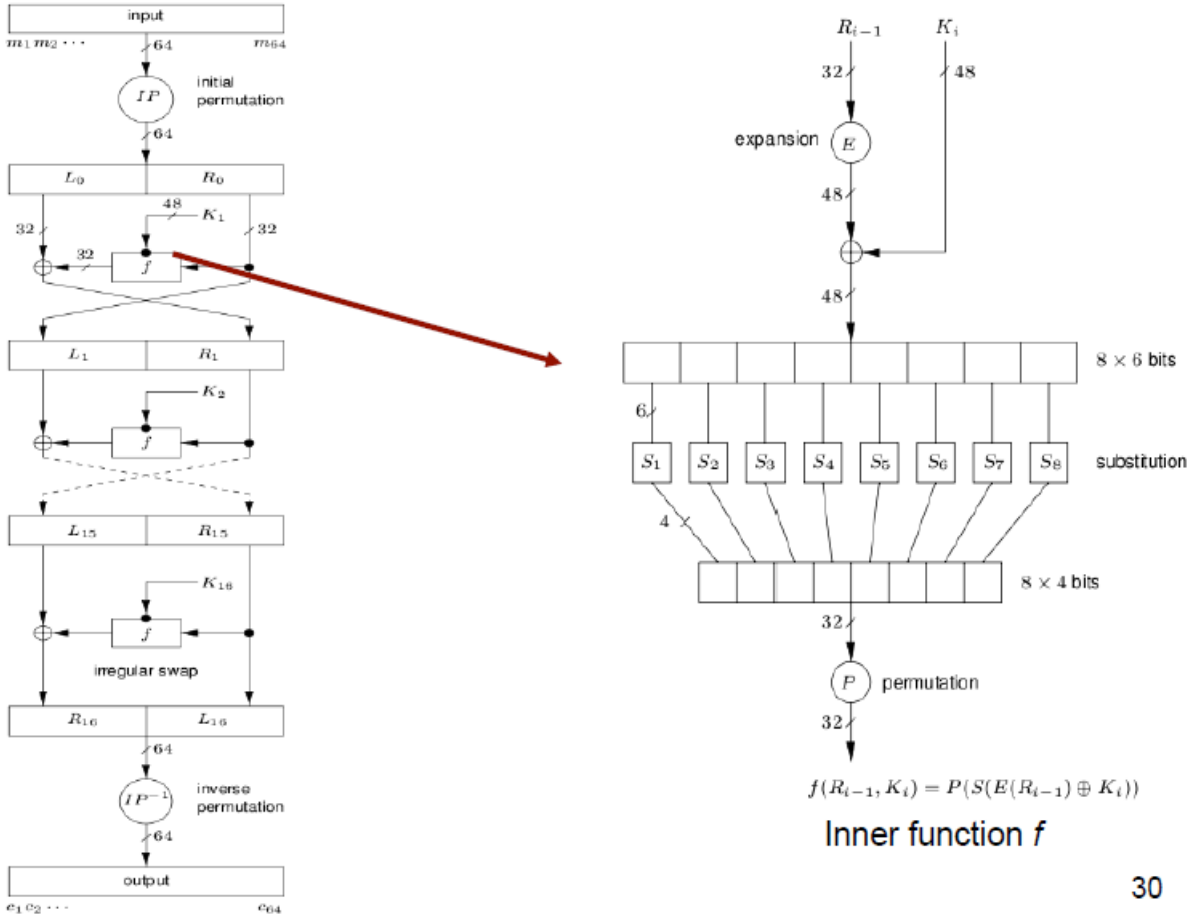
Data Encryption Standard (DES) – Overview



Initial and final permutations:

- make software implementation less efficient
- no added security

DES – Computation Path



Inner function *f*

Sample S-Box (Give a 6-Bit get a 4-Bit)

Input: 011011
Output: 1001

S_5

Outer bits \ Middle 4 bits of input	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

DES – Strength

attack method	data complexity		storage complexity	processing complexity
	known	chosen		
exhaustive precomputation	—	1	2^{56}	1 (table lookup)
exhaustive search	1	—	negligible	2^{55}
linear cryptanalysis	2^{43} (85%)	—	for texts	2^{43}
	2^{38} (10%)	—	for texts	2^{50}
differential cryptanalysis	—	2^{47}	for texts	2^{47}
	2^{55}	—	for texts	2^{55}

- Differential Cryptanalysis (1980)
- Linear Cryptanalysis (1993)
- EFF DES **Cracker** (1998)
  - EFF: Electronic Frontier Foundation – eff.org
  - Exhaustive Key Search
  - On average **4.5 days**, less than a day in 1999
  - Cost: 250,000 US$

Double/Triple Encryption

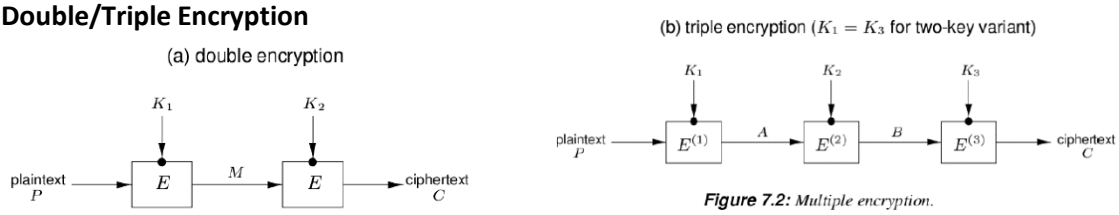


Figure 7.2: *Multiple encryption.*

Meet-In-The-Middle Attack

- For Double Encryption (2^{56} x 2^{56} = 2^{112} possibilities, this much strength can't brute force)
  - 2^{56} x 64 bits
- For Triple Encryption (2^{56} x 2^{56} x 2^{56} = 2^{168})
  - Reduces key search from 2^{2k} to 2^{k} at the expense of addition storage 2^{k}
- Requires only few plaintext-ciphertext pairs.
- Not the same as main-in-the-middle attacks.

Meet-In-The-Middle Attack (DES)

- Need to have plain text, encrypt with a key. (Don't know the key)

- $C = E_{K_2}(E_{K_1}(P))$
- $M = E_{K_1}(P) = D_{K_2}(C)$
- For Double-DES, assume we have (P,C)
  - Compute & store $E_{K_1}(P)$ for all possible K1 (2^{56} keys)
  - Compute $D_{K_2}(C)$ for all possible K2 (2^{56} keys)
  - Check the result with the previous step
  - If match occurs (K1, K2) are kept as candidate keys
  - False positives (2^{48} matches: 2^{112}/ 2^{64})
  - A second pair (P,C) reduces the false positive to 2^{-16}

Triple DES

- $C = E_{K_3}(D_{K_2}(E_{K_1}(P))$
- Two/three keys: brute-force key search costs on the order of 2^{112} operations
  - Two keys: k1 = k3
- No crypto significance of the middle decrypt operation (could be encryption as well)
  - For backward compatibility with traditional DES
    - k1=k2=k3, or k1=k2, or k2=k3

- Does not need internal storage, only one right after.
- Block size matters in this case.

Alternative to DES

- **Double-DES**, no security advantage.
- **Triple-DES,** secure, but slow.
- **Advanced Encryption Standard** (AES)
- Competition held by NIST
  - Rijndael was declared the winner in 2000
  - Developed by two cryptographers from Belgium: Joan Daemen and Vincent Rijmen

Advanced Encryption Standard

- On September 1997, NIST called for AES candidate's submission.
- In 1999, out of 15, the selection was narrowed to 5 candidates:
  - Rijndael (University, Belgium), Serpent (University, Israel + England), Twofish (Counterpane), RC6 (RSA) and MARS (IBM)
- **Rijndael versions** with a block length of 128 bits, and key lengths of 128, 192, and 256 bits are used:
  - AES-128 (n=10), AES-192 (n=12), AES-256 (n=14)
  - n = # of rounds

AES: High Level Description

- Input block 128 bits in length: consider it as a 4x4 square matrix of bytes
  - (4 x 4 = 16 bytes, 16 x 8 = 128 bits)
  - Imagine a matrix and components assigned to squares.
- Uses round keys that are the result of a key expansion. Each round key is 128 bits.

1. State = X (input), KeyExpansion (round keys are derived)
2. Initial round
   - **AddRoundKey**(State, Keyo): State ⊕ the expanded key
3. Rounds (Nr -1 times)
   - **SubBytes**(State, S-box): a **substitution step**, each byte is replaced with another according to a lookup table
   - **ShiftRows**(State): a **transposition step**, each row of the state is shifted cyclically a certain number of steps
   - **MixColumns**(State): a mixing operation, operates on the columns of the state, combining the four bytes in each column.
   - **AddRoundKey**(State, KeyR)
4. Final Round (no MixColumns)
   - **SubBytes**(State, S-box)
   - **ShiftRows**(State)
   - **AddRoundKey**(State, KeyNr)
5. Y (output) = State

AES: S-Box

- 16 x 16 matrix, one S-box (vs. DES 8 S-boxes)
- AES S-boxes construction is known
  - DES S-boxes construction is secret
- **S-boxes** based on modular arithmetic with polynomials, **can be defined algebraically**, not random.

AES – Attacks

- No practical attacks to date.
- **No attacks on full AES versions.**
- Attacks on AES-256
  - 9, 10, 11 round versions
  - But full version AES-256 involves 14 rounds. (**Safest Use The Round Numbers**)
- First key-recovery attacks on full AES (2011)
  - Faster than brute force by a factor of about **four [**2^{126} compare to 2^{128}]
- Use your own key on top of the AES, making a double even triple encryption. (Add more layers)

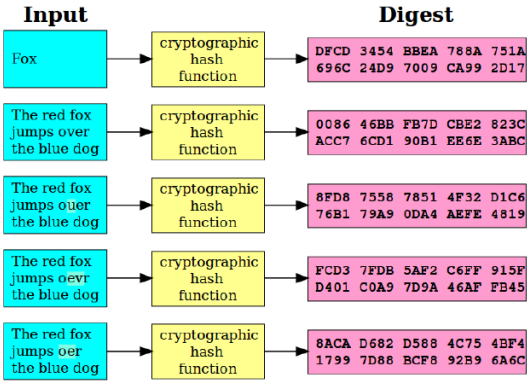
****

<u>Lecture #4: Hash</u>

Data Integrity

- How do you know that data has been altered or not?
  - E.g., How do you know when a file you downloaded has been tampered with?

Input vs. Digest



Hash Functions

- We are interested in cryptographic hash functions.
  - Also known as “message digests”
  - Different than those used in data structures.
- **Input**: A string M of arbitrary length, called the preimage.
- H(M) is a fixed-length string we call a hash-value (also called hash-code, hash-result, or hash)
- Hash-value serves as a compact, representative image of the input.
- Basic properties
  - Compressions (but unrelated to data compression)
  - Ease of computation

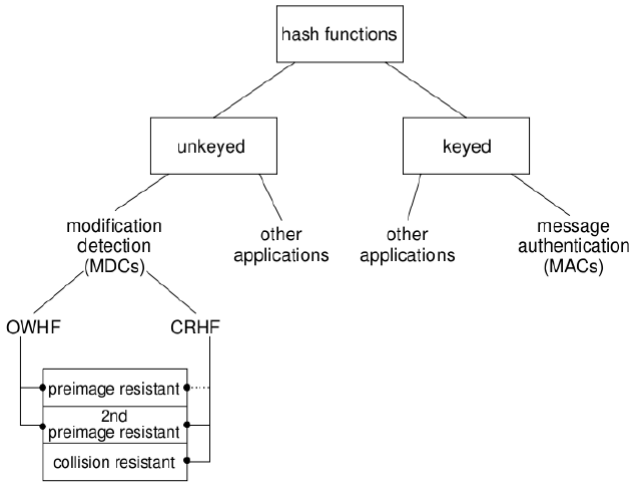
Hash Functions (Collisions)

- H(M) is a many-to-one function.
  - Say we limited M to t-bit inputs.
  - H(M) produces n-bits outputs, where t >> n
  - If H were “random”, 2^{t-n} inputs would map to each output.
  - Two random inputs would yield the same output (a collision) with probability 2^{-n} (independent of t).
- Collision should be computationally difficult to find.
  - 2^{128} probability, a small chance of collision.

Examples of Hash Functions

- **MD5**: 128-bit hash (Ron Rivest)
- **SHA-1**: 160-bit hash (NIST-NSA)
- **SHA-2**: 256/384/512-bit hash
- SHA-3: NIST announcement on October 2012
  - Open competition as for AES
  - Variable digest sizes (Implemented Now)
  - 224/256/384/512

Classification



Desirable Properties

- These are in addition to: compression & ease of computation.
- **Pre-Image Resistance:** For essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output. (Long enough not to crack)
    - Given y, to find any pre-image x' such that h(x') = y.
    - Also termed as the **one-way property**.
- **Second Pre-Image Resistance:** It is computationally infeasible to find any second input which has the same output as any **specified** input,
    - i.e., given x, to find a 2^{nd}-preimage x' ≠ x such that h(x) = h(x')
    - Expect that an attacker would need to guess 2^n inputs before finding a candidate 2^{nd}-preimage.
    - Also termed as **weak collision resistance**.
- **Collision Resistance:** It is computationally infeasible to find **any two distinct inputs** x, x' which hash to the same output, i.e., such that h(x) = h(x')
    - Note: Key difference between this and second preimage resistance is that neither x nor x' are fixed.
    - Also termed as strong collision resistance.
    - Collision resistance implies 2^{nd} preimage resistance.
- Expect an attacker would need to guess $\mathbf{2^{n/2}}$ inputs before finding a collision.
- Reason: Birthday paradox, because we are considering pairs of preimages.

Birthday Paradox

- **Question 1:** Given an arbitrary collection of people in a room, how many must be there so that the probability that **one of them shares your birthday** (same month, and day, not the year) is greater than 0.5.
    - Answer 253
    - Assumption: Birthday are equip-probable.
- **Question 2:** Given an arbitrary collection of people is a room, how many must be there so that **any two of them share a birthday** is greater than 0.5?
    - Answer 23 (Much less than the 253 above)
    - This seems unreasonably small
    - It is because we are considering pairs of people.

Applications of the Attack

- Dishonest signer
  - Signs x_1' later claims signed x_2' instead
- Dishonest verifier
  - Gets signature on x_1', later claims signature on x_2'
- Due to collision. (Sign the hash value (h(M))

More Definitions

A) **One-way hash function** (OWHF)
   - Preimage and 2nd preimage resistance

B) **Collision resistant hash function** (CRHF) ***The Most Effective**
   a. 2nd preimage and collision resistance

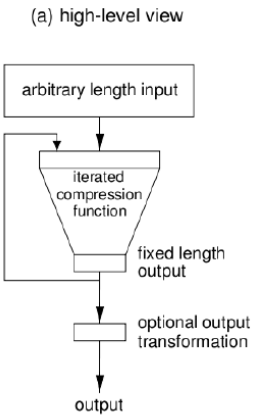
Remember: There are no known provably one-way functions.

Bitsize for Practical Security

- OWHF: n >= 80
- CRHF: n >= 160
- Assumptions: 2^{80} operations are infeasible

How to Build a Hash Function

- Hash functions using modular arithmetic.
  - **Not** recommended
- Hash functions built around block ciphers.
- Hash functions with what is termed a "dedicated" design.

