



Mississippi State
UNIVERSITY

J. A. “Drew” Hamilton, Jr., Ph.D.
Director, Center for Cyber Innovation
Professor, Computer Science & Engineering

CCI
Post Office Box 9627
Mississippi State, MS 39762

Voice: (662) 325-2294
Fax: (662) 325-7692
hamilton@cci.msstate.edu



Mississippi State University Center for Cyber Innovation



1

Domain 6: Cryptography & PKI

Reference:

Drew Hamilton Lecture Notes

Security+ Exam Guide, 5th ed.

Conklin, White, Cothren, Davis and Williams



Domain Outline

- **Cryptographic Concepts**
- **Cryptographic Algorithms**
- **Wireless Security**
- **Public Key Infrastructure**



Domain 6 Outline

- **6.1 Cryptographic Concepts**
- **6.2 Cryptographic Algorithms**
- **6.3 Wireless Security**
- **6.4 Public Key Infrastructure**



Cryptographic Concepts

Domain 6 Cryptography & PKI

Reference:

Drew Hamilton Lecture Notes

Security+ Exam Guide, 5th ed.

Conklin, White, Cothren, Davis and Williams



Mississippi State University Center for Cyber Innovation



5

6.1 Cryptographic Concepts

- **Purpose:**
 - Compare and contrast basic concepts of cryptography
 - This section introduces lots of cryptography vocabulary
 - Details will follow in the next three sections of the domain 6 presentation



6.1 Cryptographic Concepts

- **Critical elements of cryptographic systems:**
 - Proven cryptographic libraries (ALGORITHMS)
 - Proven cryptographically correct pseudorandom number generators (KEY sources)
- **Plaintext => Crypto system => Ciphertext**
- **Cryptanalysis = analyzing the ciphertext to try to determine what the plaintext (unencrypted) message was. Two special subsets of this:**
 - Differential Cryptanalysis
 - Linear Cryptanalysis



Two SubTypes of Cryptanalysis

- **Differential Cryptanalysis**
 - Comparing the plaintext (input) to the ciphertext (output) to attempt to determine the key being used
 - Requires that you have both the plaintext and the ciphertext. How?
 - You are testing your own cryptosystem
 - You managed to capture a plaintext and its corresponding ciphertext – and you expect the key to be used again
- **Linear Cryptanalysis**
 - Again, requires that you have both a plaintext message and its corresponding ciphertext
 - Run plaintext through a SIMPLIFIED cipher to try to determine what key might be used in full cipher



Intro: Types of Algorithms

- **Symmetric Encryption**
 - **Reversible: can encrypt and then decrypt using a single, shared key**
- **Asymmetric Encryption**
 - **Reversible: can encrypt and then decrypt using a KEY PAIR**
- **Hashing (a DIFFERENT way to hide info)**
 - **Not reversible; for storing secrets ... and more... later**
- **Covered in much more detail in the Cryptographic Algorithms section of this domain**



Symmetric Algorithms

- **Oldest and simplest form of encryption**
- **Basis: Both sender and receiver are holding the same secret shared key**
 - Requires mechanism for key management, so sender and receiver have access to that same shared key
 - **Critical: Store and send the key only through secure means (more on this later)**
- **“Simplest” form = Fewer computations**
 - **Faster (than asymmetric algorithms (more later))**
 - **Lower computational power required (than asymmetric)**
- **Problem: brute force attacks can succeed!**
- **Problem: key exchange and storage**



Asymmetric Algorithms

- “Public Key” cryptography
- Invented by W. Diffie and M. Hellman in 1975
- Uses two keys instead of one for each “individual” (entity participating in the crypto system). Called a key pair.
 - You have a PUBLIC key (available to everyone)
 - You have a PRIVATE key (~password; your secret)
- Encryption uses a public and a private key for each message sent
- Encryption depends on complex math, making it very difficult to determine second key, even when you have the first key (next slide)



Asymmetric Algorithms (cont)

- Remember: two separate but mathematically related keys for each entity.
 - Key #1 = private key
 - Key #2 = public key
 - Each key “undoes” the action of the other
 - If Encrypt with Key #1, the Decrypt with Key #2
 - If Encrypt with Key #2, then Decrypt with Key #1
- Public keys are distributed via CERTIFICATES
 - Certificate associates public key with entity
 - Certificate system verifies key is still valid, so public key can be “trusted”
 - More on this in fourth section in this domain



Using a Public-Private Key Pair

- **Example 1: “Digital Signature”**
 - John wants to prove a message is from him.
 - John encrypts the message with his **PRIVATE** key.
 - Any receiver can decrypt the message with John’s **PUBLIC** key.
- **Example 2: “Private Message”**
 - Susan wants to send John a private message.
 - Susan encrypts the message with John’s **PUBLIC** key.
 - Only John can decrypt the message, using his **PRIVATE** key.



Symmetric vs. Asymmetric

Symmetric

- **Less computational power needed => faster and good for less powerful computers**
- **Better for bulk transfers**
- **Key management (share and store) a challenge**
- **Does not support digital signature**

Asymmetric

- **Computational complexity much higher => slower**
- **Too complex (too many computer operations) for bulk transfers**
- **Does solve the key management issues**
- **Digital signature possible**



Symmetric vs. Asymmetric (cont)

- **Conclusions:**
 - Use a combination of symmetric and asymmetric to “get the best of both worlds.”
 - Use asymmetric encryption to pass the shared key between a sender and a receiver.
 - Improve security by sending a new shared key (aka, an EPHEMERAL key) for each connection.
 - Use the faster, less computationally complex symmetric encryption to pass large amounts of data between a sender and a receiver.



Hashing

- Not “encryption” in the plaintext to ciphertext and then back to plaintext sense
- Hashing function = special mathematical algorithm that provides a one-way (irreversible) encryption (encoding, often for the purpose of hiding) of the original information
- A hashing function should provide a unique result for each data to which it’s applied
 - Two different sets of data resulting in the same hashed result is called a collision; this is BAD.
- Hash value will be the same if someone else runs the same hash function against the same data.



What is hashing used for?

- **Store computer passwords**
 - Decreases the likelihood of password compromise
- **Ensure message integrity, using agreed-upon hash function**
 - Hash a message, producing a Message Authentication Code (MAC).
 - Send the message and the MAC produced above.
 - Receiver runs same hash function on received message.
 - If the hash produces the same MAC, message was not tampered with or modified. Can also detect non-malicious errors occurring during transmission.



What else is hashing used for?

- **Special subset of hashing:**
 - Using a previously shared secret in combination with the message, produce a Hashed Message Authentication Code (HMAC).
 - Provides authentication of sender, in addition to message integrity check.
- **Compute a message digest before applying the hash. This requires a smaller amount of data to be signed via asymmetric (computationally complex) encryption, while still providing assurances about data integrity and authentication.**
 - **But consider the requirements of computing digest!**



Hashing Basics:

- **Original data ...**
- **+ “source of additional entropy”**
 - **Salt**
 - **Initialization Vector (IV)**
 - **Nonce**
- **= Hashed result**

(entropy details to follow!)



Why add entropy before hashing?

- **Because “low entropy” data (e.g., a four-character password) could be difficult to hash and guarantee no collisions (two passwords hash to same value)**
 - **Collisions = opportunities for compromise!**
 - **More on this later**
- **Because “low entropy” data is much easier to attack via brute force**



Adding Entropy:

- **Salt: padding added to data before hashing**
 - High entropy
 - Example: 7Qjx55F8mRS931w0XfE5K2Ua5zJbY0
 - Thirty “pattern-free” characters
- **Initialization Vector (IV):**
 - Often used to achieve (pseudo)randomness even with inputs that are normally deterministic
 - Often used in wireless systems
- **Nonce: similar to salt or IV, but used only once**
 - If needed again, must use a different value



... and more Cryptography Concepts

- **Steganography**
- **Stream vs Block Ciphers**
- **More Issues Associated with Keys**
- **Data in Various States**
- **Some More Vocabulary**
- **Common Use Cases (situations to address)**



“Hiding in Plain Sight”

- **Steganography = “hiding” messages within pictures**
- **Black-and-white picture usually encoded as one-byte-per-pixel shades of gray, with values from ...**
 - **white = 0 (= 0000_0000 binary)**
 - **black 255 (= 1111_1111 binary)**
 - **{ _ just for legibility; legal symbol in Ada code}**
- **Color photos are usually encoded in Red-Green-Blue values, one byte per color (so three bytes per pixel), to represent a full-color image.**
 - **White = 0,0,0 (decimal)**
 - **Black = 255,255,255 (decimal)**



Steganography - 1 of 2

- **Basic Principle: Change just the Least Significant Bit (LSB) of each pixel to hide your message**
 - The human eye won't be able to detect the color change resulting from changing only the LSB, whether you are using the one-byte black/white scheme or the RGB full-color scheme.
 - Examples below using black/white and binary
 - 1111_1111 (pure black) looks like 1111_1110
 - One character takes 8 bits (ASCII coding), so it can be hidden in the LSB of 8 bytes of an image
 - A one-million pixel black/white image (i.e., 1,000 by 1,000 pixels) can hold 125,000 characters (1,000,000 bytes divided by 8 bytes / char)



Steganography – 2 of 2

- Showing four BYTES per line:
- 1111_1110.1111_1111.1111_1110.1111_1110
- 1111_1111.1111_1110.1111_1110.1111_1110
 - 8 LSBs = 0100_1000 = 'H'
- 1111_1110.1111_1111.1111_1111.1111_1110
- 1111_1111.1111_1111.1111_1110.1111_1110
 - 8 LSBs = 0110_1100 = 'i'
- 1111_1110.1111_1110.1111_1111.1111_1110
- 1111_1110.1111_1110.1111_1110.1111_1111
 - 8 LSBs = 0010_0001 = '!'
- Message: **Hi!**



Steganography – Remarks – 1 of 2

- **ASCII = American Standard for Information Interchange**
- **Technically, standard ASCII only uses 7 bits per character**
 - **But computers usually allocate one entire byte (8 bits) per character, because of the way computer memory is accessed**
- **There is now an Extended ASCII code that uses all 8 bits, and allows for some foreign-language characters (accent egu in French; the “double S” as in Strasse in German, etc)**



Steganography – Remarks – 2 of 2

- The difficulty of analyzing photographs to find hidden messages can be increased by changing the algorithm used to embed the messages
 - Don't start the encoding at the first byte in the image; start it at some other byte (e.g., 423).
 - Don't modify the last bit of every consecutive byte:
 - Use every other byte
 - Skip one byte, then skip two bytes, then skip three bytes, then repeat this pattern
 - ...and so on...
- The value of analyzing network traffic among suspects decreases when they're "just looking at pictures that everybody looks at..."



... and more Cryptography Concepts

- **Steganography**
- **Stream vs Block Ciphers**
- **More Issues Associated with Keys**
- **Data in Various States**
- **Some More Vocabulary**
- **Common Use Cases (situations to address)**



Block vs Stream Ciphers

- Data presents itself in one of two primary modes.
- Block data is available in large chunks at a time
 - This allows the use of much more complex ciphers
 - Transposition and substitution operations can be used
 - Data from the previous block can be used to obfuscate data from the current block (more on this in next section; XOR is a completely reversible function)
- Stream data is common with audio and video over the Web
 - Data not available in large chunks, only bit by bit or byte by byte
 - Only substitution operations are possible
 - Much less robust protection than block ciphers



Block Ciphers

- **Provide better data protection**
- **Require more memory to run**
- **Are resistant to modifications/insertions**
- **Are susceptible to error propagation, as a previous block is often used in encrypting the next block**
- **Can provide integrity verification and authentication**

More detail in next section of this domain!



Stream Ciphers

- **Faster than block ciphers, as they operate on smaller pieces of data and are less complex**
- **Are much more difficult to implement correctly**
- **Are susceptible to insertions/modifications**
- **Are less likely to produce error propagation, as they don't use prior data to encode new data**
- **Are not able to provide integrity or authentication**

More detail in next section of this domain!



... and more Cryptography Concepts

- **Steganography**
- **Stream vs Block Ciphers**
- **More Issues Associated with Keys**
- **Data in Various States**
- **Some More Vocabulary**
- **Common Use Cases (situations to address)**



More Issues Associated with Keys

- Keys are critical to cryptography
- In symmetric cryptography, the issue is maintaining the secrecy of the key
 - Out-of-band transmission: A courier might transport the key
 - In-band transmission: How do you protect the secrecy of the key during transmission?
 - Answer: Use Diffie-Hellman asymmetric key exchange! (more info in next section of this domain)
- In asymmetric cryptography, ensuring that all necessary parties have access to your public key is a primary issue.



More Issues Associated with Keys

- **Key Strength**
 - Larger (longer, more bits/bytes) key has more entropy; adds strength to encryption
 - Different algorithms use keys in different ways, so it's often not useful to try to compare key strength across different algorithms
- **Session keys**
 - Only used for one communication session
 - Generated from random seeds (fed into pseudorandom number generators)
 - Provide strong protection when properly generated and distributed during session set-up



More Issues Associated with Keys

- **Ephemeral keys**
 - By definition, used only once after generation
 - Can be incorporated into Diffie-Hellman scheme to form Ephemeral Diffie-Hellman (EDH) key exchange
- **Key Stretching**
 - Purpose: Make “weak” keys harder to break via brute force attacks
 - How: Increase the computational complexity by adding iterative rounds of processing that cannot be done in parallel (helps defeat distributed processing attacks and requires more computational power/more computing time for brute force attacks on one computer)



... and more Cryptography Concepts

- **Steganography**
- **Stream vs Block Ciphers**
- **More Issues Associated with Keys**
- **Data in Various States**
- **Some More Vocabulary**
- **Common Use Cases (situations to address)**



Data in Various States – 1 of 2

- **Data-at-Rest**
 - An entire laptop hard disk; data on the cloud
 - Protected via data encryption
- **Data-in-Use**
 - Data stored in a non-persistent state
 - In RAM, CPU caches, CPU registers
 - Data that would be lost without power
 - Need to protect this data from other processes and malware



Data in Various States – 2 of 2

- **Data-in-Transit**
 - Data being transported over a network
 - Protected by Transport Encryption
 - OSI Layer 4: The Transport Layer
 - Transport Layer Security (TLS)
 - Secure Sockets Layer (SSL) Security

 - The ability to ensure secure transport of data is essential to many modern computer systems!
- **Understanding which state the data is in is essential to determining how to protect it**



... and more Cryptography Concepts

- **Steganography**
- **Stream vs Block Ciphers**
- **More Issues Associated with Keys**
- **Data in Various States**
- **Some More Vocabulary**
- **Common Use Cases (situations to address)**



Some More Vocabulary

- **Diffusion**
 - A change in one character of the plaintext should result in a change of more than one character in the ciphertext.
 - Otherwise, changes in the ciphertext give away a lot of information about the structure of the plaintext.
- **Confusion**
 - Ensuring that each character in the ciphertext depends on several parts of the key
 - Increases randomness of output
 - Increases entropy of ciphertext



Some More Vocabulary

- **Collision**
 - **Two different inputs have the same output (e.g., for a hash function)**
 - **By definition, this is mathematically feasible:**
 - **Possible inputs to a hash function are, theoretically, infinite**
 - **Number of unique outputs of hash function are limited by size of hash output**
 - **Choice of hash function for a specific purpose is important!**



Some More Vocabulary

- **Obfuscation**
 - **Definition: Masking an item to make it unreadable (unintelligible) but still usable**
 - **Example: Storing hashed passwords**
- **Security through Obscurity**
 - **Don't make it easy for attackers to guess the names of important objects (e.g., don't name servers after the colors in the rainbow)**



Random vs. Pseudorandom Numbers

- **“Random” numbers are needed by many crypto applications**
- **True random numbers have no correlation to previously generated random numbers**
- **True random numbers have a uniform (equal) frequency distribution over the range of interest**
- **Therefore, the next “random” number cannot be predicted, even given all the previous random numbers, with a probability greater than “pure chance.”**
- **However ...**



Random vs. Pseudorandom Numbers

- **When the “random” number is generated by a computer algorithm, it’s not truly random, even though the numbers generated by the algorithm show the proper statistical distribution over the range of interest.**
 - **if one knows which algorithm is being used and what the initial “seed” value was, one can predict the next number.**
- **There are random/pseudorandom number generation algorithms that have been developed for cryptographic use; these minimize the predictability of the next number**



Random vs. Pseudorandom Numbers

- **The significance of unpredictability cannot be overstated for crypto purposes!**
- **Ensure that you use cryptographically secure random number generation methods to avoid compromising your encryption systems!**
- **This concept is important for everyone, but the implementation of these “almost perfect” random number generators is likely to be restricted to a very small group of people...**



Crypto Service Provider (CSP)

- **Software library containing cryptographic functions**
- **Example: Microsoft Windows**
 - **Microsoft CryptoAPI**
 - **Provides standard implementation for a complex set of processes**



Crypto Module

- **Definition: A hardware, software or hybrid device that provides cryptographic operations within a physical or logical boundary**
 - **Crypto keys do not leave that logical boundary**
 - **Example 1: Military hardware crypto device that is stored in a safe, loaded with a key immediately before a mission, and physically connected to a piece of military equipment (e.g., communication device) during the execution of that mission**
 - **Example 2: Key (physical) for a Secure Telephone Unit**



Perfect Forward Secrecy

- **Definition:**
 - Property of a public key system in which a key derived from another key is not compromised, even if the originating key is compromised in the future
- **How is this accomplished?**
 - In cryptography, forward secrecy (FS), also known as perfect forward secrecy (PFS), is a feature of specific key agreement protocols that gives assurances your session keys will not be compromised even if the private key of the server is compromised.



Forward Secrecy

- Forward secrecy protects past sessions against future compromises of secret keys or passwords.
- By generating a unique session key for every session a user initiates, even the compromise of a single session key will not affect any data other than that exchanged in the specific session protected by that particular key.
- Forward secrecy further protects data on the transport layer of a network that uses common SSL/TLS protocols, including OpenSSL, which had previously been affected by the Heartbleed exploit.
 - Heartbleed results from improper input validation (due to a missing bounds check) in the implementation of the TLS heartbeat extension.



... and more Cryptography Concepts

- **Steganography**
- **Stream vs Block Ciphers**
- **More Issues Associated with Keys**
- **Data in Various States**
- **Some More Vocabulary**
- **Common Use Cases (situations to address)**



Common Use Case Issues

- **Low power devices**
 - **Examples include mobile phones and other portable electronics**
 - **These have limited computational power**
 - **Special cryptographic functions have been developed for these types of devices**
 - **Example: Elliptic Curve Cryptography (to be discussed in detail in next section of this domain)**
- **Low latency requirements**
 - **Some applications have very tight time constraints**
 - **Near-real-time processing**
 - **Stream ciphers support these rigid time constraints**



Common Use Case Issues

- **High Resiliency**
 - **Systems capable of resuming normal operations after an external disruption**
- **Support of Confidentiality**
 - **Protect data at rest, in use, and in transit, via cryptographic systems**
- **Support of Data Integrity**
 - **Demonstrate that the data has not been altered**
 - **Example: Use of Message Authentication Codes (MAC) supported by hash functions**



Common Use Case Issues

- **Support of Obfuscation**
 - Protect information from casual observer
 - (Poor) Example: write down PIN code but change order of digits
- **Support of Authentication**
 - Trust identity of entity with whom you are communicating
- **Support of Non-repudiation**
 - Prove who sent a message
 - Requires that private key be kept secret, or revoked if it is compromised!



Resources vs Security Constraints

- **Cryptographic functions use computer resources**
- **A suitable balance must be found between minimizing the computer resources used for cryptography while still providing adequate security**
- **This is a critical initial consideration when initiating a cryptographic system implementation.**



Domain 6: Cryptography Concepts

- **Summary**
- **Compare and contrast basic concepts of cryptography**
- **This section introduced lots of cryptography vocabulary**



Cryptographic Algorithms

Domain 6 Cryptography & PKI

Reference:

Drew Hamilton Lecture Notes

Security+ Exam Guide, 5th ed.

Conklin, White, Cothren, Davis and Williams



6.2 Cryptographic Algorithms

- **Purpose:**
 - Address Security+ exam objective 6.2:
 - “Explain cryptography algorithms and their basic characteristics.”
 - This section includes the **NAMES** of lots of different algorithms, and key info about the characteristics of these algorithms.
 - These **NAMES** are “fair game” for the certification exam!
 - Focus on the **CONCEPTS** associated with each of these algorithms if you aren’t planning to take the certification exam.



Cryptographic Algorithms: Outline

- **Block Symmetric Algorithms (5)**
 - **Cipher Modes for BLOCK Symmetric Ciphers (4)**
- **Stream Symmetric Cipher (1)**
- **Asymmetric Algorithms (5)**
- **Hashing Algorithms (3)**
- **Key Stretching Algorithms (2)**
- **(Simple) Obfuscation Methods (3)**



Cryptographic Algorithms: Outline

- **Block Symmetric Algorithms (5)**
 - **Cipher Modes for BLOCK Symmetric Ciphers (4)**
- **Stream Symmetric Cipher (1)**
- **Asymmetric Algorithms (5)**
- **Hashing Algorithms (3)**
- **Key Stretching Algorithms (2)**
- **(Simple) Obfuscation Methods (3)**



Block Symmetric Algorithms List

- **DES**
- **3DES**

- **AES**

- **Blowfish**
- **Twofish**

(details to follow)



Block Symmetric Algorithms List

- **DES**
- **3DES**

- **AES**

- **Blowfish**
- **Twofish**

(details to follow)



DES

- **DES = Data Encryption Standard**
- **In 1973, the National Bureau of Standards, now known as the National Institute of Standards and Technology (NIST), sought proposals for a standard cryptographic algorithm.**
 - **Recall, publication of an algorithm allows testing for robustness; the security then depends on the key.**
- **A DES algorithm was developed, tested and accepted.**
 - **The algorithm was required to be recertified every 5 years by the National Security Agency (NSA).**
 - **DES passed certification in 1983, but was denied recertification by the NSA in 1987.**



DES (continued)

- **(DES Certification Woes, continued)**
- **No alternative to DES was offered when its recertification was denied.**
- **=> DES was recertified in 1993**
- **Work continued on replacements for DES**
 - **3DES (our next algorithm) is still popular**
 - **AES (to be discussed after 3DES) is new standard**