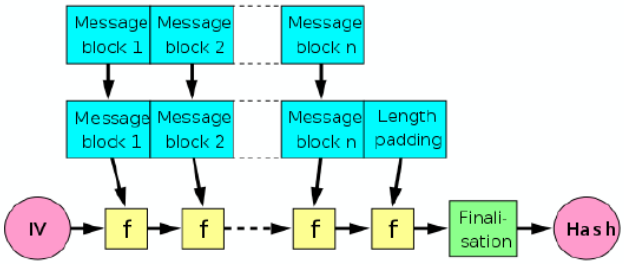


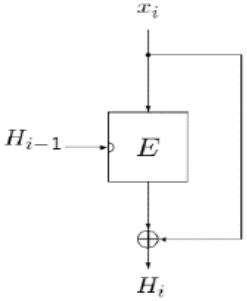
General Structure of Iterated Hash Functions (Shown to the right)

- MD5, SHA-1, SHA-256, etc. all have a similar structure.
- Known as "iterated hash functions".
- Process successive fixed-size blocks of input.

The Merkle-Damgard (MD) Construction of Hash Functions

- Goal: Construct hash function h: $\{0,1\}^* \rightarrow \{0,1\}^n$ from a compression function f: $\{0,1\}^{n+r} \rightarrow \{0,1\}^n$
  - f transforms two **fixed** length inputs to an output of the same size as one of the inputs.
  - Transformation must be one-way
- If f is collision resistant, so is h.





Hash Functions Based on Block Ciphers (Shown Above)

- Why:
  - Trust.
  - Reduce design, evaluation and implementation effort.
  - Compact implementation.
- Why Not:
  - Slow.
  - Weaknesses which are not relevant to encryption.
- Rate = # blocks hashed per encryption.

Applications of Hash Functions

- Data Integrity
  - Messages, files.
- **Hash value cannot be reversed.**
- Confirmation of knowledge, like sensitive data (E.g, passwords).
- Cryptographically secure pseudo-random bit sequences.
  - Relies on the strength of the underlying cryptographic primitives (break them, and the sequence can be predicted)

Cryptographically Secure PRBGs

- Hash functions are designed to be "**random functions**" of the input.
- Example: Linux/Dev/Random
  - Maintains entropy pool (data from random events mixed in, monitors' events in kernel).
  - To retrieve bits, pool is hashed and mixed more.
  - Keeps an estimate of how many bits added/removed – when estimate is o, /dev/random blocks, but /dev/random relies on SHA-1 to keep going.
- Still need a good source of random bits for a seed!
- Entropy pools are important (even if slower)!
- Avoid temptation to use a simple value that an attacker could guess, e.g.:
  - System Time
  - Process ID (Key = sha1(pid)) = 2^{16}

Pre-Computation Attacks

- Basic Idea: **Pre-compute all hash values** under given inputs, and only store some fraction of them in a way that optimizes future search.
- **Needs to be small to attack.**
- Also known as, **rainbow-table** (more efficient), hash-chain attacks.
  - **Hash-Chain:** Also used to mean repetitive application of hash functions.
- Example: Passwords
  - Assume all passwords use only 8 lower case characters (26^8 possible = approx.. 208 billion)
- Called a "space-time trade-off"
- Variable-length messages do not generally have this problem, since input space is so large.
- How system administrators protect against pre-computation: using "**salts**" (random value **s** prepended to **p** prior to calculating hash: H(s||p)
  - Attacker would need to build a table for each possible salt value.
- See here:

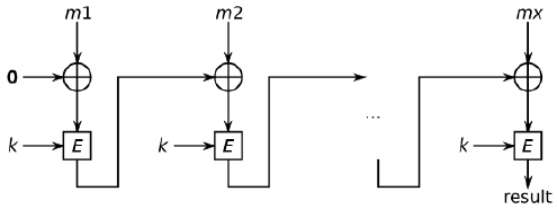
- http://en.wikipedia.org/wiki/Rainbow_table

****

Lecture #5A: Message Authentication Code (MAC)

Encryption Alone Does Not Provide Data Integrity

- Common misconception:
  - If **B** decrypts a meaningful message with shared key **k**, it must have originated from A.
- Consider
  - Blocks or packets of message can be re-ordered.
  - In some cases, plaintext does not have "meaning" or redundancy (think of a key).



Message Authentication Codes (MACs)

- Keyed hash functions are called MACs, which use both **M** and a key **k** as input.
- Goal is to add authentication to the hash-value by using a secret (symmetric) key.
- Should be difficult to create a valid MAC without knowledge of secret key.

Some Basic Uses of MACs

- Message authentication: Alice sends $M||h_k(M)$
- Message authentication and confidentiality, where authentication is tied to:
  - Plaintext: $E_{k2}(M||h_{k1}(M))$
  - Ciphertext: $E_{k2}(M)||h_{k1}(E_{k2}(M))$
- **MAC:** Does not provide **non-repudiation.**

Can We Use An MDC for a MAC?

- Not always a good idea!
- Consider:
  - h is an iterated MDC with compression function f.
  - MAC = h(k||x)
  - Attacker can compute f(MAC,y) to obtain MAC'=h(k||x||y), without knowing k.
- What about MAC = h(x||k)

Hash-Based MAC (HMAC)

- HMAC(x) = h(k||p1|h(k||p2||x)), where:
  - **k** is the **K** padded with zeroes on left so it has length of 1 block for compression function.
  - **p1** and **p2** are distinct strings of length b (repeated byte).
  - Two calls to h: inefficient?

HMAC

- H can be any cryptographic hash function.
- Output size depends on H.
- Commonly used H: MD5, SHA-1
  - HMAC-MD5 secure or not?

MAC from Block Ciphers

- CBC MAC
    - Problem if
        - Weak block cipher is used (e.g. DES)
        - Message length is variable

$PBKDF_2$ (Password-Based Key Derivation Function)

- RSA PKCS standard (also IETF RFC 2898)
- How to derive a key from a password that is resistant to password cracking?
    - Not complete resistance.
    - Goal is to increase costs of such attacks.
- Key derivation
    - DK = PBKDF2 (PRF, P, S, c dkLen)
        - Where PRF = hash or MAC function
        - P = Password
        - S = Salt Value
        - c = number of iterations
        - dkLen = required key length

PBKDF2 (Uses/Attacks)

- Used in: WPA/WPA2, WinZip, TrueCrypt, FileVault, Blackberry/Android ICS/iOS
- Parameter c is very important.
- For WPA
    - DK = PBKDF2 (HMAC –SHA1, password, ssid, 4096, 256)
- Several real-world attacks
    - http://blog.crackpassword.com/

****

Lecture #5B: Public Key Cryptography and SSL

****

Midterm Main Focus: Starting At Lecture #5B (The Latter Half Has More Emphasis)

-->Last Lecture About "Online Password Guessing" **Not Covered**

--> Some Tutorial Notes Questions (Just Read Document and Understand Once)

--> More concrete explanations of what your position is and defend yourself.

* No Calculators on Exam, no Math :D

Good Luck

****

Public Key Cryptography

- Probably the most significant advance in the history of cryptography (~3000 years).
- **Basic Insight**: In symmetric key systems, you have to share a secret key that nobody else knows.
    - What is you could "split" this key into two pieces, one that you showed to everyone, one that you kept to yourself?
- Also known as **asymmetric cryptography** (**Different Keys)**
    - vs. symmetric where the keys are the same.
- Complements symmetric cryptography (does not replace as each have advantages/disadvantages)
- Uses number theoretic concepts

    - Factoring, quadratic residue, RSA/DH problems.
- First publicly revealed by Whitfield Diffie and Martin Hellman (Stanford University) in 1976.
    - Classified community knew earlier (but only public knowledge once a 1970 report by UK CESG was declassified)
- **Uses two different keys**: **one public (K_e)**, the **other private** (**K_d**).
    - Alice generates K_d, only know to her.
    - Alice generates K_e, which she publishes (available to both Bob and Oscar).
    - Knowledge of K_e does not reveal K_d (or even give an advantage to Oscar trying to find K_d).
    - Alternative notation for Alice's keys: K_A, K_A^{-1}
- Two basic applications: encryption and signatures.
    - **Encryption**: Let anyone send you a message that only you can read.
    - **Signatures**: Let anyone verify that a message is from you.

Trapdoor One-Way Functions

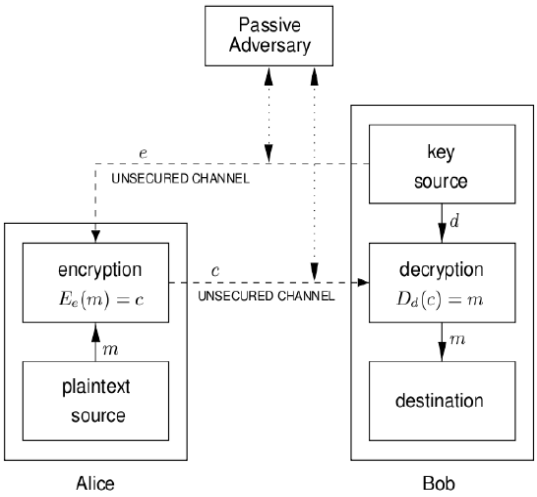
- A function f: {0,1}* -> {0,1}* is a trapdoor one-way function if and only if f(x) is a one-way function; however, given some extra information it becomes feasible to compute f^{-1}: given y, find x such that y = f(x).
- Public key cryptography relies on trapdoor one-way functions.

Symmetric vs. Public Key Characteristics

- Keys in **symmetric cryptosystem**.
    1) Random bit strings.
    2) No special mathematical properties.
    3) Generation is relatively "easy"
- Keys in **asymmetric cryptosystem**.
    1) Have special mathematical properties (e.g., large primes)
    2) Generation: Computationally expensive.
- **Key sizes are NOT comparable**.

Public Key Encryption

- **Split Key In Half One Private For You Other Public For Someone Else**
- **Alice** wants to send a message M to **Bob**. She knows his public key $K_{e,Bob}$, since it is public.
- Allice encrypts M using $K_{e,Bob}$, producing C.
- Bob decrypts C using $K_{d,bob}$, a private key generated by Bob that corresponds to $K_{e,Bob}$
    - Only $K_{d,Bob}$ can decrypt a message encrypted with $K_{e,Bob.}$
    - Notice this means that even Alice cannot decrypt C.



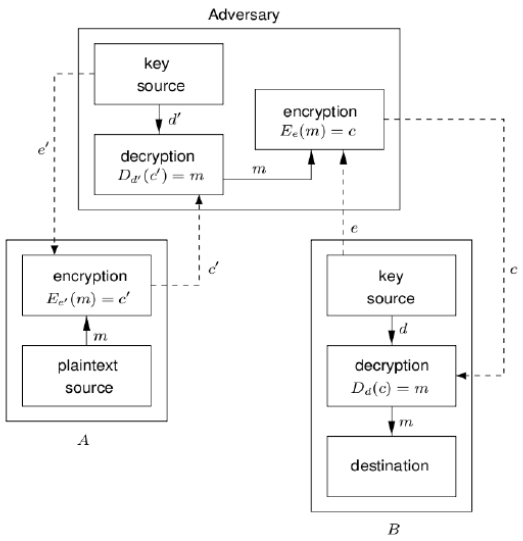
Public-Private Key Pairs

- Should be infeasible to find decryption key given encryption key and algorithm.
- Public key schemes utilise problems that are easy (P type) in one way but hard (NP type) in the other way, eg:

- Exponentiation vs. logs.
- Multiplication vs. factoring.

Public Key Encryption: MITM (Shown Right)

- What if the network attack substitute different public key between two users?
  - Main-in-the-middle attack! (MITM)
- **All public key systems are vulnerable to MITM. Must use authentic public keys!**



Public Key Signatures

- Signatures are for integrity and authentication of a message.
- Alice wants to sign a message M for Bob. She knows her own private key $K_{d,Alice}$, generated by her.
- Alice sing M using $K_{d,Alice}$, producing Alice{M}
- Note that Alice could alternately sign H(M).
- Bob verifies Alice{M} using $K_{e,Alice}$, a public key published by Alice that corresponds to $K_{d,Alice}$
  - Only $K_{d,Alice}$ can sign a message verifiable with $K_{e,Alice}$
  - Notice that anyone can verify Alice's signature with her public key, but no-one else can forge a signature that can be verified with her key.

Public Key Cryptosystems (Typical Usage Mode)

- Typically, the entire message is **not** encrypted with a public key, or singed with a private key.
  - Reason: Computational Expense
- Typical usage for encryption: Encrypt a symmetric session key using the public key, then rest of message encrypted with session key.
- Typical usage for signatures: create a hash H(M) of message M, then sign H(M) using private key.

Advantages of Public Key Crypto

- Only private key must be secret (although authenticity of public keys must be guaranteed).
- In large networks, the number of keys required is considerably smaller than if symmetric keys used.
  - Consider that with symmetric, every pair of 2 people need a different key.
  - For n users, we need: 1 + 2 + … + (n-1) = n(n-1)/2 keys = $O(n^2)$
- In a public key system, we need: 2n keys = O(n)

Disadvantages of Public Key Crypto

- Encryption much slower than symmetric ciphers.
- Key sizes are typically much larger.
- No public key cipher prove to be secure (although the same can be said for block ciphers)

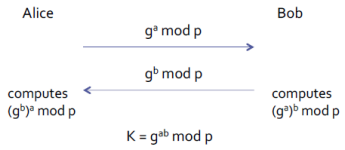
Diffie-Hellman Key Exchange

- Referred as "D-H Exchange"
  - W.Diffie and M.Hellman (1976)
  - Enables two parties to jointly **establish a shared secret key**
    - No prior shared secret
  - Weakness: No authentication
- Based on exponentiation in a finite field (modulo a prime)
  - Easy for legitimate parties to compute.
- Security relies on the difficulty of computing discrete logarithms (similar to factoring)
  - Hard for adversaries to compute.

D-H Key Exchange

- Prior to the key exchange, Alice and Bob agree on a set of parameters:
  - p, a large prime integer.
  - g, a primitive root mod p (also called a generator)
    - Meaning a number whose powers successfully generate all elements mod p.
    - For every number x between 1 and p-1 there is a power k of g such that $x = g^k \bmod p$.
  - p,g: both are public, can be used by all system users.

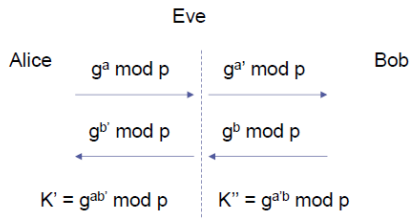
- Session key is then obtained by both parties calculating:
  - $K_{AB} = g^{ab} \bmod p$
- Alice can compute this by:
  - $y_B{}^a \bmod p$
- Bob can compute this by:
  - $y_A{}^b \bmod p$
- Oscar needs either a or b to compute this key!
  - $y_B \cdot y_A = g^{a+b} \bmod p$



- To obtain session key, Oscar would need to solve the discrete logarithm problem for either:
  - $g^a \bmod p$
  - $g^b \bmod p$
  - Notice how if logarithm of either of these found, the session key is known.
- Oscar could also try to figure out the session key based on the public key.
  - No clear way to do this.
- What else could Oscar do to attack the session key?