3DES

- **3DES = Triple DES**
- **Follow-on implementation of DES**
- **Uses multiple encryption rounds**
 - i.e., apply DES algorithm to message several times
- **Simplest Idea would be: DES alg, 3 keys**
 - Encrypt plaintext with key 1 => ciphertext 1
 - Encrypt ciphertext1 with key 2 => ciphertext2
 - Encrypt ciphertext2 with key 3 => ciphertext3
- **Actual 3DES harder to break**
 - Encrypt plaintext with key1 => ciphertext1
 - Decrypt ciphertext1 with key 2 => ciphertext2
 - Encrypt ciphertext2 with key3 => ciphertext3



3DES

- **Triple DES**
 - As of 2018, continues to be popular
 - Still widely supported
 - Much more difficult to brute force than DES
- ... but the new standard is AES



Block Symmetric Algorithms List

- **DES**
- **3DES**

- **AES**

- **Blowfish**
- **Twofish**

(details to follow)



AES

- **AES = Advanced Encryption Standard**
- **Symmetric, Block cipher**
 - Recall: symmetric = same key for encrypt and decrypt
 - Recall: Block = works on large chunks (blocks) of data at a time (as opposed to stream cipher, used on very small chunks of data)
- **Algorithm selected in Fall 2000, after evaluation of 5 proposed algorithms**
- **Three different standard (i.e., fixed) key sizes:**
 - AES-128 (128 bits = 16 bytes = 4 32-bit words)
 - AES-192 (192 bits = 24 bytes = 6 32-bit words)
 - AES-256 (256 bits = 32 bytes = 8 32-bit words)



AES

- Currently the gold standard for symmetric block cryptography algorithms
- Considered secure
 - No known efficient attacks against it
- Computationally efficient
 - Can be used even on limited-capacity devices



Block Symmetric Algorithms List

- **DES**
- **3DES**

- **AES**

- **Blowfish**
- **Twofish**

(details to follow)



Blowfish

- **Designed in 1994 by B. Schneier**
- **Symmetric block cipher**
- **Uses 64-bit blocks**
 - **= 8 bytes = 2 32-bit words**
- **Variable key length: 32 to 448 bits**
 - **448 bits = 56 bytes = 14 32-bit words**
- **Designed to run on 32-bit processors**
- **Optimized for situations with few key changes**



Blowfish: How secure?

- **Full version of Blowfish uses 16 rounds of encryption**
 - Only successful attacks have been against reduced-round variants of the algorithm
- **Why would you use a reduced number of rounds?**
 - **Processor power**
 - More rounds => more computations => uses more computer processor cycles
 - **Sensitivity of the information**
 - Fewer rounds => fewer computations => less processing capability dedicated to cryptography
 - Depending on sensitivity of info, might be “good enough”



Twofish

- **Was one of the finalists in AES competition**
 - Circa 2000
- **Scheier, Blowfish's author, on its development team**
- **Symmetric block cipher, using 128-bit blocks**
- **Variable-length key \leq 256 bits**
- **Available for public use**
- **Considered to be secure**
- **Improvement over Blowfish: less vulnerable to some classes of weak keys**



Summary: Block Symmetric Algorithms

- **DES – no longer in use**
- **3DES – based on DES; still popular**
- **AES – new gold standard for symmetric block cryptography**
- **Blowfish – not bad, but vulnerable to weak keys**
- **Twofish – Finalist for AES**



Cryptographic Algorithms: Outline

- **Block Symmetric Algorithms (5)**
 - **Cipher Modes for BLOCK Symmetric Ciphers (4)**
- **Stream Symmetric Cipher (1)**
- **Asymmetric Algorithms (5)**
- **Hashing Algorithms (3)**
- **Key Stretching Algorithms (2)**
- **(Simple) Obfuscation Methods (3)**



Cipher Modes for Block Symmetric Ciphers

- **ECB = Electronic Code Book**
- **CBC = Cipher Block Chaining**
- **CTM (aka, CTR) = Counter Mode**
- **GCM = Galois Counter Mode**

(details follow)



Cipher Modes for Block Symmetric Ciphers

- **ECB = Electronic Code Book**
- **CBC = Cipher Block Chaining**
- **CTM (aka, CTR) = Counter Mode**
- **GCM = Galois Counter Mode**



ECB

- **“Simplest” mode**
 - Defined for conceptual basis
 - Should not actually be used in crypto systems
- **Just divide message to be encrypted into blocks and then encrypt each block separately**
- **Problem: identical plaintext blocks produce identical ciphertext blocks, so attacker can tell they are identical**
 - So what?
 - Simplifies attacker’s decryption task: less (possibly MUCH less) data to decrypt
 - There might be “frequency analysis” value to the repetition



Cipher Modes for Block Symmetric Ciphers

- **ECB = Electronic Code Book**
- **CBC = Cipher Block Chaining**
- **CTM (aka, CTR) = Counter Mode**
- **GCM = Galois Counter Mode**



CBC

- **Concept:**
 - Repetition between blocks is hidden by “chaining” one block into the next block, using XOR function
- **Description:**
 - 1. Use Initialization Vector (IV) to obfuscate first block of plaintext by XORing the first block with the IV.
 - 2. Encrypt the obfuscated first block.
 - 3. XOR the encrypted first block with the plaintext second block, then encrypt this result to produce the second block of ciphertext.
 - 4. Repeat until entire message (block 1..n) is encrypted:
 - $\text{Temporary}[n] = \text{XOR plaintext}[n] \text{ with ciphertext}[n-1]$
 - **Encrypt Temporary[n]**



CBC: the XOR function

- **Background:**
 - XOR is a bit by bit logical function
 - XOR returns 1 if the two bits are different and 0 if same
 - $XOR(0,0) = 0$. $XOR(0,1) = 1$. $XOR(1,0) = 1$. $XOR(1,1) = 0$.
 - The XOR function, executed twice, “reverses” itself and returns the original result

 - Text: 1011_1000 Key: 0101_1011

 - Obfuscated = XOR (Key, Text) = 1110_0011
 - Text = XOR(Key, Obfuscated) = 1011_1000



Cipher Modes for Block Symmetric Ciphers

- **ECB = Electronic Code Book**
- **CBC = Cipher Block Chaining**
- **CTM (aka, CTR) = Counter Mode**
- **GCM = Galois Counter Mode**



CTM

- **Uses a “counter” function to generate a nonce to use for encryption of one block**
 - **Generates a new nonce for each individual block**
- **Process: Repeat for each block**
 - **Generate nonce**
 - **Encrypt using the key**
 - **XOR the above with the plaintext**
- **Each block can be encrypted independently, so parallel processing can be used**
- **The ciphertext independence of identical plaintext blocks is a result of the use of the nonce**



Cipher Modes for Block Symmetric Ciphers

- **ECB = Electronic Code Book**
- **CBC = Cipher Block Chaining**
- **CTM (aka, CTR) = Counter Mode**
- **GCM = Galois Counter Mode**



GCM

- This is an extension of the Counter Mode (CTM)
- This mode adds a Galois field
 - Mathematical representation (details beyond scope of this class)
 - Can be parallelized
- The Galois field adds an authentication capability to the cipher
- GCM used in many international standards
 - IEEE 802.1ad
 - IEEE 802.1AE



Summary: Cipher Modes for Block Symmetric Ciphers

- **ECB = Electronic Code Book**
 - Simple conceptual definition; not actually used
- **CBC = Cipher Block Chaining and**
- **CTM (aka, CTR) = Counter Mode** are considered secure and are the two most widely used modes.
- **GCM = Galois Counter Mode**
 - Adds authentication capability to cipher



Cryptographic Algorithms: Outline

- **Block Symmetric Algorithms (5)**
 - **Cipher Modes for BLOCK Symmetric Ciphers (4)**
- **Stream Symmetric Cipher (1)**
- **Asymmetric Algorithms (5)**
- **Hashing Algorithms (3)**
- **Key Stretching Algorithms (2)**
- **(Simple) Obfuscation Methods (3)**



Symmetric STREAM Cipher: RC4

- **RC = Rivest Cipher (designed by Ron Rivest, MIT)**
- **RC4 = most widely used stream cipher**
 - Found in Transport Layer Security (TLS) and WiFi Protected Access (WPA)
 - Can use variable key length, from 8 to 2048 bits
 - Common RC4 implementations use 128-bit keys
- **Very fast algorithm: enciphers plaintext in a stream, bit by bit, via (bitwise) XOR of plaintext stream and a generated key stream**
- **Vulnerable to weak keys that can generate bytes closely correlated with key bytes**
 - RC4 implementations must include weak key detection



Cryptographic Algorithms: Outline

- **Block Symmetric Algorithms (5)**
 - **Cipher Modes for BLOCK Symmetric Ciphers (4)**
- **Stream Symmetric Cipher (1)**
- **Asymmetric Algorithms (5)**
- **Hashing Algorithms (3)**
- **Key Stretching Algorithms (2)**
- **(Simple) Obfuscation Methods (3)**



Asymmetric Algorithms List

- **RSA**
- **DSA**
- **Elliptic Curve**
- **Diffie-Hellman**
- **PGP/GPG**

(details to follow)



Asymmetric Algorithms List

- **RSA**
- **DSA**
- **Elliptic Curve**
- **Diffie-Hellman**
- **PGP/GPG**



RSA

- **One of the first public key crypto systems**
- **Named for its inventors: Rivest (also developed RC4), Shamir and Adleman**
- **Uses the product of two very large (100-200 digit) prime numbers**
- **Based on the difficulty of factoring such large numbers**
- **Simple method, but has survived over 30 years**
- **Slow (can be 100 times slower than DES)**



Asymmetric Algorithms List

- **RSA**
- **DSA**
- **Elliptic Curve**
- **Diffie-Hellman**
- **PGP/GPG**



DSA

- **DSA = Digital Signature Algorithm**
- **Purpose:**
 - Provide traceability to author, and non-repudiation
- **Common implementation:**
 - Derivative of ElGamal signature method
 - Detailed in Federal Information Processing Standard 186 series
 - Covered by patent U.S. government has released worldwide, without royalties required
 - Uses a per-message random signature value, k
 - Must be kept secret; should not be reused
 - Reuse => compromise (Sony, signing software for PS3)



Asymmetric Algorithms List

- **RSA**
- **DSA**
- **Elliptic Curve**
- **Diffie-Hellman**
- **PGP/GPG**



Elliptic Curve Cryptography

- **Basis: Elliptic Curve is “simple” function**
 - Gently looping curve on X,Y plane.
 - Defined by equation: $y^{**2} = x^{**3} + ax^{**2} + b$
 - **** = raised to the power of ($y^{**2} = y * y$; $x^{**3} = x * x * x$)**
 - *** = multiplication**
 - **Special property: You can add two points on the curve together and get a third point on the curve**
 - **PUBLIC key based on:**
 - **Users agree on elliptic curve equation (specify a and b above).**
 - **Users agree on ONE fixed point on curve**
 - **Third component based on each user choosing his/her own secret random number**



Elliptic Curve Cryptography (cont)

- **Public shared info:**
 - Equation for curve: F
- **In order to generate a shared secret:**
 - User 1: chooses secret random number, $K1$
 - User 1: computes **Public_Key_1 (PK1) = $K1 * F$**
 - User 2: chooses secret random number, $K2$
 - User 2: computes **Public_Key_2 (PK2) = $K2 * F$**