DH – Main in the Middle Attack

- Attack can intercept modify, insert, delete messages on the network.
- Even can translate messages between Alice & Bob without being noticed.
- Similar attacks possible on RSA & other PKC protocols.



Security of DH

- Discrete Logarithm Problem (DLP)
  - Given p, g, g^a mod p
  - What is a?
- DH problem (DHP)
  - Given p, g, g^a mod p
  - What is g^{ab} mod p?
- Conjecture: DHP is as hard as DLP.

A) **PAKE – Password Authenticated Key Exchange**

- **PAKE** Definition:
  - Two parties share only a "weak"/low-entropy secret, e.g., a password.
  - Establish a secure channel between the users using **only** the shared secret + mutual authentication.
  - The channel must be protected by a high-entropy session key.
  - Offline guessing attacks on the weak secret must be infeasible.
  - Prevention of online guessing is a non-goal.
- PAKE protocols, require no PKIs

PAKE – A "Catch 22"

- But how to derive a high-entropy secret from a low-entropy one?
  - Is it even possible?
- Simple Key Transport
  - Assume two parties A, B share a password P
  - Protocol:
    1) A generates a random number R (e.g., 256 bits long)
    2) A -> B : $\{R\}_p$
    3) A <- B: $\{plaintext\}_R$
- How to launch a dictionary attack on P?
- **Once you have a week key, it is very hard almost impossible to make a strong key.**

B) **Encrypted Key Exchange (EKE)**

- **Start from a weak key, and end up with a strong key.**

DH-EKE – Protocol

- PWD: shared password

1. A → B: A, g, p, $\{g^x \bmod p\}_{PWD}$
2. B → A: B, $\{g^y \bmod p\}_{PWD}$
3. A → B: $\{C_A\}_K$
4. B → A: $\{C_A, C_B\}_K$
5. A → B: $\{C_B\}_K$

- K = g^{xy} mod p; x, y: random values for each session
- Achieves:
  - Mutual authentication
  - Strong session key
  - No offline attacks on PWD
  - Key confirmation

C) RSA Encryption & Signatures

- Discovered in 1977, first method for encryption and signatures.
- Security based on intractability of the **integer factorization problem**.
- Most widely used public key cryptosystem.
- Based on exponentiation in finite field over integers modulo a prime.
  - Exponentiation is easy for those with key.
- Uses large integers for key size
  - E.g., 1024, **2048**, 4096 (1024 no more)
  - 1024-bit RSA is claimed to be equivalent of 80-bit symmetric key.
  - Note: Similar key sizes for: DH, ElGamal, RSA
- Security due to cost of factoring large numbers.

RSA Key Generation

- Performed rarely, when an entity (e.g., Alice) sets up a public/private key pair (e,d,n).
  - n is the modulus.
  - e is the encryption exponent.
  - d is the decryption exponent.

Algorithm 8.1 (HAC) -- steps that Alice must follow:

1. Generate 2 large random (and distinct) primes *p* and *q*, each roughly the same size.
2. Compute n = pq and $\phi(n) = (p-1)(q-1)$.
   - Note that $\phi(n)$ (pronounced "phi") means the number of values relatively prime to n.
     - $\phi(n)$ is called Euler's phi function
     - If 2 numbers are relatively prime, they have no common divisors > 1.
3. Select a random integer *e*: $1 < e < \phi(n)$
   - **gcd(e, ϕ(n)) = 1**
   - Often e is a low number for encryption efficiency.
4. Use extended Euclidean algorithm (Algorithm 2.107, HAC) to compute the unique integer d, $1 < d < \phi(n)$, such that **ed ≡ 1 (mod ϕ(n))**.
5. Alice's public key is (e,n)
   - private key is d

- n is used as a modulus in the encryption and decryption operations.
- It is extremely important that the factors p & q of the modulus n are kept secret, since if they become known, the system can be broken.
  - This is how the system relies on integer factorization for security.
- Note that different users will have different moduli n.

RSA Key Generation Consideration

- Primes p and q must not be easily derived from modulus n = pq
  - Otherwise, it is easy to know $\phi(n) = (p-1)(q-1)$
  - From $\phi(n)$ and e, attacker can use Extended Euclidean Algorithm to find d.

- To ensure p and q not easily derivable from n, the primes must be sufficiently large.
  - Normally, this means guessing a large number and using probabilistic tests to determine if prime.

RSA Encryption

- We call e (of the public key) the encryption exponent, and d (of the private key) the decryption exponent.
- Let's say that Bob wants to send a message to Alice.
- Bob must first obtain Alice's public key (e,n).
- Bob does the following:
  1) First obtain Alice's authentic public key (e,n)
  2) Take the message M and represent it as an integer m in the interval [0,n-1].
     a. If the message cannot be represented in [0,n-1] it must be separated into blocks.
  3) Compute $c = m^e \bmod n$
  4) Send c to Alice.
- To decrypt c, Alice does the following:
  1) Compute $m = c^d \bmod n$

How decryption works:

- Since $ed \equiv 1 \pmod{\phi(n)}$, there exists an integer k
  - $ed = 1 + k.\phi(n)$
- $c^d = m^{e.d} = m^{1 + k.\phi(n)}$

$= m^1 . (m^{\phi(n)})^k$

$= m^1 . (1)^k$

- Note this is because of Euler's theorem: $(m^{\phi(n)} \bmod n = 1$ where $\gcd(m,n) = 1.)$

$= m^1$

$= m$

****

Digital Signatures

- **Digital Signature:** A data string which associates a message with some originating entity.
- **Digital Signature Scheme:** Secret signing key and a public verification key.
- **Service Provided:**
  - Authentication
  - Data Integrity
  - Non-Repudiation (MAC does not provide this).
- It is normally advised that the signing key-pair is different than the key-pair used for encryption.
- **With Appendix:** Require message as input to verification algorithm.
  - Normally a cryptographic hash of the message is signed.
  - Less susceptible to existential forgery attacks.
  - Examples: EIGamal, DSA
- **With Message Recovery:** A priori knowledge of the message is not required for verification.

- This message signed can be recovered from the signature itself.
- RSA, Rabin

RSA Signatures

- Recall in RSA encryption, the public key is (e,n), private key is d.
- For RSA signatures, the key-pair is generated the same way as for RSA encryption.
- Note that RSA signatures are deterministic and provide message recovery.
- Can use a publicly known redundancy function R, and inverse R^{-1}. R maps message strings to RSA inputs.
- If Alice wants to sign a message m, she:
  - Computes m' = R(m)
  - Computes $s = m'^{d} \bmod n$
  - Alice's signature for m is s.
- If Bob wants to verify Alice's signature s, he:
  - Obtains Alice public key (e,n).
  - Computes $m' = s^{e} \bmod n$.
  - Verifies that m' is in the redundancy space.
  - Recovers $m = R^{-1}(m')$
- Note that Alice is effectively "encrypting" using her private key and Bob "decrypting" using Alice's public key.
  - Reverse of usual encryption, where Bob encrypts using Alice's public key, and Alice decrypts using her private key.
  - No confidentiality: anyone with Alice's public key can recover the message.
  - Does provide authentication and integrity guarantees: Only the holder of Alice's private key can create a valid message that has a signature verifiable with her public key.

RSA Signatures with Appendix

- RSA signatures can be modified to be a scheme with appendix as follows:
  - Alice calculates hash of m: h(m)
  - Alice signs h(m) instead of m.
  - Alice transmits (m,s).
  - Bob can verify message by calculating h(m) and comparing h (m) to the verified signature.
- This is what PGP (Pretty Good Privacy) uses to create signatures on email (using SHA-1).

Establishing Authenticity

- Digital signatures provide integrity, authentication, and non-repudiation, but we still have a problem.
  - How do we obtain someone's public key over a network, in a way that we can trust it belongs to the claimed entity?

Certificates

- **Certificate:** Set of public keys and identification information.
- Typically signed by some other source, signatures included in certificate.
  - Signatures provide integrity and authenticity guarantees.
  - Source is normally a **trusted third party** (also called: certificate authority (CA)).
  - Root certificates are self-signed.

- **Example:** x.509 certificates, PGP ("web of trust", no trusted third parties)
- Recall MITM attacks, where the recipient (Bob) has been fooled into thinking a message came from the claimed sender (Alice).
    - If a trusted third party signs Alice's certificate, Bob has some extra assurance that the certificate he received is actually Alice's, and the signed messages verifiable with the certificate are truly from Alice.
- Notice how we are moving the authentication problem of each certificate to a trusted third party.
- Is this very different from what we do with e.g., passports.

Self-Signed Certificates

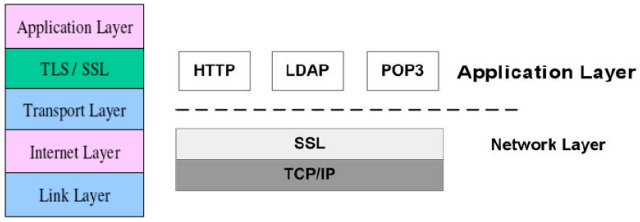
- If not self-singed, someone who sings the key can first modify the key or other information.
    - But then again, they could also just forge the entire certificate. But if entire certificate is forged, there would potentially be different certificates in the "web of trust" (PGP) for the same user, should be a warning sign.
- Also, it can provide assurance that the owner of the certificate actually has the private key.

SSL/TLS (Design and Recent Attacks)

Enforced Security Properties

- Confidentiality
- Message Authenticity
- Authentication
    - Server Authentication
    - Client Authentication
- SSL/TLS do not enforce
    - Non-Repudiation
    - Availability

SSL Architecture



SSL / TLS run beneath application protocols and just above TCP

Notation

Ver_i	SSL version number of party i
$Suite_i$	Cryptographic preferences of party i
N_i	Random nonce generated by party i
$Secret_i$	Random secret generated by party i
K_i^+	Public encryption key of party i
V_i	Public verification key of party i
$\text{sign}_i\{\ldots\}$	Signed by party i
$\{\ldots\}_{K_i^+}$	Encrypted by public key K_i^+
$Messages$	All messages up to this point
$\langle I \rangle$	Message is intercepted by the intruder

Protocol A

$C \rightarrow S$	$C,\ Ver_C,\ Suite_C$
$S \rightarrow C$	$Ver_S,\ Suite_S,\ K_S^+$
$C \rightarrow S$	$\{Secret_C\}_{K_S^+}$

⟨Change to negotiated cipher⟩

Attacks on Protocol A

- The intruder can insert his own key in the server's Hello Message.

$C \to S$ $\quad C,\ Ver_C,\ Suite_C$

$S \to C\langle I\rangle$ $\quad Ver_S,\ Suite_S,\ K_S^+$

$I \to C$ $\quad Ver_S,\ Suite_S,\ \underline{K_I^+}$

$C \to S\langle I\rangle$ $\quad \{Secret_C\}_{K_I^+}$

$I \to S$ $\quad \{Secret_C\}_{K_S^+}$

⟨Change to negotiated cipher⟩

Protocol B/C: Add Client Authentication

$C \to S$ $\quad C,\ Ver_C,\ Suite_C$

$S \to C$ $\quad Ver_S,\ Suite_S,\ \mathrm{sign}_{CA}\{S, K_S^+\}$

$C \to S$ $\quad \mathrm{sign}_{CA}\{C, V_C\},\ \{Secret_C\}_{K_S^+},$
$\mathrm{sign}_C\{\mathrm{Hash}\,(Secret_C)\}$

⟨Change to negotiated cipher⟩

Attack on Protocol C

Even though the intruder can modify neither keys nor shared secret in Protocol *C*, it is able to attack the plaintext information transmitted in the *Hello* messages. This includes the parties' version numbers and cryptographic preferences.

$C \to S\langle I\rangle$ $\quad C,\ Ver_C,\ Suite_C$

$I \to S$ $\quad C,\ \underline{Ver_I},\ \underline{Suite_I}$

$S \to C\langle I\rangle$ $\quad Ver_I,\ Suite_S,\ \mathrm{sign}_{CA}\{S, K_S^+\}$

$I \to C$ $\quad Ver_I,\ \underline{Suite_I},\ \mathrm{sign}_{CA}\{S, K_S^+\}$

$C \to S$ $\quad \mathrm{sign}_{CA}\{C, V_C\},\ \{Secret_C\}_{K_S^+},$
$\mathrm{sign}_C\{\mathrm{Hash}\,(Secret_C)\}$

⟨Change to negotiated cipher⟩

Attack on Protocol E

I observes a run of the protocol and records all of C's messages. Some time later, I initiates a new run of the protocol, ostensibly from C to S, and replays recorded C's messages in response to messages from S. Even though I is unable to read the recorded messages, it manages to convince S that the latter is talking to C, even though C did not initiate the protocol.

Protocol E: Add Post-Handshake Verification of All Messages

$C \to S$ $\quad C,\ Ver_C,\ Suite_C$

$S \to C$ $\quad Ver_S,\ Suite_S,\ \mathrm{sign}_{CA}\{S, K_S^+\}$

$C \to S$ $\quad \mathrm{sign}_{CA}\{C, V_C\},\ \{Secret_C\}_{K_S^+},$
$\mathrm{sign}_C\{\mathrm{Hash}\,(Secret_C)\}$

⟨Change to negotiated cipher⟩

$S \to C$ $\quad \{\mathrm{Hash}\,(Messages)\}_{\mathrm{Master}(Secret_C)}$

$C \to S$ $\quad \{\mathrm{Hash}\,(Messages)\}_{\mathrm{Master}(Secret_C)}$

$C \to S$ $\quad C,\ Ver_C,\ Suite_C$

$S \to C$ $\quad Ver_S,\ Suite_S,\ \mathrm{sign}_{CA}\{S, K_S^+\}$

$C \to S$ $\quad \mathrm{sign}_{CA}\{C, V_C\},\ \{Secret_C\}_{K_S^+},$
$\mathrm{sign}_C\{\mathrm{Hash}\,(Secret_C)\}$

⟨Change to negotiated cipher⟩

$S \to C$ $\quad \{\mathrm{Hash}\,(Messages)\}_{\mathrm{Master}(Secret_C)}$

$C \to S$ $\quad \{\mathrm{Hash}\,(Messages)\}_{\mathrm{Master}(Secret_C)}$

Next run of the protocol ...

$I \to S$ $\quad C,\ Ver_C,\ Suite_C$

$S \to C\langle I\rangle$ $\quad Ver_S,\ Suite_S,\ \mathrm{sign}_{CA}\{S, K_S^+\}$

$I \to S$ $\quad \mathrm{sign}_{CA}\{C, V_C\},\ \{Secret_C\}_{K_S^+},$
$\mathrm{sign}_C\{\mathrm{Hash}\,(Secret_C)\}$

⟨Change to negotiated cipher⟩

$S \to C\langle I\rangle$ $\quad \{\mathrm{Hash}\,(Messages)\}_{\mathrm{Master}(Secret_C)}$

$I \to S$ $\quad \{\mathrm{Hash}\,(Messages)\}_{\mathrm{Master}(Secret_C)}$

Protocol F: Add Nonces

$C \rightarrow S$ $\quad C,\ Ver_C,\ Suite_C,\ N_C$

$S \rightarrow C$ $\quad Ver_S,\ Suite_S,\ N_S,\ \text{sign}_{CA}\{S, K_S^+\}$

$C \rightarrow S$ $\quad \text{sign}_{CA}\{C, V_C\},\ \{Secret_C\}_{K_S^+},$
$\text{sign}_C\{\text{Hash}\,(Secret_C)\}$

⟨Change to negotiated cipher⟩

$S \rightarrow C$ $\quad \{\text{Hash}\,(Messages)\}_{\text{Master}(Secret_C)}$

$C \rightarrow S$ $\quad \{\text{Hash}\,(Messages)\}_{\text{Master}(Secret_C)}$

Export Control

- The 512-bit export grade encryption was a compromise between "dumb and dumber".
- In theory it was designed to ensure that the NSA would have the ability to 'access' communications, but still 'good enough' for commercial use.
- US servers needed to support both strong and weak crypto.
- SSL designers used a 'cipher suite' negotiation mechanism to identify the best cipher both parties could support.
- Export control is gone, but EXPORT cipher suite continues.

Why OpenSSL was not changed?

- Most modern clients won't offer export grade ciphersuites. In theory this means that even if the server support export-grade crypto, your session will use strong crypto.
- Almost no servers, it was believed, even offer export-grade ciphersuites anymore.
- Even if you do accidentally negotiate an export-grade RSA ciphersuite, a meaningful attack still requires the attacker to factor a 512-bit RSA key, for every single connection.

The MITM Attack

1) In the client's Hello message, it asks for a standard 'RSA' ciphersuite.
2) The MITM attacker changes this message to ask for 'export RSA'.
3) The server responds with a 512-bit export RSA key, signed with its long-term key.
4) The client accepts this weak key due to the OpenSSL/SecureTransport bug.
5) The attacker factors the RSA modulus to recover the corresponding RSA decryption key.
6) When the client encrypts the 'pre-master secret' to the server, the attacker can now decrypt it to recover the TLS 'master secret'.
7) From here on out, the attacker sees plaintext and can inject anything it wants.

More From Number Theory

- Z denotes all integers
  - Z = {.... -3, -2, -1, 0, 1, 2, 3, ...}
- Z_n denotes all integers mod n
  - Z_n = {0, 1, 2, 3, ... n-1}
  - Add, sub, multiplication in Z_n done mod n
  - Z_9 = {0, 1, 2, 3, ... 8}
    - What is (3 * 4) in Z_9?
- The multiplicative inverse of a (mod n) is an interger x, such that:
  - $ax \equiv 1 \pmod n$
  - x is denoted as a^{-1}
  - Note: both a, $x \in Z_n$
  - a is invertible if gcd(a, n) = 1
  - $4^{-1} = 7$ in Z_9

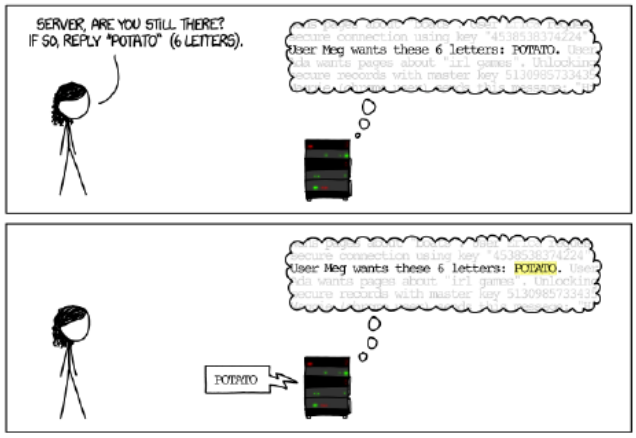
Groups

- A group $(G, *)$ is a set G on which a binary operation $*$ is defined which satisfies the following axioms:
  - Closure: For all $a, b \in G$, $a * b \in G$
  - Associative: For all $a, b, c \in G$, $(a * b) * c = a * (b * c)$
  - Identity: $\exists\ e \in G$ s.t. for all $a \in G$, $a * e = a = e * a$
  - Inverse: For all $a \in G$, $\exists\ a^{-1} \in G$ such that
    - $a * a^{-1} = a^{-1} * a = e$
- Example: $(Z_7, +)$
  - elements: {0,1,2,3,4,5,6}
  - 5+4=2
  - 0 is identity element
  - every element x has an inverse y such that $x+y=0$,
    - what is the inverse of 3?
    - 3+4=0, thus the inverse of 3 is 4.
- Example: $(Z_7, \times)$
  - elements: (1,2,3,4,5,6)
  - what is the identity element?
  - what is the inverse of 3?
    - 3×1=3, 3×2=6, 3×3=2, 3×4=5, 3×5=1, 3×6=4
- A group G is finite if |G| is finite
  - Order of G = number of elements in G
- G is cyclic, if there is an element $\alpha \in G$ such that for each $b \in G$, there is an integer i with:
  - $b = \alpha^i$
  - α is called the generator of G

Heartbeat Measurement

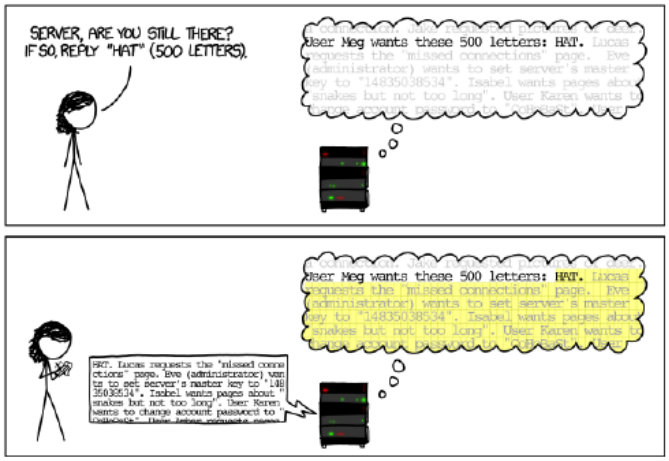
- 44 out of top 100 Alexa sites were vulnerable.
- Estimated 24%-55% HTTPS servers in total.
- Also affected:
  - SMTP + TLS
  - Tor
  - Embedded Devices
- **No Evidence of Exploitation**

Heartbeat (XKCD/1354)



Heartbleed

- No length checking so you could exploit any value you like!



****

Usability and Human Factors in Security

- "Only amateurs attack machines; professionals target people." – Bruce Schneier

Outline

- A usability analysis and user study of PGP. (1999)
- Compromising computers/data by exploiting human behaviors.
- A user study on phishing attacks. (2006)
- Ah high-impact attack based on phishing. (2011)

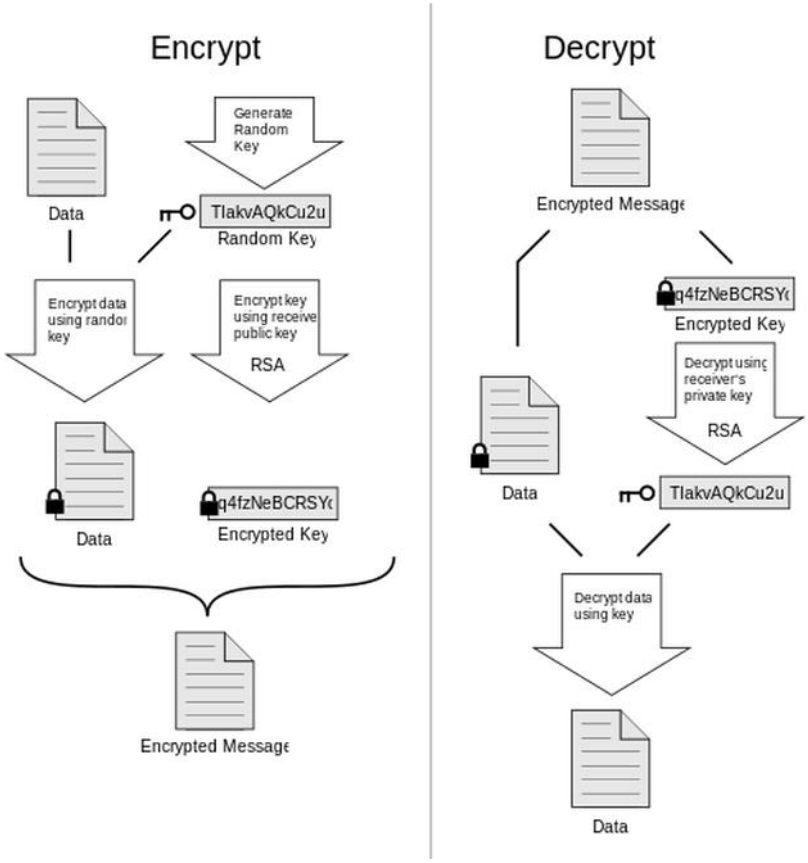
PGP Steps

1) Install PGP/GPG/OpenPGP.
2) Generate/import key pairs.
3) Protect your private keys.
4) Integrate/enable in your mail client.
5) Publish your public key.
6) Collect public keys of your contacts.
    - http://www.pgpi.org/doc/pginto/

***AMAZING LINK SUMMARY OF THE WHOLE COURSE!!!!!

- http://www.pgpi.org/doc/pginto/

PGP Encryption/Decryption



Human Factors in Security

- Many security errors are can be attributed to users.
- Nicer UI is not the solution.
- Applicable to *ALL* users.
- Core Issues:
    - Mental model / semantic model.
    - Motivation.
    - Psychology.

"Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0"

- https://www.cs.berkeley.edu/~tygar/papers/Why_Johnny_Cant_Encrypt/OReilly.pdf

Hypothesis and Conclusions

- Hypothesis: "…effective security requires a different usability standard, and that it will not be achieved through the user interface design techniques appropriate to other types of consumer software."
- Conclusions: "Making security usable will require the development of domain-specific user interface design principles and techniques"
- "User interface design for effective security remains an open problem"

Usability for Security: Definition

- Security software is usable if the people who are expected to use it:
  1) Are reliably made aware of the security tasks they need to perform.

2) Are able to figure out how to successfully perform those tasks.
3) Don't make **dangerous errors.**
4) Are sufficiently comfortable with the interface to continue using it.

Problematic Properties of Security

1. **The Unmotivated User Property**
    a. Secondary Goal
    b. Optimistic View ("security is working", "I've got nothing to hide")
    c. Get help from psychology studies (re: user capabilities, how to motivate people)
2. The Abstraction Property
    a. Policies are abstract.
    b. Policies vs. implementation.
3. The Lack of Feedback Property.
    a. How to design effective warning dialogs.
    b. How to communicate system state.
    c. Too much feedback can also be bad.
4. **The Barn Door Property**
    a. "Once a secret has been left accidentally unprotected, even for a short time, there is no way to be sure that it has not already been read by an attacker."
5. The Weakest Link Property
    a. A single user error is enough to defeat a system.
    b. Users must follow ALL security guidelines.

Expected Usability Standard for PGP

1. Understand that privacy is achieved by encryption, and figure out how to **encrypt/decrypt email**.
2. Understand the **authentication** is achieved through digital signatures, and figure out how to sign email and how to verify signatures on email from other people.
3. Understand that in order to sign email and allow other people to send them encrypted email a **key pair must be generated**, and figure out how to do so.
4. Understand that in order to allow other people to verify their signature and to send them encrypted email, they **must publish their public key**, and figure out some way to do so.
5. Understand that in order to verify signatures on email from other people and send encrypted email to other people, they must **acquire those people's public keys**, and figure out some way to do so.
6. Manage to avoid such **dangerous errors** as accidentally failing to encrypt, trusting the wrong public keys, failing to back up their private keys, and forgetting their pass phrases.
7. Be able to succeed at all of that above within **a few hours** of reasonably motivated effort.

PGP's Usability Evaluation Methods

- Cognitive Walkthrough
    - Simulated use by a domain expert.
    - Step through the use of the software as if they were novice users, attempting to mentally simulate what they think the novices' understanding of the software would be at each point, and looking for probable errors and areas of confusion.
    - Learnability evaluation.
- User study
    - Test with real users.

Cognitive Walkthrough: Findings

- Visual metaphors for security primitives.
  - So far from anything else in real-world.
- Different Key Types
  - RSA vs. DH/DSS
- Trusting Key Server
  - Web of Trust.
  - The role of a key server, trusted? Semi-trusted?
- Key Management Policy
  - Validity of public key to person mapping.
  - Trust in a given public key (i.e., owner of the key) as a certifier of other public keys.
  - Validity rating of a key is set by policy (non-obvious to users)
- Irreversible Actions
  - Accidentally deleting the private key.
  - Accidentally publicizing a key.
  - Accidentally revoking a key.
  - Forgetting the passphrase.
  - Failing to back up the key rings.
- More Such Actions
  - Sending a message to wrong recipients.
  - Sending a confidential message as plaintext.
  - Failing in signature verification.
  - Failing in revocation check.
  - Trusting the wrong keys.
- **How to communicate the consequences of these actions?**
- Consistency of terminologies
  - Encoding vs. Encrypting
  - Info is displayed/documented in several UI items and files; consistency requires some effort.
- "Too Much Information"
  - Defining what is important and what is not.
  - How to display info.
  - Basic, intermediate, and advanced levels in UI?

User Study Results

- Dangerous errors.
  - 3/12 sent confidential message in plaintext.
  - Two realized the mistake; one did not.
  - One forgot the passphrase.
- Figuring out how to encrypt with any key.
  - 1/12 failed to encrypt at all.
  - Two took close to half an hour.
- Figuring out the correct to encrypt with.
  - 7/11 used their own keys!
  - One generated key pairs for others!!
- Decrypting an email message.

    - 2/5 succeeded.
- Deciding whether to trust keys from the key server.
    - 3/8 members expressed concern.

****

Humans as an Attack Vector

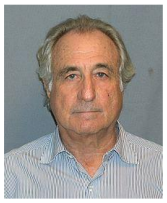
Hacking the Human: Why?

- Real attacks exploit psychology + technology
    - Example: Phishing, vishing, pretexting.
    - Targeted vs. Generic
    - Easy to launch online "imitation" attacks
        - Consider a physical bank branch vs. a bank website.
    - Deception is not new in human history.
        - But nature has trained us for detecting real-world, face-to-face deception.
- "The Art of Deception"
    - Kevin Mitnick
    - All his attacks were based on social engineering.

Attacker's Goals

- Information Extraction.
- Malware Infection
    - APT (Advanced Persistent Threat)
    - Long-term attack vector
- Stealth
- Monetary Gains

Targeting Humans: An Old Problem



- Targeting human psychology is not a unique problem in computer security.
    - Advertisements
    - Financial Fraudsters (Ponzi Schemes)
        - "Bernie" Madoff
    - Politicians
    - Terrorists

Why Phishing Works?