 - User1: generates shared secret, **$S = K1 * PK2$**
 - **$S = \text{user 1's random number times user 2's public key}$**
 - User 2: can generate same secret, **$S = K2 * PK1$**
 - **$S = \text{user 2's random number times user 1's public key}$**



Elliptic Curve Cryptography: The Algebraic Proof

- The shared secret, S , can be shown to be the same through algebraic operations, given:
 - S (user 1) = $K1 * PK2$ (and $PK2 = K2 * F$)
 - S (user 2) = $K2 * PK1$ (and $PK1 = K1 * F$)
 - S generated by user 1 (to share with user 2):
 - = $K1 * PK2$
 - = $K1 * (K2 * F)$ (substitute $PK2 = K2 * F$)
 - = $(K1 * K2) * F$ (Associative property of $*$)
 - = $K2 * (K1 * F)$ (Commutative and Associative prop of $*$)
 - = $K2 * PK1$ (substitute $PK1 = K1 * F$)
 - S generated by user 2 (to share with user 1):
 - = $K2 * PK1$
 - Proof complete! $S(\text{user 1}) = S(\text{user 2})$



Asymmetric Algorithms List

- **RSA**
- **DSA**
- **Elliptic Curve**
- **Diffie-Hellman**
- **PGP/GPG**



Diffie-Hellman

- Very commonly used family of protocols for electronic key exchange, which lets two users with no prior contact exchange a key for SYMMETRIC algorithm use via an ASYMMETRIC public-private key system
- Diffie-Hellman is the gold standard for electronic key exchange!
- Used in many network layers/protocols
 - Secure Sockets Layer (SSL)
 - Transport Layer Security (TLS)
 - Secure Shell (SSH)
 - IP Security (IPSec)



Diffie-Hellman (DH) Protocols

- There are several sets of DH protocols
- DH Groups determine strength (length) of the key used in the key exchange process
 - DH1: key is 768 bits = 96 bytes = 24 32-bit words
 - DH2: key is 1024 bits = 128 bytes = 32 32-bit words
 - DH5: key is 1536 bits = 192 bytes = 48 32-bit words
- DHE = Diffie-Hellman Ephemeral
 - Temporary (one-time) key used for key exchange, rather than reusing same key
- ECDH = Elliptic Curve Cryptography (ECC) DH
 - Uses ECC to get DH key
- Ephemeral ECDH = generates one-time use ECC key



Asymmetric Algorithms List

- **RSA**
- **DSA**
- **Elliptic Curve**
- **Diffie-Hellman**
- **PGP/GPG**



PGP and GPG

- **PGP = Pretty Good Privacy**
 - Created in 1971 by Zimmerman
 - Uses both symmetric and asymmetric algorithms
 - Asymmetric algorithms allow transmission of shared key (for use in symmetric algorithms)
 - Flexibility to share a key through secure method
 - Symmetric algorithms perform bulk of encryption
 - Much faster than asymmetric algorithms
 - “The best of both worlds”
 - Originally available for free under noncommercial license
 - Now a Symantek commercial product
- **GPG next!**



GPG

- **GPG = Gnu Privacy Guard**
 - Open source implementation of OpenPGP standard
- **Command-line based**
- **Designed to protect electronic communications**
 - Example: email
- **Operation similar to PGP**
- **Includes capability to manage public and private keys**



Summary: Asymmetric Algorithms

- **RSA:** simple; product of two very large primes
- **DSA:** provides digital signatures
- **Elliptic Curve:** Simple but mathematically hard to break
 - Private key = secret random number chosen by user
 - Public key based on elliptic curve equation
- **Diffie-Hellman:** Gold standard for exchange of keys
 - Several variations within DH protocols
- **PGP/GPG** (PGP commercial; GPG open source)
 - Combine asymmetric and symmetric algorithms to provide security primarily for electronic communications (e.g., email)



Cryptographic Algorithms: Outline

- **Block Symmetric Algorithms (5)**
 - **Cipher Modes for BLOCK Symmetric Ciphers (4)**
- **Stream Symmetric Cipher (1)**
- **Asymmetric Algorithms (5)**
- **Hashing Algorithms (3)**
- **Key Stretching Algorithms (2)**
- **(Simple) Obfuscation Methods (3)**



Hashing Algorithms: Outline

- **Critical Principles**
- **MD5 (MD family)**
- **SHA family**
- **RIPEMD family**

(details to follow)



Hashing Algorithms

- **Critical Principles**
- **MD5 (MD family)**
- **SHA family**
- **RIPEMD family**



Hashing: Critical Principles

- **Same input always produces same output**
 - **Deterministic**
- **No “collisions” can be produced by hash fcn**
 - **Defn: two (different) inputs produce the same output**
 - **Avoiding collisions critically important!**
- **Collisions are theoretically possible for some hash functions (example: data compression)**
 - **If inputs can be of variable length, then set of inputs is theoretically infinite**
 - **If hash output is a specified number of bits (e.g., 256), then set of outputs is clearly not infinite**
- **Collisions are usually avoided by very complex processing within hash function**



Hashing: Critical Principles

- **Recall: Hash functions are not reversible**
 - There is no way to retrieve the original input to the hash function
- **Hash function uses**
 - Save passwords in more secure format
 - Data compression: produce digest (shortened version) of message
 - Can be used to detect message tampering or corruption
 - Can provide smaller “chunk” of data to digital signature algorithm



Hashing Algorithms

- **Critical Principles**
- **MD5 (MD family)**
- **SHA family**
- **RIPEMD family**



Hashing: The MD Family

- **MD5 = Message Digest 5 (prior versions exist)**
- **Developed by Ron Rivest (MIT) in 1991**
 - Author of RC4, Blowfish/Twofish
- **Purpose:**
 - Use secure method, reliable method to...
 - Compress message of variable length...
 - And produce output of a specific number of bits
- **Problem in 2007 with MD5: Collision!**
 - Two completely different Win32 executables could produce identical hash outputs
 - Catastrophic for security against malware
 - Forced adoption of stronger hash



Hashing Algorithms

- **Critical Principles**
- **MD5 (MD family)**
- **SHA family**
- **RIPEMD family**



Hashing: The SHA Family

- **SHA = Secure Hash Algorithm**
 - Published by NSA and NIST (National Institute of Standards and Technology)
 - Family of protocols for use in Digital Signature Standard
 - **SHA-1: 1993. Vulnerable to collisions. No longer used.**
 - **SHA-2: Replaced SHA-1.**
 - Input lengths up to 2^{64} bits (2 to the 64th power)
 - Variants with different hash output lengths: SHA-224, SHA-256, SHA-384, SHA-512 (SHA-n; n = # of bits in output)
 - **SHA-3: 2012. Completely different hash function**
 - Keccak hash won NIST competition to become SHA-3
 - Relatively new; not widely adopted yet
 - **SHA-2 and SHA-3 algorithms still approved for use**



Hashing Algorithms

- **Critical Principles**
- **MD5 (MD family)**
- **SHA family**
- **RIPEMD family**



Hashing: The RipeMD Family

- **RipeMD = RACE Integrity Primitives Evaluation Message Digest**
 - Developed by RACE Integrity Primitives Evaluation consortium
 - Originally provided 128-bit hash output
 - Problem with collisions led to development of RIPEMD-160
- **RipeMD-160**
 - Based on Ron Rivest's MD-4 algorithm
 - Enhanced with two parallel channels and five rounds
 - Outputs five 32-bit words (= 160 bits)
 - Extensions have been developed
 - RipeMD-256 (256-bit output)
 - RipeMD-320 (320-bit output)
 - “Not inherently stronger than RipeMD-160”



Special Hashing Subset: HMAC

- **HMAC = Hashed Message Authentication Code**
- **“Basic” MAC** can be used to determine if a message has been changed during transmission (**message integrity**)
- **With addition of secret key and crypto algorithm, can produce HMAC**
 - **Can now prove authenticity, in addition to integrity, of message**
- **Popular algorithms for HMAC:**
 - **MD5 (collision attack method not available here, so OK to use for HMAC purposes)**
 - **SHA-256**
 - **RipeMD-160**



Summary: Hashing Algorithms

- **Critical Principles**
 - **Must avoid collisions**
- **MD5 (MD family)**
 - **Intended use: Message Digest**
 - **Collisions for Win32 executables!**
 - **Still used in Hashed Message Authentication Codes**
- **SHA family**
 - **Intended use: Digital Signature Standard**
 - **SHA-1 vulnerable to collisions; no longer used**
 - **SHA-2 and SHA-3 families still approved for use**
- **RIPEMD family**



Cryptographic Algorithms: Outline

- **Block Symmetric Algorithms (5)**
 - **Cipher Modes for BLOCK Symmetric Ciphers (4)**
- **Stream Symmetric Cipher (1)**
- **Asymmetric Algorithms (5)**
- **Hashing Algorithms (3)**
- **Key Stretching Algorithms (2)**
- **(Simple) Obfuscation Methods (3)**



Key Stretching Algorithms

- **Purpose: Convert weak keys (that would be easily broken) to stronger keys**
- **Method: Increase computational complexity of key by executing multiple iterative rounds of computations.**
 - **Example: Extend key length via multiple rounds of variable-length hashing, increasing number of output bits each time**
 - **Computationally “expensive” but not an issue for infrequently-changed keys (e.g., password in 180 days)**
- **BCRYPT Algorithm**
- **PBKDF2 Algorithm**



BCRYPT Key Stretching Algorithm

- **BCRYPT uses the Blowfish cipher and salt**
- **Adds an adaptive function to increase the number of iterations**
- **Its computational requirements are acceptable for infrequently-changed keys (“single use”)**
- **Extremely hard to defeat via brute force due to the number of attempts required**
 - **Not “computationally feasible” at this time**



PBKDF2 Key Stretching Algorithm

- **PBKDF2 = Password-Based Key Derivation Function 2**
 - Designed to produce a key, based on a password
- Uses a password and a salt
- Applies an HMAC to the input thousands of times
 - Check md5calc.com to see the complexity of MD5 output for simple inputs!
- Repetitive application of HMAC makes brute force attacks computationally unfeasible
 - For foreseeable future...



Cryptographic Algorithms: Outline

- **Block Symmetric Algorithms (5)**
 - **Cipher Modes for BLOCK Symmetric Ciphers (4)**
- **Stream Symmetric Cipher (1)**
- **Asymmetric Algorithms (5)**
- **Hashing Algorithms (3)**
- **Key Stretching Algorithms (2)**
- **(Simple) Obfuscation Methods (3)**



(Simple) Obfuscation Methods

- **XOR**
- **Substitution Ciphers**
 - ROT13
 - Viginere Cipher
- **Transposition Ciphers**



(Simple) Obfuscation Methods

- **XOR**
- **Substitution Ciphers**
 - ROT13
 - Viginere Cipher
- **Transposition Ciphers**



XOR

- **XOR = “Exclusive OR”**
 - Recall: bitwise XOR
- **Easily reversed: XOR is its own inverse!**
 - To encrypt, XOR (Plaintext, Key)
 - To decrypt, XOR(Ciphertext, Key)
- **Very fast operation!**
- If key is same length as message and key is only used once (ephemeral), then XOR is a “perfect” mathematical cipher
- If key is shorter than message (\Rightarrow must be reused), risk of successful hostile decryption increases



(Simple) Obfuscation Methods

- **XOR**
- **Substitution Ciphers**
 - ROT13
 - Viginere Cipher
- **Transposition Ciphers**



Substitution Ciphers

- **Principle: Character by character substitution from plaintext into ciphertext**
- **Trivial example: “ROT13”**
 - **Alphabetical substitution: “rotate” 13 characters forward from plaintext character to get ciphertext character**
 - **Like XOR, it is its own inverse (26 letters in alphabet)**
 - **Apply once to encrypt**
 - **Apply a second time to decrypt**
 - **For illustration purposes only; not a useful cipher**



Substitution Ciphers

- **Viginere Cipher**
 - “Polyalphabetic” substitution
 - Use a “specific algorithm” to convert a single plaintext character to its ciphertext equivalent
 - Rearrange the alphabet being used for substitution before applying the “specific algorithm” to the next plaintext character
 - Increases processing complexity
 - Obscures repeated letters and makes frequency analysis much more difficult



(Simple) Obfuscation Methods

- **XOR**
- **Substitution Ciphers**
 - ROT13
 - Viginere Cipher
- **Transposition Ciphers**



Transposition Ciphers


- **Definition:**
 - Change the order of the characters, according to a specific algorithm
- **(Trivial) Examples**
 - Input: 0123 Output: 3210
 - Algorithm: Reverse the entire “message”
 - Input: 0123 Output: 2301
 - Algorithm: Rotate right two positions; wrap around
- Transposition concept may be incorporated as a step in a much more complex algorithm



6.2: Cryptographic Algorithms: Summary

- **Block Symmetric Algorithms (5)**
 - **Cipher Modes for BLOCK Symmetric Ciphers (4)**
- **Stream Symmetric Cipher (1)**
- **Asymmetric Algorithms (5)**
- **Hashing Algorithms (3)**
- **Key Stretching Algorithms (2)**
- **(Simple) Obfuscation Methods (3)**



- 
- **End of section 6.2 COPY ME before using!**
 - **THIS IS THE END OF THE WORK
I HAVE ACCOMPLISHED SO FAR**
 - **THIS CONCLUDES CHAPTER 27**



Wireless Security

Domain 6 Cryptography & PKI

Reference:

Drew Hamilton Lecture Notes

John Wu

Pinway Pang

Security+ Exam Guide, 5th ed.

Conklin, White, Cothren, Davis and Williams



Mississippi State University Center for Cyber Innovation



133

IEEE 802.11 Standards

802.11	The original WLAN Standard. Supports 1 Mbps to 2 Mbps.
802.11a	High speed WLAN standard for 5 Ghz band. Supports 54 Mbps.
802.11b	WLAN standard for 2.4 Ghz band. Supports 11 Mbps.
802.11e	Address quality of service requirements for all IEEE WLAN radio interfaces.
802.11f	Defines inter-access point communications to facilitate multiple vendor-distributed WLAN networks.
802.11g	Establishes an additional modulation technique for 2.4 Ghz band. Intended to provide speeds up to 54 Mbps. Includes much greater security.
802.11h	Defines the spectrum management of the 5 Ghz band for use in Europe and in Asia Pacific.
802.11i	Address the current security weaknesses for both authentication and encryption protocols. The standard encompasses 802.1X, TKIP, and AES protocols.



Original 802.11 Security

- **Service set identifier (SSID)**
 - A simple code that identifies the WLAN.
 - Clients must be configured with the correct SSID to access their WLAN.
- **Media access control (MAC)**
 - MAC address filtering restricts WLAN access to computers that are on a list you create for each access point on your WLAN.
- **Wired equivalent privacy (WEP)**
 - Encryption and authentication scheme that protects WLAN data streams between clients and access points (AP) This was discovered to have flaws.



WEP Flaws

- **Two basic flaws undermined its use for protection against other than the casual browser - eavesdropper**
 - **No defined method for encryption key refresh or distribution**
 - **Pre-shared keys were set once at installation and rarely if ever changed**
 - **Use of RC4 which was designed to be a one-time cipher not intended for multiple message use**
 - **But because the pre-shared key is rarely changed, same key used over and over**
 - **Attacker monitors traffic and finds enough examples to work out the plaintext from message context**
 - **With knowledge of the ciphertext and plaintext, can compute the key**



Encryption

- **WEP Flaw**
 - Takes about 10,000 packets to discover the key
 - Large amounts of known data is the fastest way of determining as many keystreams as possible
 - The information may be as innocuous as the fields in the protocol header or the DNS name query
 - Monitoring is passive so undetectable
 - Simple tools and instructions freely available to spit out the key
 - Legal experts postulate this type of monitoring may not be illegal



Other Problems

- **SSID (service set identifier)**
 - Identifies the 802.11 devices that belong to a Basic Service Set (BSS).
 - A BSS is analogous to a LAN segment in wired terms
 - SSID is meant as a method to identify what Service Set you want to communicate with; not as a security layer authentication
 - Even when using WEP, the SSID remains fully visible
 - Some mgfr even allow the WLAN cards to poll for the SSID and self configure



Other Problems

- **MAC (media access control)**
 - Possible to restrict access by MAC address on many AP (access points) by means of an ACL
 - All standards compliant NIC cards, including WLAN cards, should have unique MAC, some software allow this address to be 'spoofed'
- **Spoofing Wireless**
 - Is easy
 - Unlike internet devices which have routing issues to overcome, IP addresses of wireless devices can be manually changed at will
 - Some networks systems serve up the IP address dynamically



Improved Security Standards

- **802.1x Authentication (2001)**
- **WPA (Wi-Fi Protected Access) (2002)**
- **802.11i (2003-4)**



802.1X Authentication and EAP

- **802.1X**
 - Framework to control port access between devices, AP, and servers
- **Uses Extensible Authentication Protocol (EAP) (RFC 2284)**
 - Uses dynamic keys instead of the WEP authentication static key
 - Requires mutual authentication protocol
 - User's transmission must go thru WLAN AP to reach authentication server performing the authentication
 - Permits number of authentication methods
 - RADIUS is the market de facto standard



EAP Types

- **EAP-TLS (Transport Level Security) (RFC 2716)**
 - EAP is extension of PPP providing for additional authentication methods
 - TLS provides for mutual authentication and session key exchange
 - Negotiated mutual key becomes Master-Key for 802.11 TKIP
 - Requires client & server certificates (PKI based)
 - Deployed by Microsoft for its corporate network
 - Shipping in Windows 2000 and XP



Other EAP Types

- **EAP-TTLS**

- “Tunneled” TLS -- -- uses two TLS sessions

- Outer--TLS session with Server certificate for server authentication
 - Inner Inner--TLS session using certificates at both ends and password

- Protects user’ s identity from intermediary entities

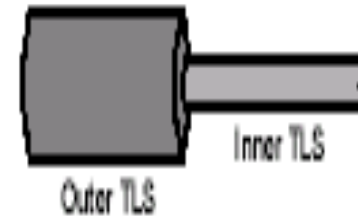
- **PEAP (Protected Extensible Authentication Protocol)**

- Similar to EAP-TTLS, but only allows EAP for authentication

- Server authentication via Server certificate

- User’ s password delivered through SSL protected channel
 - Session continues when user’ s password verified

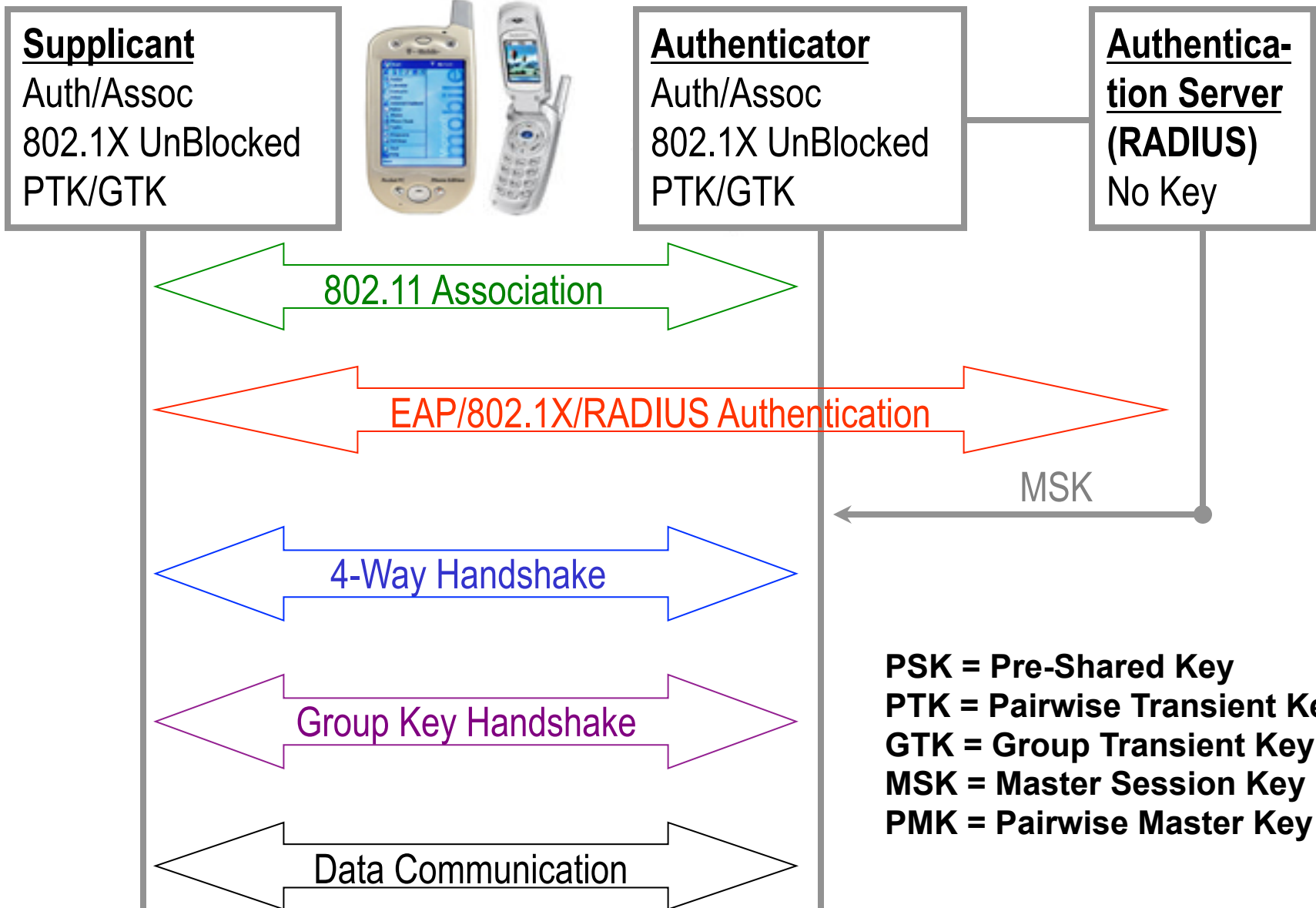
- Client-side certificate optional



WPA Interim 802.11 Security

- **Wi-Fi Protected Access (WPA)**
- **Interim Solution between WEP and 802.11i**
 - **Plugs holes in legacy 802.11 devices; typically requires firmware or driver upgrade, but not new hardware**
 - **Subset of the 802.11i and is forward compatible**
- **Sponsored by the Wi-Fi Alliance**
 - **Will require WPA for current certifications**
- **Support announced by Microsoft, Intel, others**