1. Natural Trust: Peers, authority
    a. Attacks via "trusted" contacts.
2. Target the Vulnerable: Elderly, young-adults, low-income.
    a. Attacks have social.
3. Fit with natural work-flow
    a. How to detect phishing emails when banks also send similar emails?
4. Greed
5. Social Validation

Phishing User Study

- "Why Phishing Works" (CHI 2006)

    - http://ww.eecs.berkeley.edu/~tygar/papers/Phihing/why_phishing_works.pdf
- Key Results
    1. Good phishing websites fooled 90% of participants.
    2. Existing anti-phishing browsing cues are ineffective.
    3. On average, our participant group make mistakes on our test set 40% of the time.
    4. Popup warnings about fraudulent certificates were ineffective.
    5. **Neither education, age, sex, previous experience, nor hours of computer use showed a statistically significant correlation with vulnerability to phishing.**

Why Phishing Works: Analysis of Past Attacks

- Identified three main factors from an archive of attacks involving social engineering.

1. Victims' lack of knowledge
    a. How to remain up-to-date?
2. Visual deception.
3. Bounded attention.
    a. Specifically, in identifying security "indicators"

****

The RSA SecureID Hack (April 2011)

- http://blogs.rsa.com/anatomy-of-an-attack/

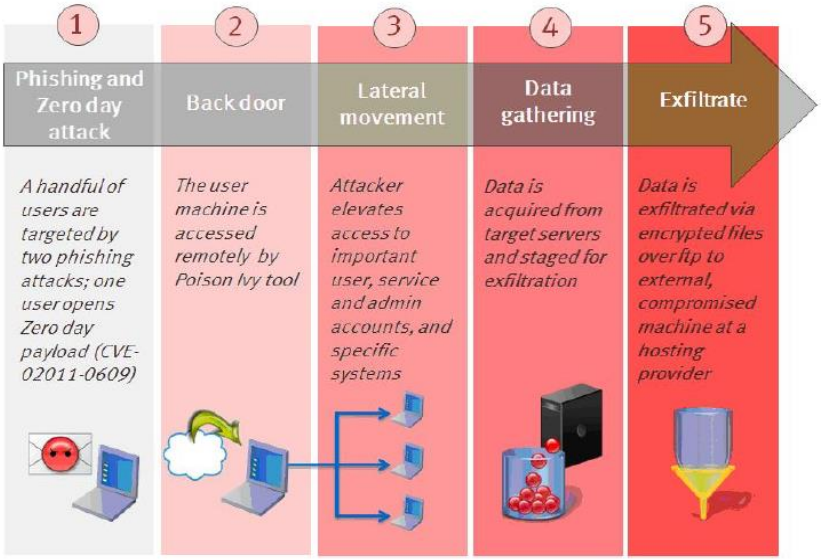
What is RSA SecurID?

- Hardware authentication token.
- Generates dynamic code using (valid for 30-60 sec)
    - Time.
    - Secret seed for each token.
- Authorize servers must be in sync and have *all* seeds
    - RSA uses own server for SecurID authentication for *all* of its customers (~40 million in 2011)

Use of RSA SecurID

- Generally used as a second factor.
    - In conjunction with a password.
- Used by many businesses, military.
    - High-value targets.
- Remember: Three factors of authentication**.**
    - Something you **know, have, are**

Attack at a Glance



The Attack: Social Engineering

- Phishing emails were sent to two small groups of employees (not high profile or high value targets). The email subject line read" 2011 Recruitment Plan."
- One of the employees retrieved it from their Junk mail folder, and opened the attached excel file ("**2011-Recruitment-Plan.xls"**).
- The spreadsheet contained a zero-day exploit that installs a backdoor through an **Adobe Flash** vulnerability (CVE-2011-0609)

The Attack: Infection

- An APT called "Poison Ivy" variant was installed in a reverse-connect mode.
    - More difficult to detect, as the PC reaches out to the command and control.
- The attacker first harvested access credentials from the compromised users (user, domain admin, and service accounts).
- They performed privilege escalation on non-administrative users in the targeted systems, and then moved on to gain access to key high value targets, which included process experts, and IT and non-IT specific server administrators.

The Attack: Exfiltration

- The attacker established access to staging servers at key aggregation points. Then they went into the servers of interest, removed data and moved it to internal staging servers where the data was aggregated, compressed and encrypted for extraction.
- The attacker then used FTP to transfer many password protected RAR files from the RSA file server to an outside staging server at an external, compromised machine at a hosting provider.
- The file were subsequently pulled by the attacker and removed from the external compromised host to remove any traces of the attack.

****

A Different Twist: Can We Use Deception for Defense?

Deception in Nature

- How many toads are here?

Human Use of Deception

- As old as human existence…
- In War: "The Art of War" –Sun Tzu
  - "All warfare is based on deception"
  - Extensively used during the cold war.
  - First Iraq war.
- In Computer Security:
  - "The Cuckoo's Egg" –Cliff Stoll
  - "An Evening with Berferd" – William Cheswick

Deception Background

- Some old papers (up to 2012)
  - http://all.net/journal/deception/index.html
- The Deception Toolkit (DTK): all.net/dtk
- Honeypots: Many systems.
- Tripwire.
- Honeywords (Jules and Rivest)
- BogusBite (Yue and Wang)
- Uvauth (Zhao and Mannan)

Computer Deception

- Computer Deception is:
  - Planned actions to mislead attackers.
  - To cause them to:
    - Confuse.
    - Take/avoid specific actions.
  - Must be mixed with some "truth".
  - Attackers must be given "hope".

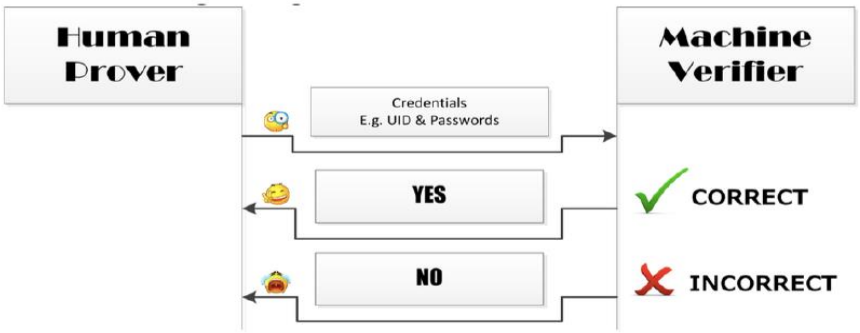
Deception Advantages

- Traditional Security: Detect/stop attacker's actions.
- Deception: Focus on attacker's perceptions.
  - "computers can lie" could be a powerful defense mechanism.
- Advantages:
  - False info can reduce value of leaked databases.
  - Learning the attacker's moves.
  - Slow down the attacker.

Using Deception against Online Guessing Attacks

- "**Explicit Authentication Response Considered Harmful"** (NSPW 2013)
- http://users.encs.concordia.ca/~mmannan/publications/uvauth-nspw13-post-proceedings.pdf

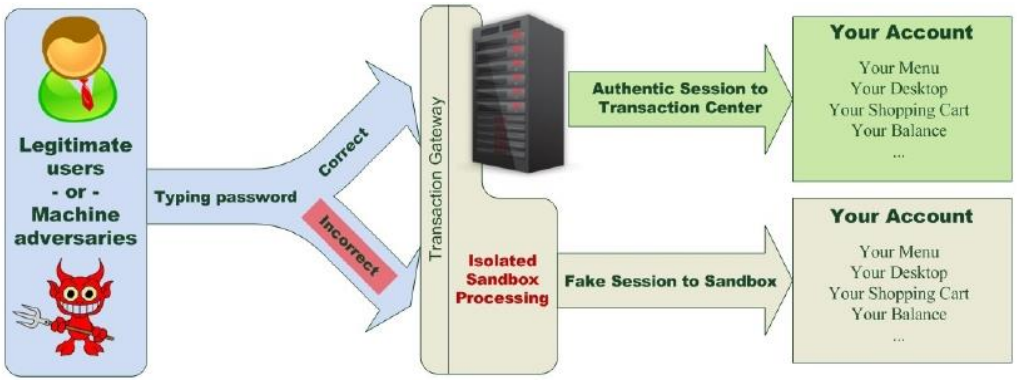
Root Cause of Online Guessing Attacks



Thread Model

- Attackers can solve CAPTICHAs.
- Non-personal interaction.
- Attacks from a large botnet.
- Attacker is patient.
- Assumption: No one knows what the account looks like better than the users themselves.
- Our Proposal is Called: Uvauth
    - User-verified authentication

Uvauth: At a Glance



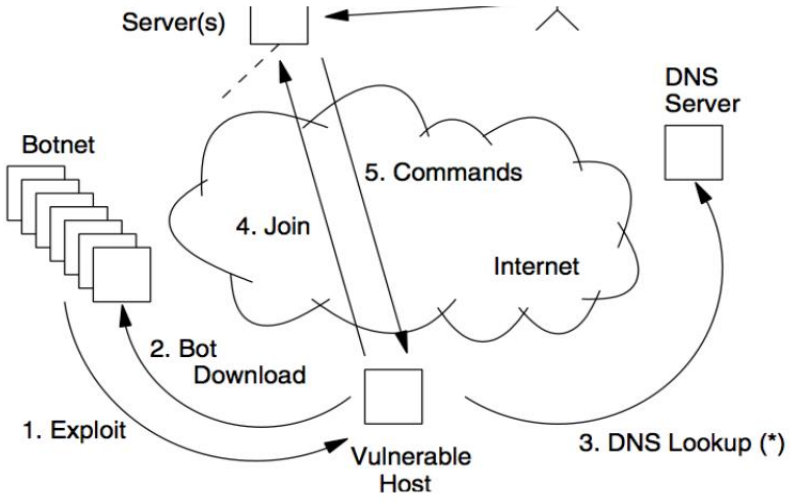
****

Botnets

Botnets – Definitions & Features

- **Botnets:** Networks or infected end-hosts, called bots, which are under the control of a human operator commonly known as a botmaster.
- **Bot Software:** Advanced malware that makes the functionality of a compromised host available to the botmaster.
- **Bots Can:**
    - Propagate like worms.
    - Hide from detection like many viruses.
    - Attack like many stand-alone tools.
    - Have an integrated command and control system.

Botnet – An Overview



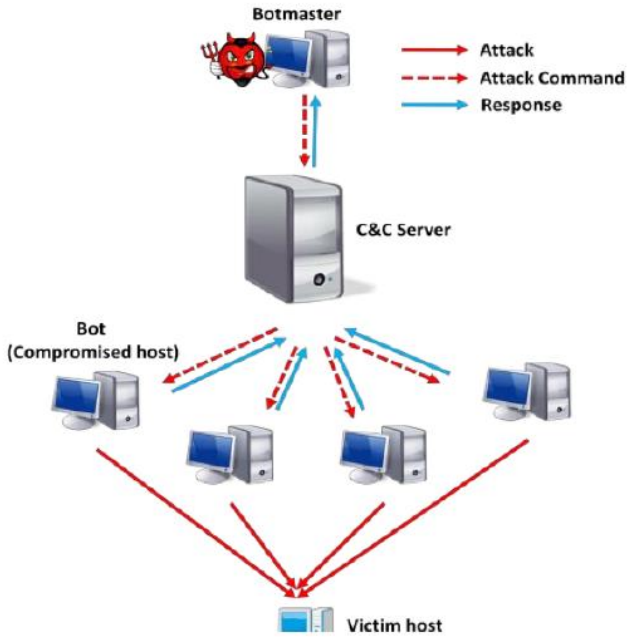
Propagation and Compromise

- How to put your bot in a user PC?
  - How about asking the user?
  - Users will not agree to do so, but generally they don't care if the machine is "usable" after an infection.
- Several independent infection/propagation mechanisms
  - OS/browser/application vulnerabilities.
  - Open file shares.
  - P2P Networks.
  - Backdoors from a previous infection.
  - Social Engineering, "curious George" attacks.
  - Trojaned applications.

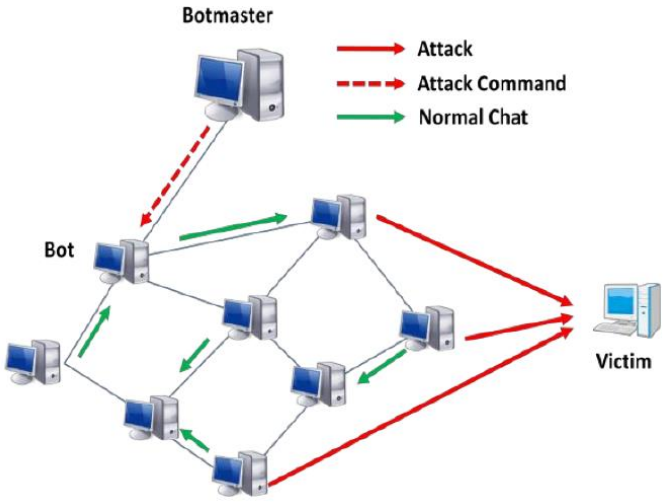
Command and Control

- A central part in ALL botnet design.
  - Centralized: Better control and efficient attacks, but single-point vulnerability.
    - Channels: IRC, HTTP; servers can also be hierarchical.
  - P2P: Robust against disruption, but inefficient and unreliable from attacker's viewpoint.

Centralized Architecture (IRC/HTTP)



P2P Architecture



DNS and Botnets

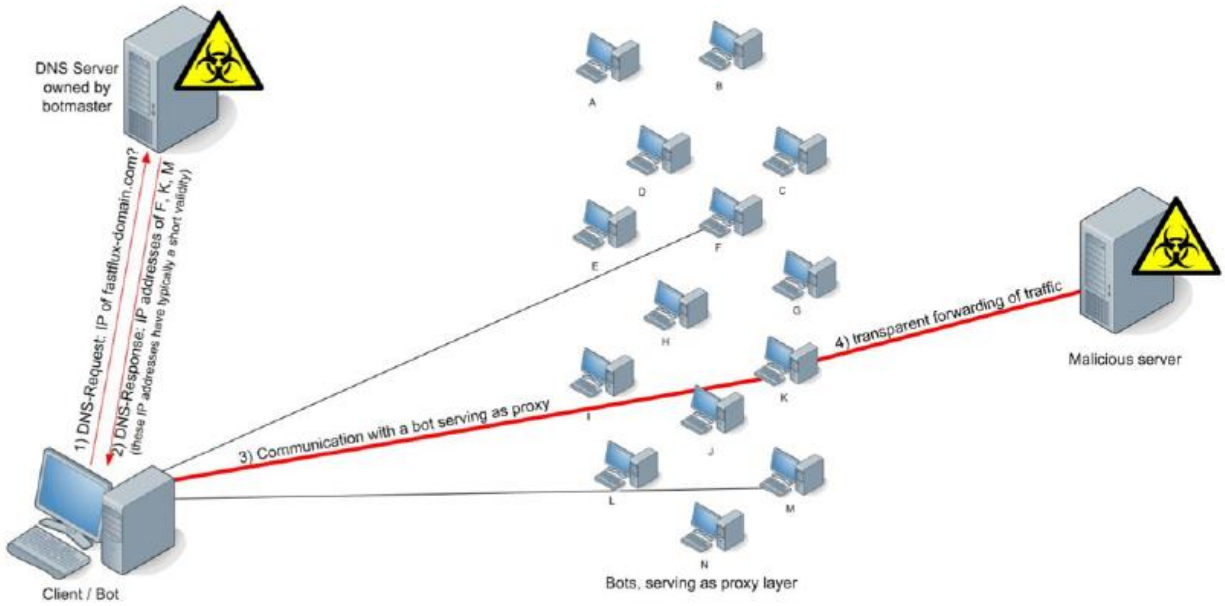
- Centralized C&C relies on DNS to move command servers from one IP to another.
- How to hide your malware/phishing servers?
- De-registration of malicious domains can happen.
    - Result in loss of control.
    - One main to dismantle botnets.
- How to survive, if you are a botmaster?
    - Fast-Flux Service Networks (FFSN)
    - Similar to Content Delivery Networks (CDN)