Wireless Encryption

- **WEP** – shared secret. Can be cracked in 3 to 30 sec
- **WPA** – uses RC4 w/ 128 bit keys. IV of 48 bits. Temporal Key Integrity Protocol (TKIP) providing different key per packet
- **WPA2** – AES instead of RC4. TKIP replace w/ Counter-Mode/CBC-MAC protocol (CCMP)
- **Extensible authentication protocol**
 - **EAP-TLS** – client and server mutually authenticate & use certs
 - **EAP-TTLS** – less secure than EAP-TLS
 - **EAP-PEAP** – encrypted tunnel but less secure than EAP-TLS



WPA

- **Improves WEP encryption**
- **Based on TKIP protocol and algorithm**
 - **Changes the way keys are derived**
 - **Refreshes keys more often**
 - **Adds message integrity control to prevent packet forgeries**
- **Benefits**
 - **Encryption weakness improved but not solved**
 - **Some concern that TKIP may degrade WLAN performance without hardware accelerator**
 - **But protects current device investment**
 - **Will be available sooner than 802.11i**



WiFi Protected Access (WPA)

- **Works similarly to 802.1X authentication**
 - Both Clients and AP must be WPA enabled for encryption to and from 802.1X EAP server
 - Key in a pass phrase (master key) in both client and AP
 - If pass phrase matches, then AP allows entry to the network
 - Pass phrase remains constant, but a new encryption key is generated for each session



WPA2

- **802.11i – WPA2**
 - **Full implementation**
 - **Adopted in September, 2004**
 - **Replaced WPA with WPA2-AES in 2004**
 - **Backwards compatible with WPA**
 - **Uses AES-CCMP**
 - **Advanced Encryption Standard – Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (Very Strong)**
 - **Provides RSN (Robust Security Network)**



Robust Security Network via 802.1X

- **PMK – Pairwise Master Key**
 - Sent from the AS to the Authenticator
 - Both the Supplicant and Authenticator now have the same PMK
 - PMK is permanent for the entire session
 - **Must generate a Pairwise Transient Key for encryption of data.**
 - Done using 4-way handshake



Robust Security Network via 802.1X

- **4-Way Handshake**
 - Confirm that the client holds the PMK.
 - Confirm that the PMK is correct and up-to-date.
 - Create pairwise transient key (PTK) from the PMK.
 - Install the pairwise encryption and integrity keys into IEEE 802.11.
 - Transport the group temporal key (GTK) and GTK sequence number from Authenticator to Supplicant and install the GTK and GTK sequence number in the STA and, if not already installed, in the AP.
 - Confirm the cipher suite selection.



802.11i

John Wu
Electrical and Computer Engineering Dept.
Auburn University, AL 36849-5201
wu@eng.auburn.edu



Mississippi State University Center for Cyber Innovation



152

Robust Security Network via 802.1X

- **Nonce**
 - A value that shall not be reused with a given key, including over all reinitializations of the system through all time.



Robust Security Network via 802.1X

- **PTK (Pairwise Transient Key – 64 bytes)**
 - 16 bytes of EAPOL-Key Confirmation Key (KCK)– Used to compute MIC on WPA EAPOL Key message
 - 16 bytes of EAPOL-Key Encryption Key (KEK) - AP uses this key to encrypt additional data sent (in the 'Key Data' field) to the client (for example, the RSN IE or the GTK)
 - 16 bytes of Temporal Key (TK) – Used to encrypt/decrypt Unicast data packets
 - 8 bytes of Michael MIC Authenticator Tx Key – Used to compute MIC on unicast data packets transmitted by the AP
 - 8 bytes of Michael MIC Authenticator Rx Key – Used to compute MIC on unicast data packets transmitted by the station
- **Last two only used when TKIP is used.**



WPA2-PSK

- **Pre-Shared Key Mode**
 - Network traffic encrypted using a 256 bit PMK
 - User enters key (Pairwise Master Key)
 - 64 hex digits
 - 8-63 Printable ASCII characters
 - Takes the passphrase, salts it with SSID of AP, then runs it through 4096 iterations of HMAC-SHA-1



TKIP

- **Temporal Key Integrity Protocol**
 - Quick fix to overcome the the reuse of encryption key problem with WEP
 - Combines the pre-shared key with the client' s MAC and and larger IV to ensure each client uses different key stream
 - Still uses WEP RC4, but changes temporal key every 10K packets
 - Mandates use of MIC (Michael) to prevent packet forgery
- **Benefits**
 - Uses existing device calculation capabilities to perform the encryption operations
 - Improves security, but is still only a short-term fix



TKIP Identification and Goals

- **TKIP: *Temporal Key Integrity Protocol***
- **Deploy as a software patch in already deployed equipment**
 - **Must conform to 1st generation Access Point MIP budget**
- **Short term only, to permit migration from existing equipment to more capable equipment without violating security constraints**
 - **Patch old equipment from WEP to TKIP first**
 - **Interoperate between patched and unpatched first generation equipment until all have been patched**
 - **Finally deploy new equipment**
- **Security Goals: Address all known WEP problems**
 - **Prevent Frame Forgeries**
 - **Prevent Replay**
 - **Correct WEP's mis-use of encryption**
 - **Never reuse keys**

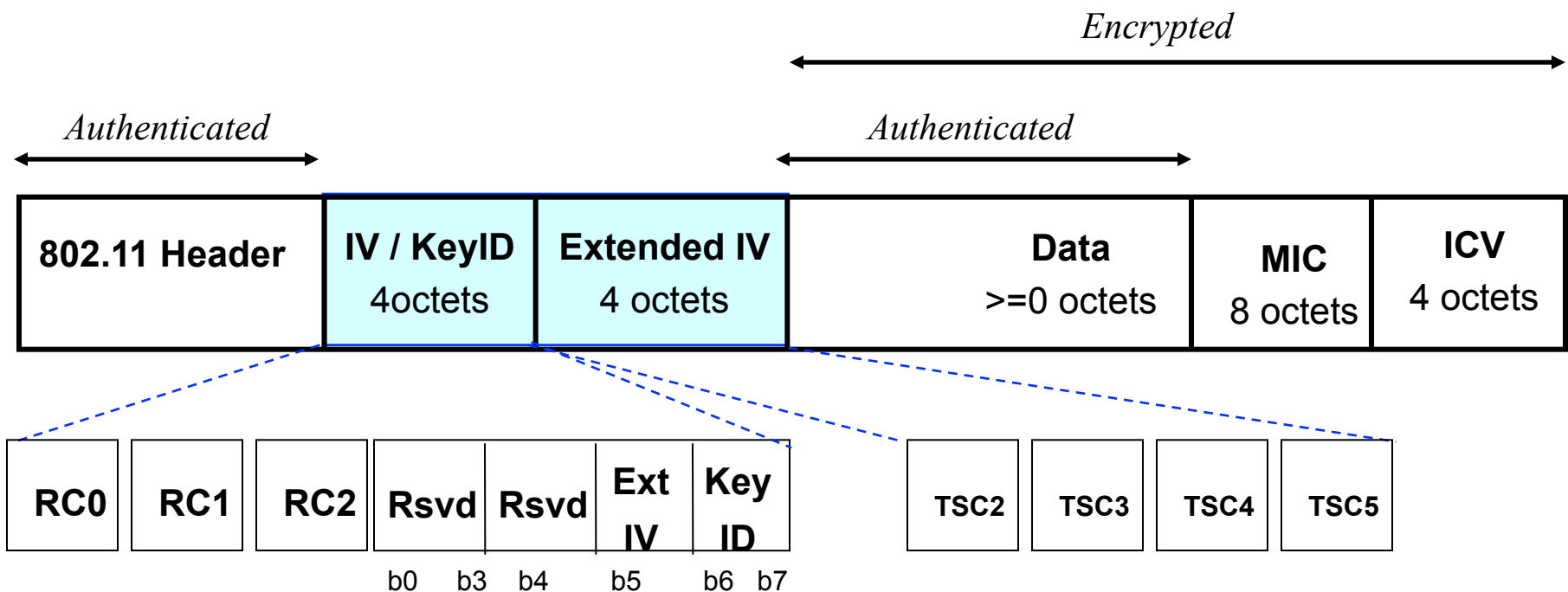


TKIP Overview

- **TKIP: Temporal Key Integrity Protocol**
- **Features**
 - **New Message Integrity Code (MIC) called Michael to detect forgery attempts**
 - **Since existing APs are MIP constrained, Michael cannot always provide desired level of assurance**
 - **Supplement Michael with Counter-measures, to increase forgery deterrence**
 - **Enforce frame order with a Replay protection mechanism**
 - **Extend WEP sequence space, to limit complexity of key renegotiation**
 - **Rescue WEP's mis-use of RC4 encryption that allows reused WEP hardware, because environment is so MIP constrained.**
 - **Make operation visible through appropriate counters**
 - **Under WEP it was infeasible to detect when you were under attack**
- **Meets goal of field upgradeable WEP fix**



TKIP Design (1) – MPDU Format s



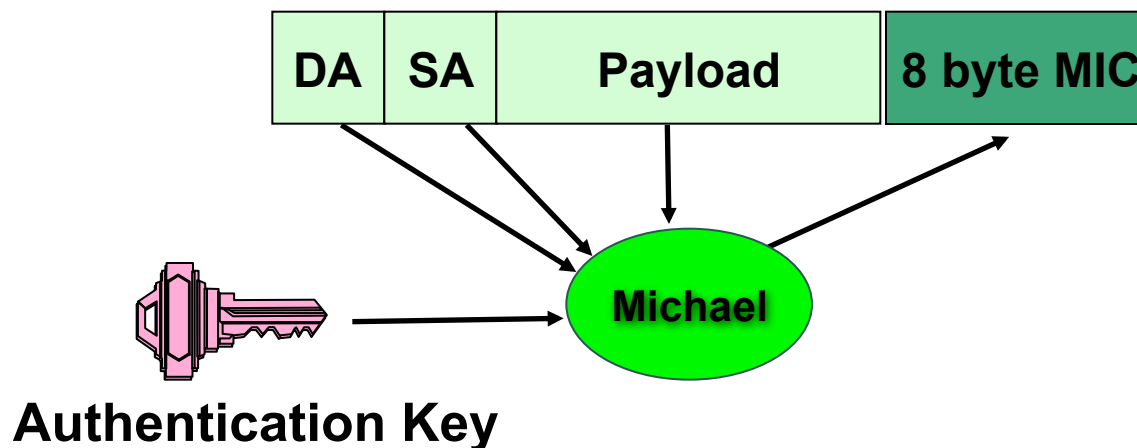
TKIP Design (2) – Keys

- **1 128 bit encryption key**
 - **Constrain forced by some WEP off-load hardware**
 - **So somehow must prevent key reuse**
- **2 64-bit data integrity keys**
 - **AP and STA each use a different key for transmit**



TKIP Design (3) -- Michael

- *Protect against forgeries*
- Must be cheap: CPU budget ≤ 5 instructions/byte
- Unfortunately is weak: a 2^{29} message differential attack exists
- Computed over MSDUs (MAC Service Data Unit), while WEP operates on MPDUs (Message Protocol Data Unit)
- Uses two 64-bit keys, one in each link direction