Fast-Flux

- One domain name is assigned to multiple (100's or 1000's) IP addresses.
- IP addresses are swapped in and out of flux with extreme frequency (e.g, 5 minutes)

- **Double Flux:** Fast-flux with a proxy layer (Optional)
    - See also: http://www.honeynet.org/node/136

Example Fast Flux



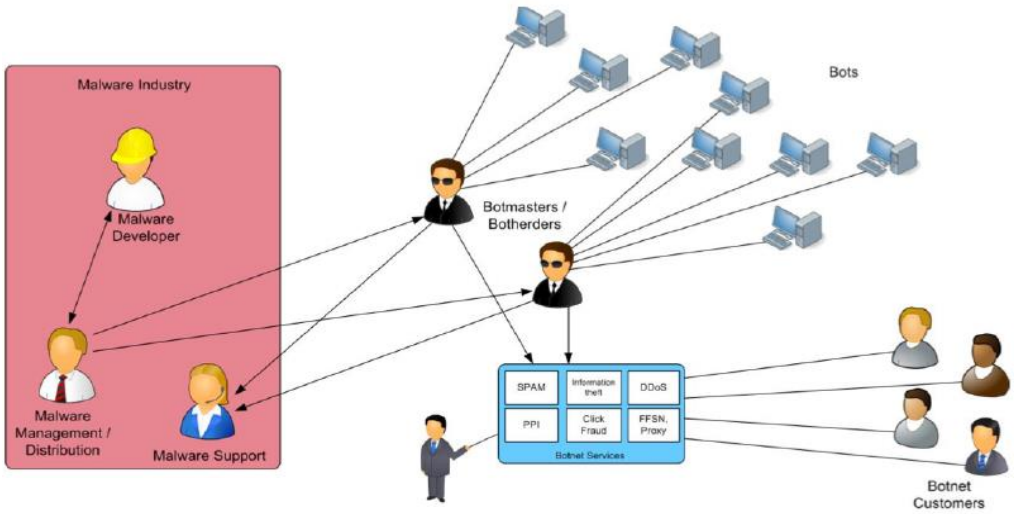
Dynamic Domain Names

- Still having a single domain is problematic.
- Register multiple domains.
    - But names can be learned in advanced and blocked.
    - Bots can be reverse-engineered to extract domains.
- Solution: **Domain Generation Algorithms (DGA)**
    - Generated domain names depending on one or more external information sources serving predictable seed values that can be accessed by bots and botmaster.
    - **Seed Values:** Timestamp, Twitter Trends

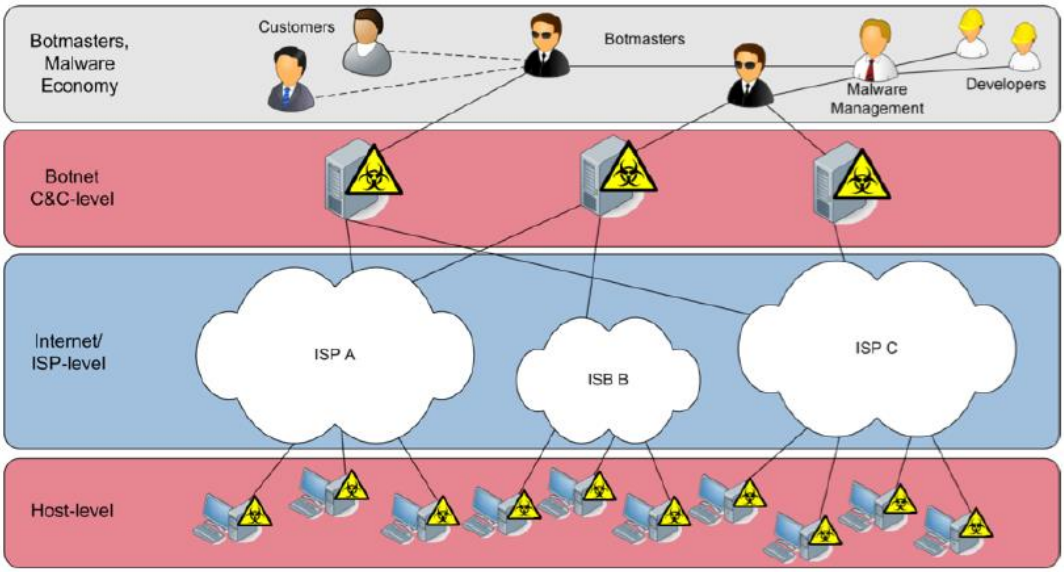
Botnet Usage

- Imagine: What will you do if you own the most robust and powerful computer at your hand?
- Can you put it into any good use?
- Known uses:
    - DDos, spam, credentials theft, click-fraud.
    - Pay-per-install (malware/adware/badware)
    - Political Agenda.
        - Estonia Attack (2007)
        - GhostNet (2009), Shadow Network (2010)
        - Stuxnext (2010), Flame (2012)

Botnet Economy – At a Glance



Botnet – Layers of Operation



Measurement and Detection Techniques

- **Passive Techniques**
  - Data collected from observations (honeypots)
  - Does not interfere with botnet activities
    - Transparent to botmaster.
- **Active Techniques**
  - Actively interact with the botnet to understand it.
  - Better understanding and measurements.
  - Bot activities may be disrupted, detectable by botmaster.
  - Researchers may be targeted by botmaster.
- **Revere Engineering of Bots**
  - Several anti-reverse engineering techniques are used: obfuscation, encryption, dynamic updating.

A) Passive Techniques

- **Packet Inspection**, through detection signatures.
- Match **protocol fields** or packet payload against pre-defined patterns of bot traffic, such as:
    - Packet with shell-code.
    - Communication with known malicious IPs.
    - Unrelated/unwanted protocol run by a server.
        - A file server that suddenly begins to communicate via IRC.
- **Implemented In:** Intrusion detection systems (IDS)
    - HIDS & NIDS
    - Also related tool, **intrusion prevention systems (IPS)**

Packet Inspection (Drawbacks)

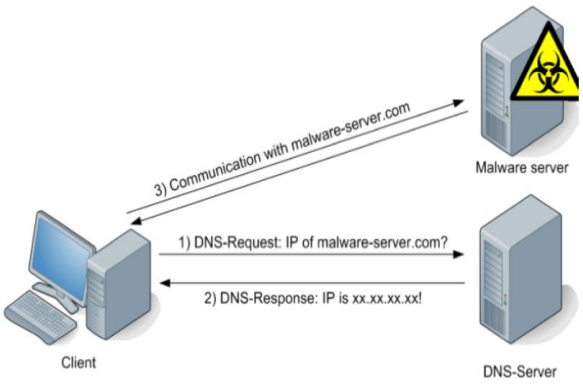
- Full packet inspection is costly and not scalable.
- Traffic sampling may not capture many important features.
- <u>High False Negatives:</u> May not detect anything beyond the signature database; multi-packet payload; encrypted payload.
- Dealing with false positives is difficult.

B) Active Techniques

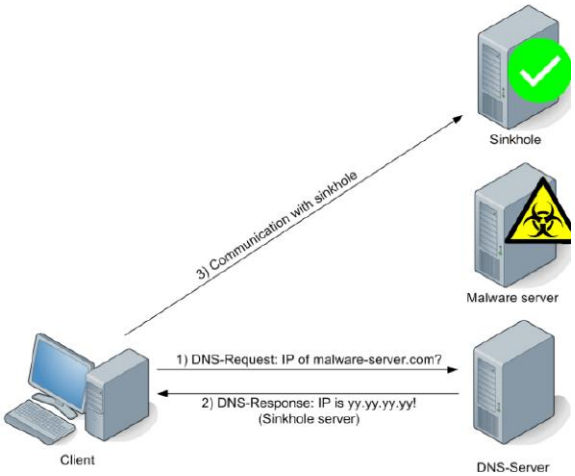
Sinkholing

- Redirecting or dropping traffic destined to a C&C server, malware distribution server or attack server.
- Sinkholing provides a view of the botnet's live population.
- <u>Recent Example:</u> Stuxnext measurement by Symantec.
- Running a sinkhole may get tricky.
    - What to do with sensitive data (ID theft, government data)

Sinkholing – Before Redirection



Sinkholing – After Redirection



Infiltration

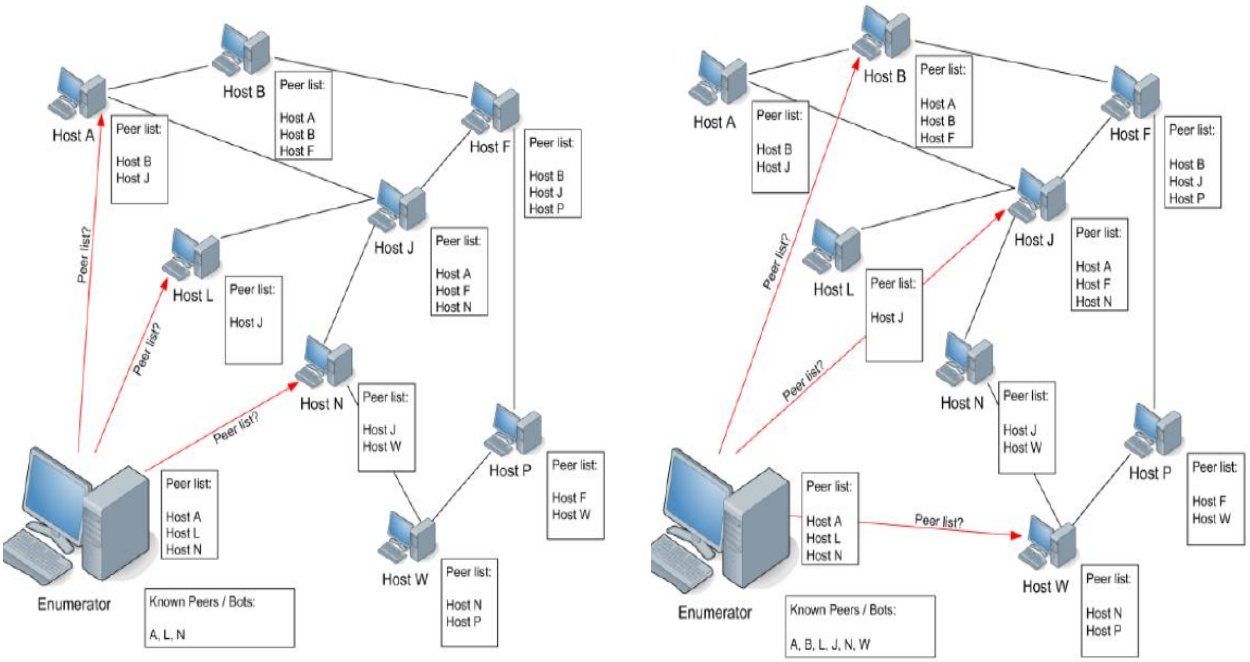
- <u>Software-Based</u>
    - Learning from inside.
    - Take control of botnet (protocol reverse engineering, exploiting "vulnerabilities")
- <u>Hardware-Based</u>

- Access to the ISP that's hosting C&C servers.
- Can monitor all traffic to/from servers.

Tracking P2P Botnets

- Robust against tracking, only few peers are known to each bot.
  - No central server to contact/track.
  - Used often: Storm, Waledac, Conficker
- Can still be measured
  - Recursive request of peer list (if available/supported by the protocol used)

Recursive Peer Request



Technical Countermeasures

- Blacklisting
- Fake credentials
  - Attacks the business model.
- Take down of C&C servers.
  - Law enforcement, ISP peer pressure.
  - Only applicable to centralized infrastructure.
- Blocking port 25 (restrict spamming)
- Remote patching, disinfection
  - Legality?

Recent Advancements

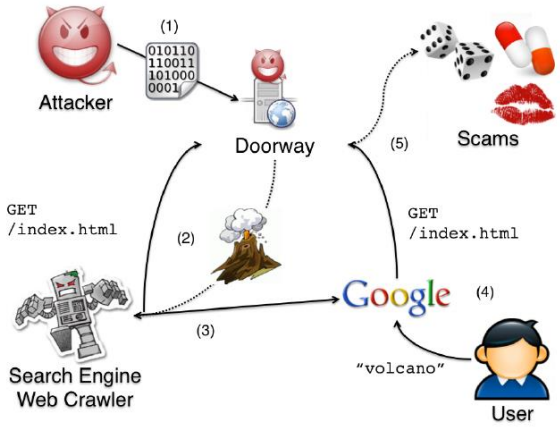
- Black Hat USA 2014 – "CloudBots: Harvesting Crypto Coins like a Botnet Farmer"
  - http://www.syscan360.org/slides/2014_EN_CloudBots_RobRaganOscarSalazar.pdf
- Build a botnet from freely available cloud services.
  - What are the advantages/disadvantages?
    - Free, more resourceful, reputable IP blocks.

        - Difficult to automate account creation?
    - Do we have enough services to exploit?
        - Amazon EC2, Google App Engine, CloudFoundry
- Can you use such a botnet?
    - Yes: Network scanning, password cracking, DDoS, LiteCoin mining.
    - How about using "less-resourceful" smart devices?

Botnets from other Resources

- How about using "less-resourceful" smart devices?
    - Smart-TVs, fridges, thermostats?
- How about compromised servers as bots?
    - Building botnets using compromised web servers.
    - Example botnet: GR
    - Analysis paper:
        - "Juice: A Longitudinal Study of an SEO Campaign", NDSS2013
        - http://internetsociety.org/doc/juice-longitudinal-study-seo-botnet

Steps in a Search Poisoning Attack



SEO Kit

- An **SEO Kit** is software installed on compromised sites.
    - Allows backdoor access for botmaster.
    - Performs Black Hat SEO (i.e. cloaking, content generation, user redirection)
    - Typically they are obfuscated code snippets injected into pages.

****

Tor and Internet Censorship

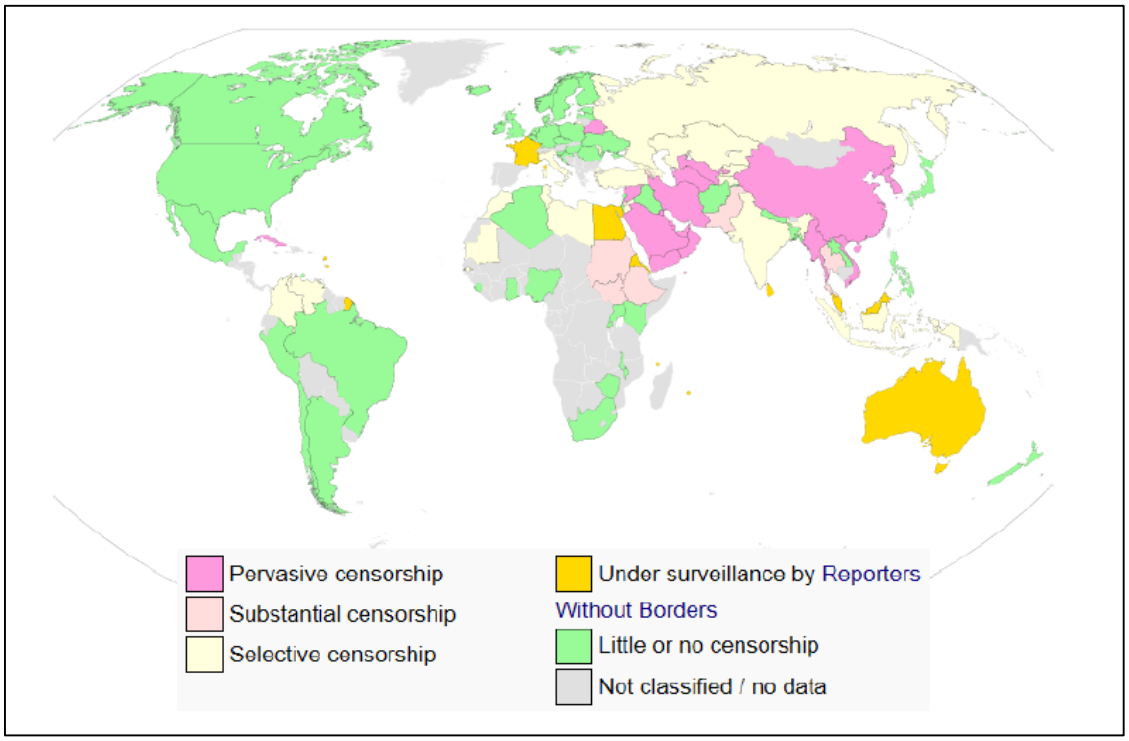
Content Sources

- Vitaly Shamitkov's Course
    - http://www.cs.utexas.edu/~shmat/courses/cs6431/anonnetworks.pdf
- "The Parrot is Dead: Observing Unobservable Network Communications"
    - Oakland2013 paper
    - http://freehaven.net/anonbib/cache/oakland2013-parrot.pdf
- Wikipedia…

Censorship of Internet Content

- The “Great Firewall” of China
  - But see here first: http://en.wikipedia.org/wiki/Internet_censorship
- May block pages with certain (e.g. political, religious) content.
- Change existing content on-the-fly to insert material that the attackers would like users to view.
- Can server malicious content.

Internet Censorship by Country



Existing Anti-Censorship Solutions

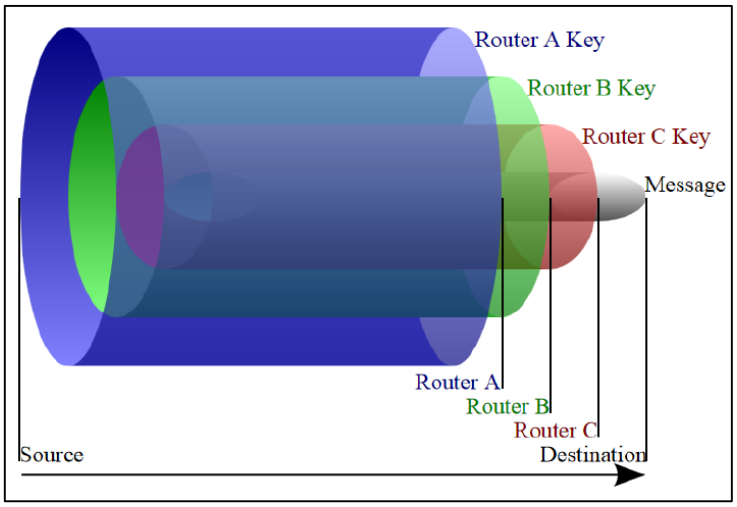
- Public/Private Web Proxies
  - How to determine whether a proxy is good or not?
- Personal proxies (Psiphon.ca)
  - How to make it available to intended users?
- Anonymity Networks
  - Tor
  - Freenet
  - AN.ON/JAP Anonymizer

Tor

- Second-generation onion routing network.
  - http://tor.eff.org
  - Specifically designed for low-latency anonymous Internet Communications (e.g, Web Browsing)
  - Running since October 2003
- Hundreds of nodes on all continents.
- Over 2,500,000 users.
- “Easy-to-use” client

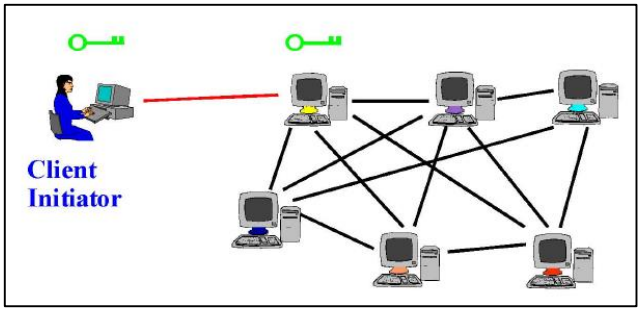
- Freely available, can use it for anonymous browsing.

Onion Routing

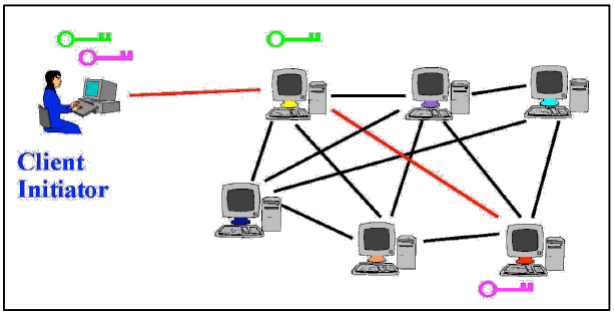


Tor Circuit Setup

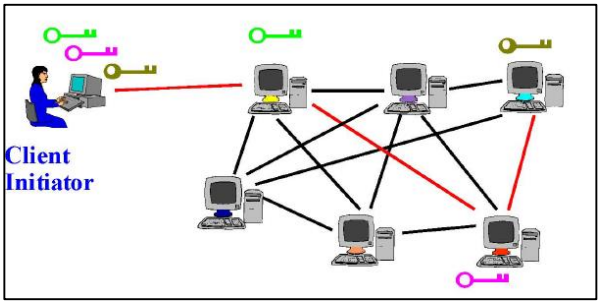
- **1)** Client proxy establishes a symmetric session key and circuit with Onion Router #1.



- **2)** Client proxy extends the circuit by establishing a symmetric session key with Onion Router #2
  - Tunnel through Onion Router #1

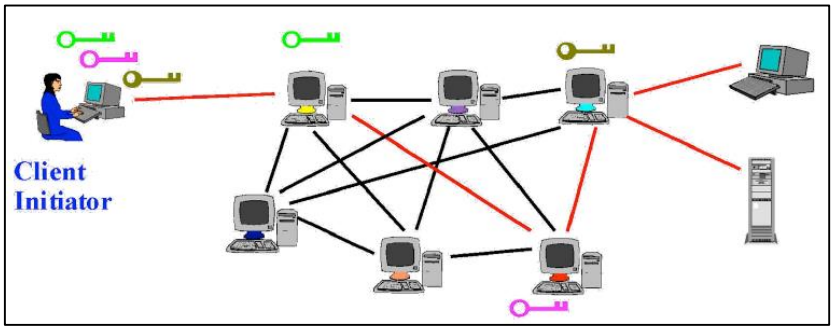


- **3)** Client proxy extends the circuit by establishing a symmetric session key with Onion Router #3
  - Tunnel through Onion Routers #1 and #2

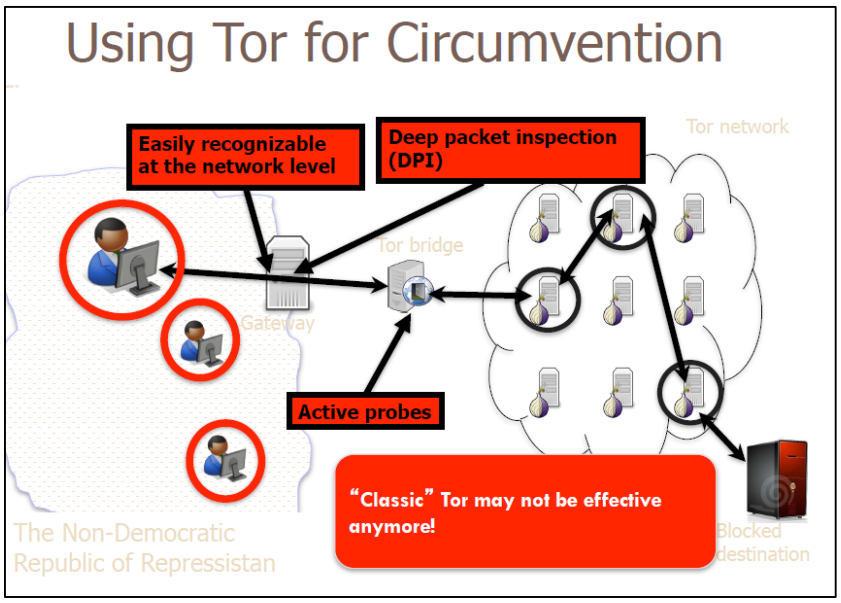


Using a Tor Circuit

- Client application connected and communicate over the established Tor circuit.
    - Datagrams decrypted and re-encrypted at each link.



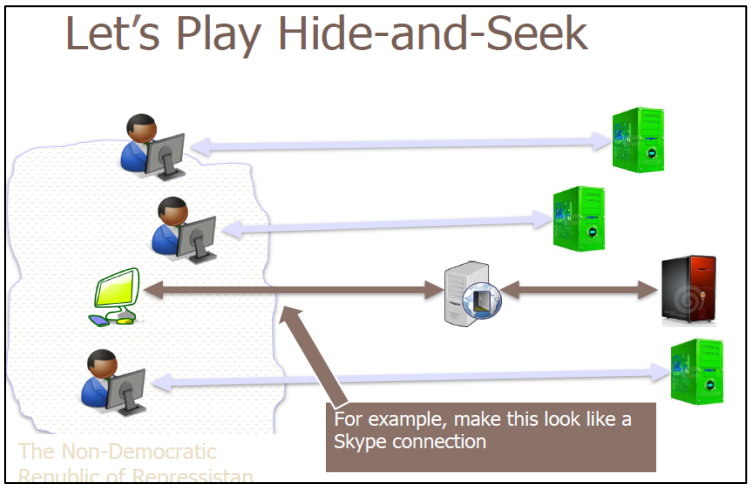
Using Tor for Circumvention



Tor: Weaknesses

- Blocking connection to directory servers/relay nodes/bridges.
- Filtering of Tor Traffic (via fingerprinting)
- **Sybil Attack** – Deploy malicious nodes to observe and correlate traffic.
    - May claim to provide large bandwidth to attract more traffic.
    - May reveal source identity from traffic analysis

Let's Play Hide-and-Seek



Goal: Unobservability

- Censors should not be able to **identify** circumvention **traffic**, **clients** or **servers** through passive, active or proactive techniques.

Unobservability by Imitation

- "Parrot Systems" imitate a popular protocol like Skype or HTTP
  - SkypeMorph (CCS 2012)
  - StegoTorus (CCS 2012)
  - CensorSpoofer (CCS 2012)

Core Issue

- DPI techniques are now affordable.
  - E.g, Tor private bridges did not solve the Tor Relay blocking issue.
- So tor has moved toward **"Pluggable Transports"**
  - Transform the Tor traffic flow between a client and a bridge.
  - DPIs will "see" innocent-looking transformed traffic.
  - A list of pluggable transports:
    - https://www.torproject.org/docs/pluggable-transports.html.en

Deployed Tor Pluggable Transports

- **Obfsproxy** is a Python framework for implementing new pluggable transports. It supports the obfs2 and obfs3 pluggable transports. Status: Deployed
- **Format-Transforming Encryption (FTE)** transforms Tor traffic to arbitrary formats using their language descriptions.
- **ScrambleSuit** is a pluggable transport that protects against follow-up probing attacks and is also capable of changings it's network fingerprint (packet length distribution, inter-arrival times)
- **Meek** is a transport that uses HTTP for carrying bytes and TLS for obfuscation. It uses a trick to talk to the third party so that it look like it is talking to an unblocked server.

FTE

- CCS2013 paper
  - https://kpdyer.om/publications/ccs2013-fte.pdf

- Allows the user to input a regex of their choosing and output ciphertext that are guaranteed to match it.
- Enables a built-in mechanism for forcing misidentification by regex-based DPIs.
- Testing against 7 DPIs, including a commercial one.

Targets in the Parrot Paper

- Three Target Systems:
  - **SkyeMorph**: Skype video calls (Tor pluggable transport)
  - **StegoTorus:** Skype + HTTP (Tor pluggable transport)
  - **CensorSpoofer:** SIP based VoIP (stand-alone program)
- Also, their variants
  - **Note:** These transports are not included in Tor Browser Bundle.

Claims and Reasons

- Simple, low-budget attacks are successful.
- **"Unobservability by imitation" is fundamentally flawed**.
- Convincingly mimicking a **sophisticated distributed system** like Skype, with multiple, inter-dependent sub-protocols and correlations, is an insurmountable challenge.
- To win, the censor needs only to find a few discrepancies, while the parrot must satisfy a daunting list of imitation requirements.
- It is not enough to mimic some protocol; the parrot must plausibly mimic a *specific implementation* of the protocol down to every quirk and implementation-specific bug.

Background

- Requirement for a Parrot system:
  - Generated traffic and end-points must remain "undetected".
  - Mimic an uncensored protocol.
  - Must target a widely-used protocol.
- Skype: Proprietary VoIP protocol on a P2P overlay network.
  - Elements: Client, Login Server**, supernode**
- IETP-Based VoIP
  - Protocols: Network discovery, session control, media transmission.

Adversary Model

- Capabilities
  - Passive: DPI, statistical traffic analysis.
  - Active: Content modifications, throttling bandwidth, connection reset.
  - Proactive
    - Discovery of Tor bridges by initiating connections to random or suspected IP addresses.
- Knowledge Classification
  - Local Adversary (LO)
  - State-Level Oblivious Adversary (OB)
  - State-Level Omniscient Adversary (OM)

Requirements for Parrot Systems

1. Mimicking the protocol in its entirety.

2. Mimicking reaction to errors and network conditions.
3. Mimicking typical traffic.
4. Mimicking implementation-specific artifacts.

1) Mimicking The Protocol In Its Entirety.

- **Correct:** Follow specification.
- **SideProtocols:** Must mimic all channels and side protocols of its target.
- **IntraDepend**
    - Changes in the main channel often cause observable activity in the control channel and vice versa.
    - Must faithfully mimic all dynamic dependences and correlations between sub-protocols.
- **InterDepend**
    - Trigger other protocols whenever the target protocol would have triggered them.
    - Mimic the target protocol's response.

2) Mimicking Reaction to Errors and Network Conditions

- **Err**
    - Malformed packets, invalid requests, unwanted traffic (e.g., packets from other sessions), etc.
    - Spec may prescribe how certain errors should be handled but error handling is often underspecified and left to the discretion of the implementation.
    - Differences in error handling can thus be used to fingerprint implementations.
- **Network**
    - Packets drops and reorder, dynamic changes in the throughput of certain links.
    - Packet losses and congestion cause media applications to lower codec quality and/or adjust transmission rates.