TKIP Design (4) – Countermeasures

- Check CRC, ICV, and IV before verifying MIC
 - Minimizes chances of false positives
 - If MIC failure, almost certain active attack underway
- If an active attack is detected:
 - Stop using session keys
 - Rate limit key generation to 1 per minute
- Why 1 Minute?
 - Michael design goal is 20 bits of security
 - But best attack we know is 2^{29}
 - Need to rate limit how fast attacker can generate forgery attempts
 - Since infeasible to rate limit attacker, instead rate limit attacker's effective attempts, i.e., how many WLAN will respond to
 - 1 year $\approx 2^{19}$ seconds
 - If design meets its design goal, this means on average at most 1 successful forgery per year
 - If the 2^{29} is best attack, then 1 successful forgery every 500 years



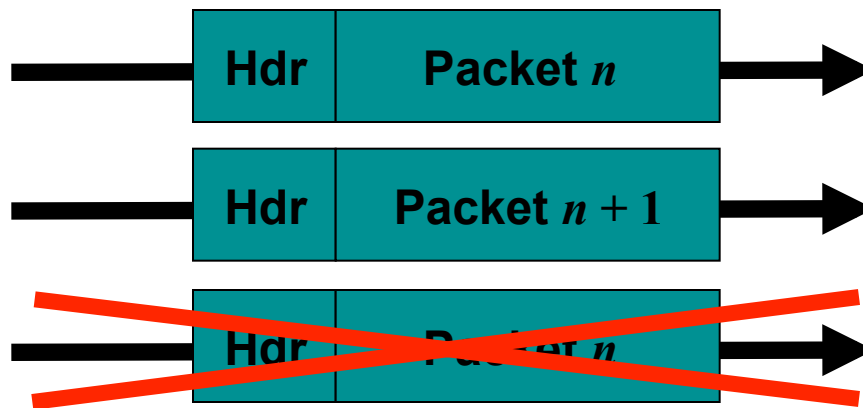
TKIP Design (5) – Replay Protection

Protect against replay

- reset packet sequence # to 0 on rekey
- increment sequence # by 1 on each packet
- drop any packet received out of sequence
- work with 802.11e QoS: QoS intentionally reorders packets
- Within each QoS Traffic Class:



Wireless
Station



Access Point



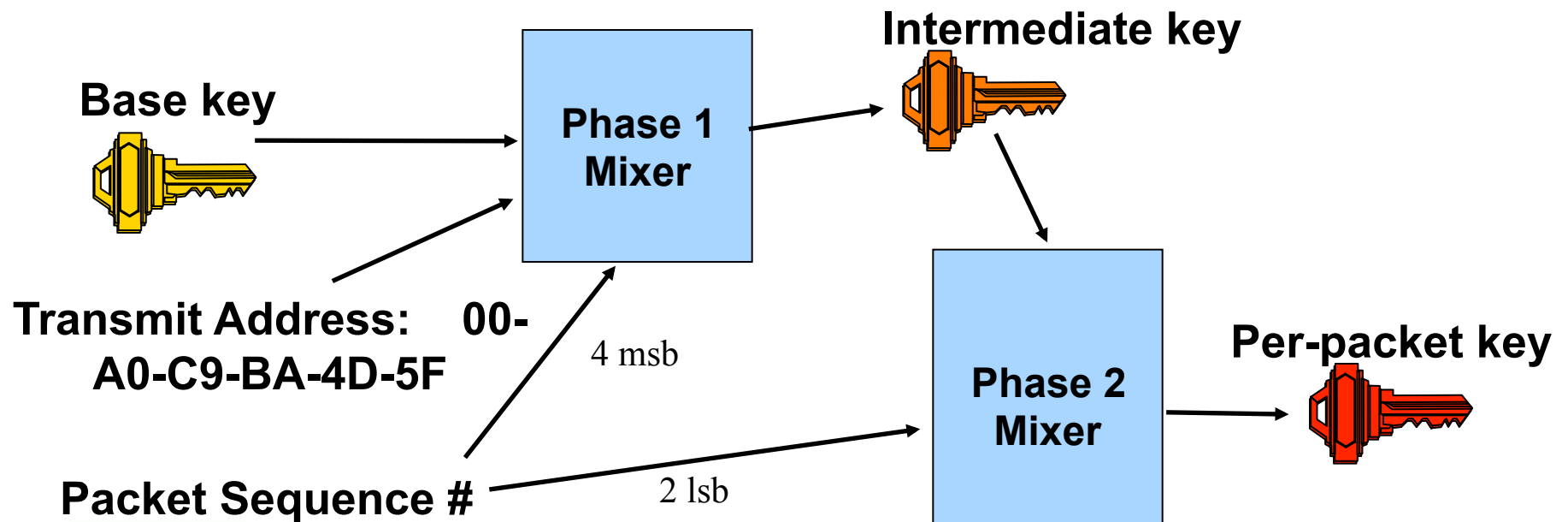
TKIP Replay Discussion

- **Sequence numbers for different MPDUs (fragments) of same MSDU must be sequential, or fragmentation attacks enabled**



TKIP Design (6) – Key Mixing

- *Stop WEP's encryption abuse*
- Build a better per-packet encryption key...
- ... by preventing weak-key attacks and decorrelating WEP IV and per-packet key
- must be efficient on existing hardware



TKIP Security Discussion

- **Michael transforms forgery attacks into less harmful denial of service attacks**
 - Differential cryptanalysis shows that an attacker can produce valid MIC in roughly 2^{29} tries by random guessing
 - Counter-measures added to rate limit effect of forgery attack
 - Encrypt the MIC, to limit knowledge attacker gains from either a successful or unsuccessful forgeries
- **Replay mechanism detects and discards replay**
- **Key mixing recovers WEP hardware by eliminating encryption abuse**
 - Auto-correlation analysis shows that keys produced by key mixing are correlated for sequence numbers n and $n+65536$
 - But we know of no other vulnerabilities and no way to exploit this
- **Mixing Transmit address defends against address hijacking and key reuse**



TKIP Summary

- **TKIP appears to provide weak but genuine security**
 - External review by Ron Rivest, David Wagner, John Kelsey, Susan Langford, and others
- **TKIP meets goal of software deployment on almost all existing equipment**
 - Does not appear to significantly degrade performance over WEP
 - Meets market's requirement for a migration path based on pre-existing installed base
- **TKIP is interoperable**
 - Interoperability demonstrated through the standard Wi-Fi test suite
- **Attacks become visible through TKIP counters and counter-measure invocation**



AES-CCMP Identification and Goals

- **AES-CCMP: 128 bit *AES* in Counter Mode with *CBC-MAC* Protocol**
- **All new design with few concessions to WEP**
 - **Costs \approx 40 instructions/byte in software, so requires new Access Point hardware**
- **Long term solution**
 - **Apply lessons learned from IPsec and 802.10 designs**
 - **Base on state-of-the art crypto**
 - **Extensible, to allow reconfiguration with any other 128 bit block cipher**
 - **Forward compatibility required with all 802.11 amendments, both planned and under development**
- **Security Goals: Address all known WEP problems**
 - **Prevent Frame Forgeries**
 - **Prevent Replay**
 - **Correct WEP's mis-use of encryption**
 - **Never reuse keys**



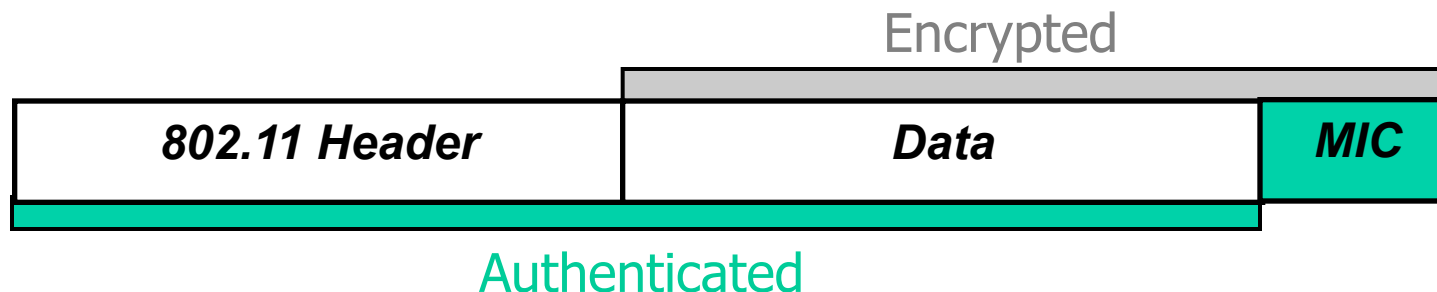
Counter Mode with CBC-MAC

- **Authenticated Encryption combining Counter (CTR) mode and CBC-MAC, using a single key**
 - CCM mode assumes 128 bit block cipher
 - IEEE Std 802.11i uses AES
- **Designed for IEEE Std 802.11i**
 - By D. Whiting, N. Ferguson, and R. Housley
 - Intended only for packet environment
 - No attempt to accommodate streams

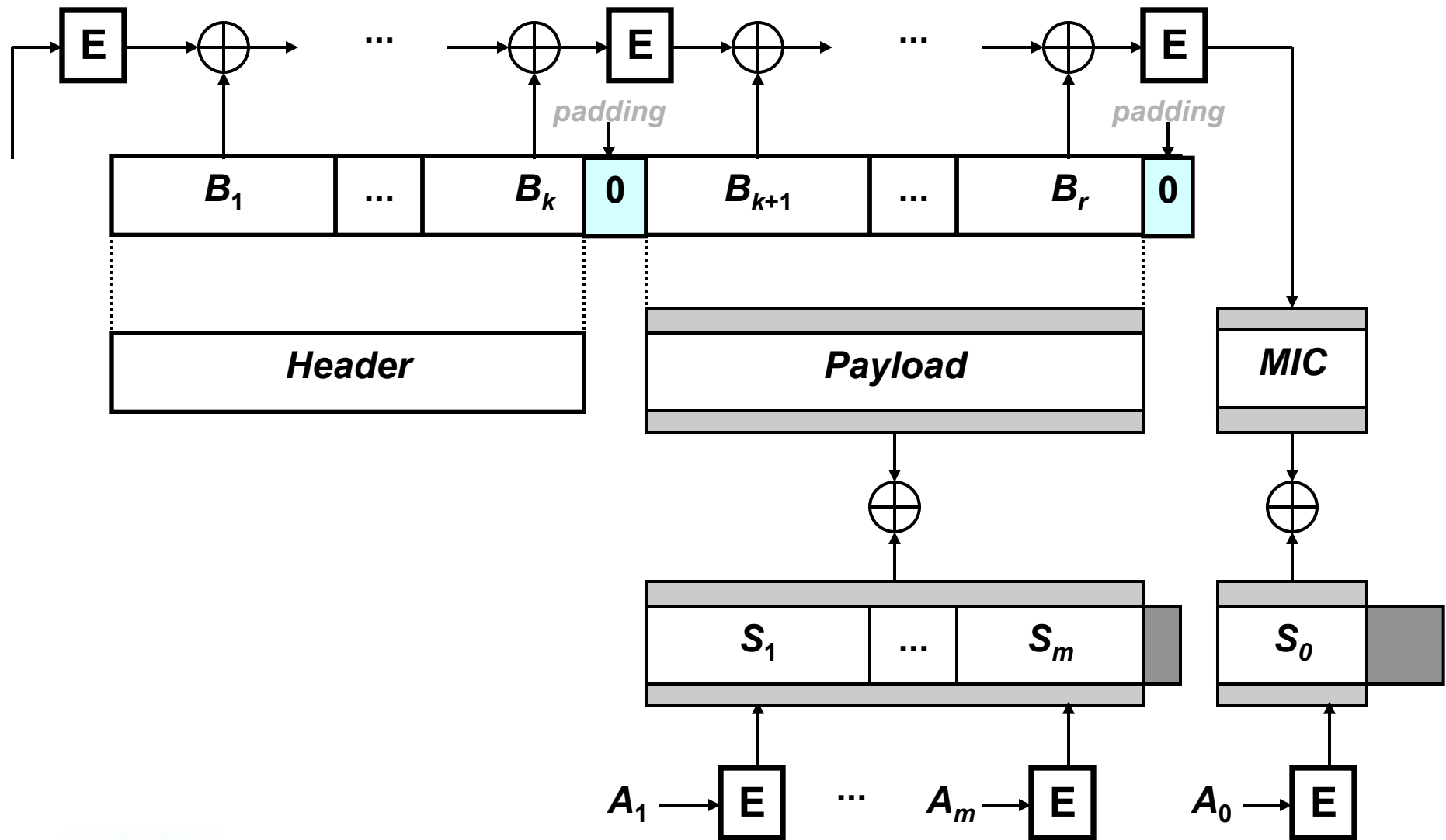


CCMP Overview

- Use CBC-MAC to compute a MIC on the plaintext header, length of the plaintext header, and the payload
- Use CTR mode to encrypt the payload
 - Counter values 1, 2, 3, ...
- Use CTR mode to encrypt the MIC
 - Counter value 0