3) Mimicking Typical Traffic

- **Content**
    - Network protocols have specific formats for headers and payloads, all of which must be mimicked.
        - HTTP headers, PDF metadata
- **Patterns**
    - Protocols produce characteristic patterns f packet sizes, counts, inter-packet intervals, and flow rates.
- **Users**
    - User behavior often produces recognizable patterns as the network level. For example, a typical Skype user only makes a few Skype calls at a time.
- **Geo**
    - CDN
      SIP-based VoIP clients always connect to the geographically closest SIP server.
    - TOM-Skype

4) Mimicking Implementation-Specific Artifacts

- **Soft**
  - For inter-operability, each implementation generally complies with its (often idiosyncratic) interpretation of the standard, but often with characteristic quirks and tell-tale signs.
  - Browser implementations.
  - Different version of Apache Web server contain different bugs, which can be triggered by a remote user to identify the version.
- **OS**

Evaluation Results: Skype Imitations

- Can be detected by
  - Low-cost, passive attacks.
- "Improved" version of SkypeMorph and StegoTorus
  - Active and proactive attacks.

Passive Attacks

- Exploit
  - Deviations from genuine Skype.
  - Reuse of pre-recorded Skype traces.
  - Reuse of client-generated Skype traces.

Attack	Imitation requirement	Adversary	SkypeMorph	StegoTorus-Embed
Skype HTTP update traffic (T1)	`SideProtocols`	LO/OB/OM	Satisfied	Failed
Skype login traffic (T2)	`SideProtocols`	LO/OB/OM	Satisfied	Failed
SoM field of Skype UDP packets (T3)	`Content`	LO/OB/OM	Failed	Failed
Traffic statistics (T4, T5)	`Pattern`	LO/OM	Satisfied	Satisfied
Periodic message exchanges (T6, T7)	`SideProtocols`	LO/OB/OM	Failed	Failed
Typical Skype client behavior (T8)	`IntraDepend`	LO/OM	Failed	Failed
TCP control channel (T9)	`SideProtocols`	LO/OB/OM	Failed	Failed

Active and Proactive Attacks Against "Improved" Skype Parrots

Attack	Imitation requirement	Adversary	Skype	SkypeMorph+ and StegoTorus+
Verify supernode behavior by flushing supernode cache	`SideProtocols` `IntraDepend`	Proactive, LO/OM	The target node serves as the adversary's SN, e.g., relays his Skype calls	Rejects all Skype messages
Drop a few UDP packets	`Network`, `Err`	Active, LO/OB/OM	A burst of TCP packets on the control channel (Fig. 1)	No reaction
Close TCP channel	`IntraDepend`, `SideProtocols`	Active, LO/OB/OM	Ends the UDP stream immediately	No reaction
Delay TCP packets	`IntraDepend`, `SideProtocols`, `Network`	Active, LO/OM	Reacts depending on the type of TCP messages	No reaction
Close TCP connection to a SN	`IntraDepend`, `SideProtocols`	Active, LO/OB/OM	Client initiates UDP probes to find other SNs	No reaction
Block the default TCP port for TCP channel	`IntraDepend` `SideProtocols`	Active, LO/OB/OM	Connects to TCP ports 80 or 443 instead	No reaction

Evaluation: StegoTorus Chopper & HTTP (Passive Attacks)

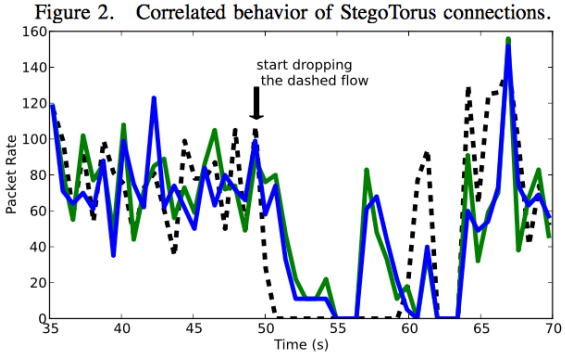
- Multiple concurrent HTTP and Skype connections from the same IP address.
- Multiple connections carry packets from the same Tor session.
- File-Format Semantics: Missing Objects

- PDF (xref Table)

Active and Proactive Attacks

- StegoTorus-HTTP: No Real Web Server
  - Simple replay of messages.
  - Can be easily fingerprinted.
  - Error responses can also be revealing.

Figure 2. Correlated behavior of StegoTorus connections.



****

Gracewipe: Secure and Verifiable Deletion under Coercion

Hiding the Existence of Data May Be Necessary

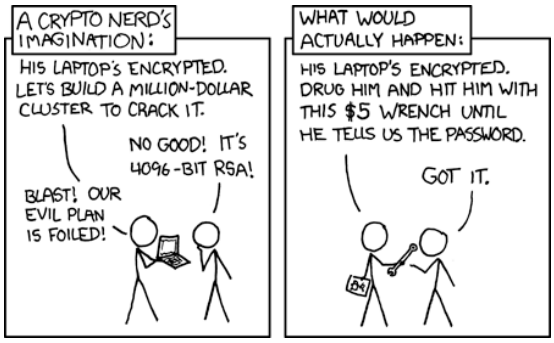
Why Coercion

Deniable storage encryption for mobile devices

Mobiflage

- A **Difficult** Problem.
- A **Long Lasting**, Tried Problem.

Coercion: Physical Control



Coercion: Time Is On Attacker's Side

"We've got all the time in the world, Repetto, so why don't we start with your childhood memories ?"

History

- First known definition (Marcus Ranum; 1990)

The Rubber-Hose Technique of Cryptanalysis

- A rubber hose is applied forcefully and frequently to the soles of the feet until the key of the cryptosystems is discovered.
- A process that can take a **surprisingly short time** and is quite computationally **inexpensive.**

The Stenographic File System

- First Academic Proposal (1998)

Ross Anderson

Roger Needham

Adi Shamir

The "Rubberhose" Filesystem (1997-2000)

- Pre-cursor of TrueCrypt's Hidden Volume


Julian Assange


Suelette Dreyfus


Ralf Weinmann

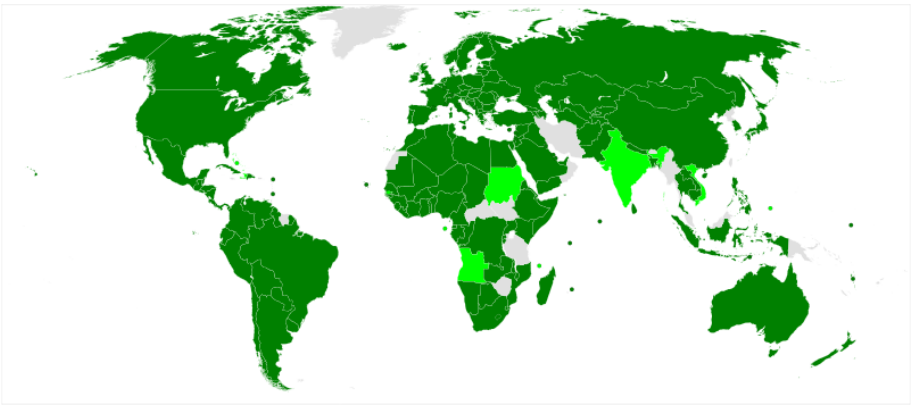
Is Coercion Real?

History of Torture (Right)

- Assyrians Skinning: Thousands of years ago (2600-605 BC)

UN Conventions Against Torture (Below)

- Map of nations that signed/ratified the convention (May 2009)



But In Reality (Top Right)

- Sata Jabar at Abu Ghrabi Prison, 2003

Any Known Case?

Existing Mitigations

1) Hiding the data

- Plausibly Deniable Encryption (PDE)
    - TrueCrypt
    - StegFS: multi-level hidden file system.

2) Deleting the Data

- Various secure deletion schemes.
    - ATA Secure Erase: Overwriting-based deletion.
    - Cryptographic deletion.

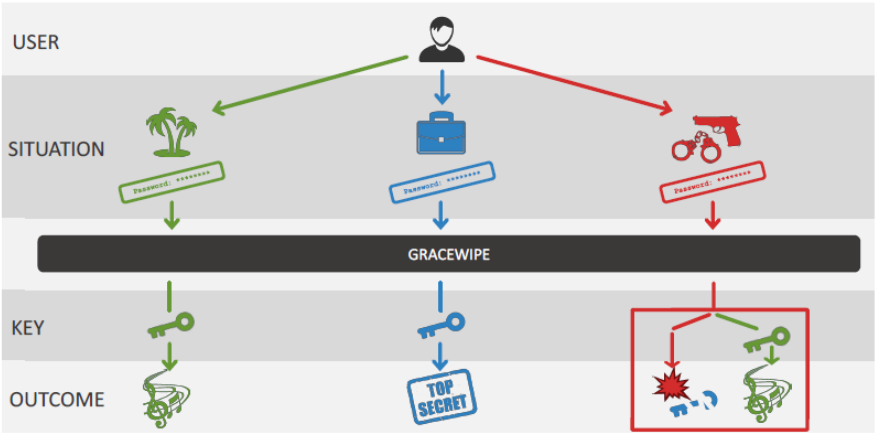
Academic Approaches (Important Slide)

1) Limited Try (Eric Rescorla; Requires HSM)
    - TPM Chip, to read the key from the BIOS.
2) Panic password schemes.
    - Cannot use locally because only designed for online systems. ***
    - Other problem is online parties you have to trust fully. ***
3) Passwords from measuring skin conductance (USENIX Security 2010)
    - When you are under stress apply on passwords, when not under stress real password. ***
4) Implicitly-learned passwords (USENIX Security 2012)
    - Playing a game. Though problem it took too long. ***

Our Solution: Gracewipe

- A pre-OS environment that acts upon user-entered password
    - Unlock keys and/or
    - Destroy keys
    - madiba.encs.concordia.ca

Gracewipe: Overview



Goals

1) **Undetectable** deletion triggering.
2) **Uninterruptable** deletion process.
3) **Verifiable** (cryptographically) deletion outcome.
4) Password Guessing
    - Offline: **Infeasible**
    - Online (via Gracewipe): **Risk of Deletion**
    - **Use Only Off-The-Shelf Hardware**

Terminology

- Two Systems:

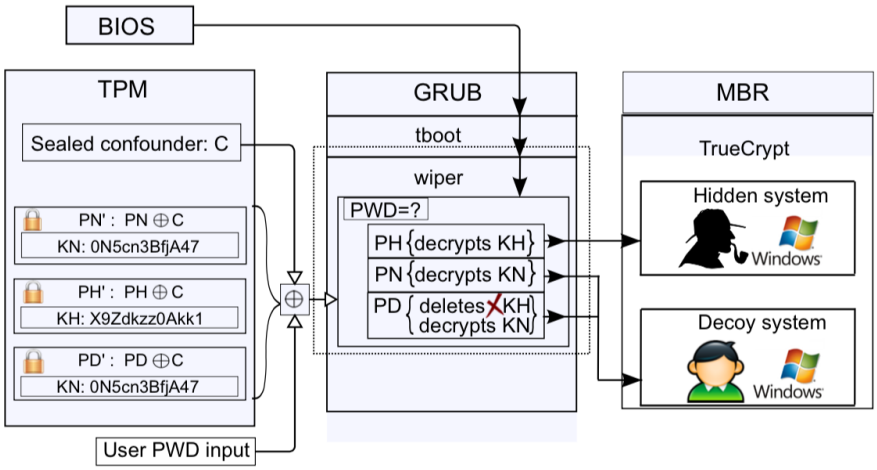
- ***Decoy*** (protected by key KN)
- ***Hidden*** (protected by key KH)

- Three (sets of) Passwords:
  - ***Normal Password*** (PN) that decrypts KN.
  - ***Hidden Password*** (PH) that decrypts KH.
  - ***Deletion Password*** (PD)

Assumptions

1. Rational adversary.
2. Very high-value data.
3. No simple attacks against TPM/TXT/FDE
4. The attacker does not get control over the machine when the hidden key is in memory.
   a. No cold boot prevention.
5. No evil maid attacks.
   a. Never trust an "unattended" machine.
6. No truth-serum, lie-detector.
7. Trustworthy storage encryption (TrueCrypt/SED)

Design and Workflow



Implementation

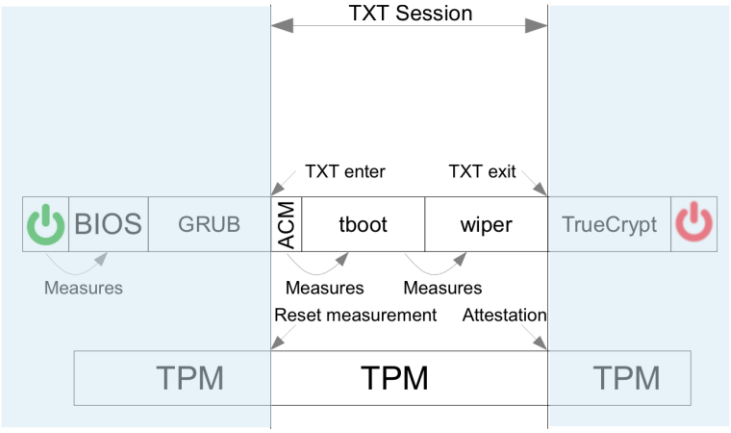
- Two Options:
  - TrueCrypt: Requires PH, PN, PD
  - SED: Requires PH, PD
- At the end, we boot to:
  - TrueCrypt/SED with Windows 7
  - SED with Linux (Ubuntu)

Building Blocks

- Trusted Platform Module (**TPM**)
  - Onboard Secure Storage
- Intel TXT Late Launch
  - Secure CPU Execution Mode
- A target FDE System

- **TrueCrypt**
  - Encryption tool with PDE
- **Self-Encrypting Drive** (SED)
  - Disk with Internal Encryption Engine

Chain of Trust



TPM

- Cryptographic Processor + RAM + NVRAM + Secure I/O
- Platform Configuration Registers (PCRs as RAM)
- Security-Enabling Operations
  - Extend: Chaining measurements to PCRs.
  - Seal: Binding data to platform in a given state.
  - Secure Storage: Protected NVRAM.

Implementation Challenges

- Pre-OS Environment (context switch, deice access etc.)
- Inherent Technical Restrictions (Intel TXT with Microsoft Windows)
- Effort to bridge all components (for minimum changes to maintain)

Security Analysis

1. Access to TPM commands via TPM pins.
2. Booting from a separate media.
3. Copying disk data first.
4. Attack SED interfaces.
5. User diligence.

Limitations

1. **TPM Deadlock**
   a. Can be fixed; may affect TPM's lifetime.
2. **Limited Number of PDs**
   a. Fixed (several password schemes)
3. **Degraded Disk I/O without DMA**
   a. Fixed

Summary

1. Coercion: A very strong but **realistic** adversary model.
2. Gracewipe: Enables a **non-bypassable**, verifiable deletion environment.
3. Exploring other uses of Gracewipe
    a. But it still a solution in progress.
    b. Comments/suggestions will be appreciated.

****