CCM Mode

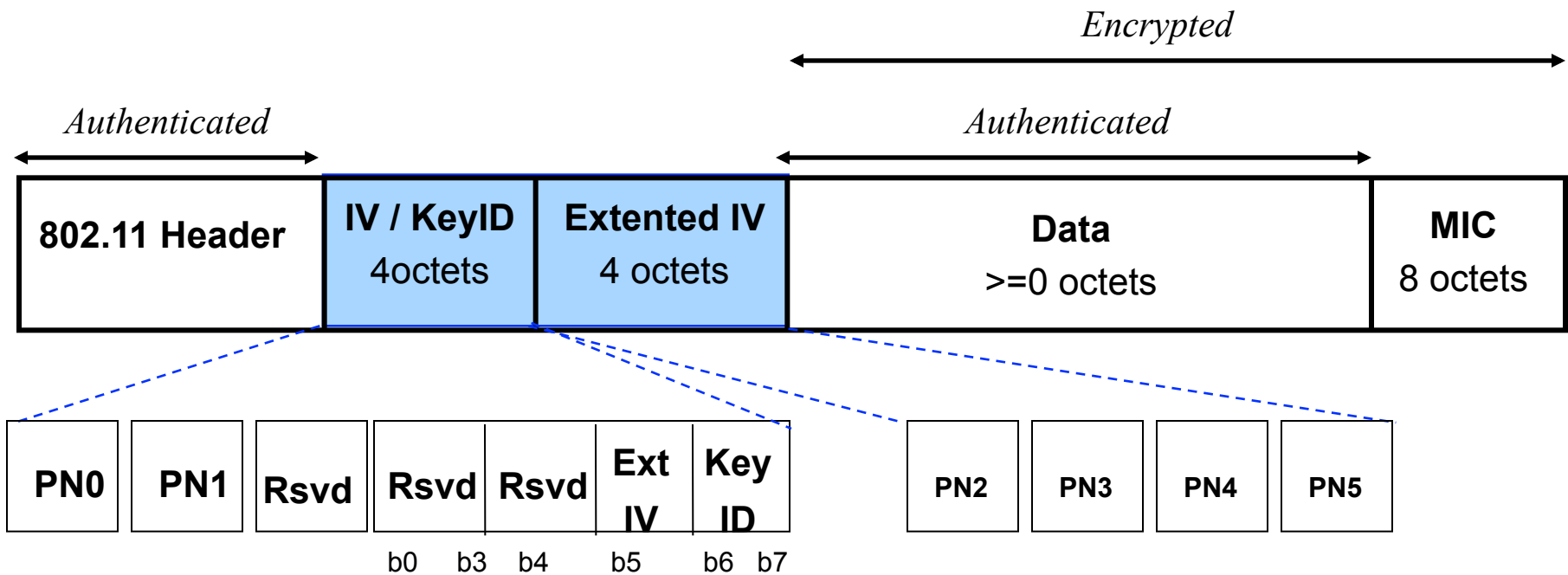


CCM Properties

- **CTR + CBC-MAC (CCM) based on a block cipher**
- **CCM provides authenticity and privacy**
 - **A CBC-MAC of the plaintext is appended to the plaintext to form an *encoded* plaintext**
 - **The encoded plaintext is encrypted in CTR mode**
- **CCM is packet oriented**
- **CCM can leave any number of initial blocks of the plaintext unencrypted**
- **CCM has a security level as good as other proposed combined modes of operation, including OCB**
 - **Danish cryptographer Jakob Jonsson proved CCM is secure if block cipher is secure – EUROCRYPT 2002**



CCMP MPDU Format



CCM Usage by CCMP

- Needs one fresh 128-bit key
 - Same 128-bit Temporal key used by both AP and STA
 - CBC-MAC IV, CTR constructions make this valid
- Nonce (A_0 , B_0) construction in CCMP's use of CCM:
 - $A_0 = \text{Tag}_0 \parallel 0x00 \parallel \text{Transmit-Address} \parallel \text{Frame-Sequence-Number}$
 - $B_0 = \text{Tag}_1 \parallel 0x00 \parallel \text{Transmit-Address} \parallel \text{Frame-Sequence-Number}$
 - Transmit-address is 6 octets
 - Frame-Sequence-Number is 8 octets and includes the QoS Priority
 - Sequence-Number must be sequential within a single MSDU
- 802.11 Header bits manipulated by normal protocol operation set to 0 prior to application of AES-CCM
- Sequence numbers must be sequential within MPDUs from same MSDU



AES-CCMP Summary

- **AES-CCMP appears to meet all 802.11i security goals**
 - External review by Ron Rivest, David Wagner, Phil Rogaway, and others
- **AES-CCMP is interoperable**
 - Interoperability demonstrated through the standard Wi-Fi test suite
- **AES can be replaced with any other secure 128 bit Cipher**
- **No known intellectual property encumbrances**
- **Reports attacks through counters**



Data Protection Protocol Comparison

	<u>WEP</u>	<u>TKIP</u>	<u>CCMP</u>
<i>Cipher</i>	RC4	RC4	AES
<i>Key Size</i>	40 or 104 bits	128 bits encryption, 64 bit auth	128 bits
<i>Key Life</i>	24-bit IV, wrap	48-bit IV	48-bit IV
<i>Packet Key</i>	Concat.	Mixing Fnc	Not Needed
<i>Integrity</i>			
<i>Data</i>	CRC-32	Michael	CCM
<i>Header</i>	None	Michael	CCM
<i>Replay</i>	None	Use IV	Use IV
<i>Key Mgmt</i>	None	802.11i 4-Way Handshake	802.11i 4-Way Handshake



WPA2 Key Re-Installation Attacks

- **Assigned CVE identifiers**
- **The following Common Vulnerabilities and Exposures (CVE) identifiers were assigned to track which products are affected by specific instantiations of the key reinstallation attack:**
 - **CVE-2017-13077: Reinstallation of the pairwise encryption key (PTK-TK) in the 4-way handshake.**
 - **CVE-2017-13078: Reinstallation of the group key (GTK) in the 4-way handshake.**
 - **CVE-2017-13079: Reinstallation of the integrity group key (IGTK) in the 4-way handshake.**
 - **CVE-2017-13080: Reinstallation of the group key (GTK) in the group key handshake.**
 - **CVE-2017-13081: Reinstallation of the integrity group key (IGTK) in the group key handshake.**
 - **CVE-2017-13082: Accepting a retransmitted Fast BSS Transition (FT) Reassociation Request and reinstalling the pairwise encryption key (PTK-TK) while processing it.**
 - **CVE-2017-13084: Reinstallation of the STK key in the PeerKey handshake.**
 - **CVE-2017-13086: reinstallation of the Tunneled Direct-Link Setup (TDLS) PeerKey (TPK) key in the TDLS handshake.**
 - **CVE-2017-13087: reinstallation of the group key (GTK) when processing a Wireless Network Management (WNM) Sleep Mode Response frame.**
 - **CVE-2017-13088: reinstallation of the integrity group key (IGTK) when processing a Wireless Network Management (WNM) Sleep Mode Response frame.**



Reference

- **Client Chaplin, Emily Qi, Henry Ptasinski, Jesse Walker and Sheung Li, “802.11i overview,” doc.: IEEE 802.11-04/0123r1, available from http://www.ieee802.org/16/liaison/docs/80211-05_0123r1.pdf**



Wireless VPNs

John Wu
Drew Hamilton



Mississippi State University Center for Cyber Innovation



179

Security Issues

- **Eavesdropping**
- **Modification of packets in transit**
- **Identity spoofing**
- **Denial of service**
- **Packet replay**



Security Solutions

- **IPSec-based Virtual Private Networks**
 - Open Standard
 - Public-key schemes have been theoretically proven secure by formal analysis
 - Well tested by the general public
- **Harris SecNet: Secure Wireless Local Area Network (SWLAN)**
 - NSA-certification of TS and below is expected Fall 2006
 - Controlled Cryptographic Item (CCI)
 - Limited testing



IPSec VPN

- **IPSec aims to provide a framework of open standards for secure communications over IP**
 - **Protect every protocol running on top of IPv4 and IPv6**
- **Data of this project is protected by**
 - **Encryption: AES-256 for top secret operations**
 - **SHA-1 (FIPS 180-2): for authentication of data and origin**



IPSec: Network Layer Security

IPSec = AH + ESP + IPcomp + IKE

Protection for IP traffic
AH provides integrity and origin authentication
ESP provides confidentiality and integrity

Compression

Sets up keys and algorithms for AH and ESP

ESP (encapsulated security payload): is used in this project



IPSec Components

- **ESP relies on an existing security association**
 - Idea: parties must share a set of secret keys and agree on each other's IP addresses and crypto algorithms
- **Internet Key Exchange (IKE)**
 - Goal: establish security association for AH and ESP
 - If IKE is broken, AH and ESP provide no protection!



Secure Key Establishment

- **Goal: generate and agree on a session key using some public initial information**
- **What properties are needed?**
 - **Authentication (know identity of other party)**
 - **Secrecy (generated key not known to any others)**
 - **Forward secrecy (compromise of one session key does not compromise keys in other sessions)**
 - **Prevent replay of old key material**
 - **Prevent denial of service**
 - **Protect identities from eavesdroppers**



Key Management in IPSec

- **Manual key management**
 - Keys and parameters of crypto algorithms exchanged offline (e.g., by phone), security associations established by hand
- **Pre-shared symmetric keys**
 - New session key derived for each session by hashing pre-shared key with session-specific nonces (number used once)
 - Standard symmetric-key authentication and encryption
- **Online key establishment**
 - Internet Key Exchange (IKE) protocol
 - Use Diffie-Hellman to derive shared symmetric key



IKE & IPsec Result

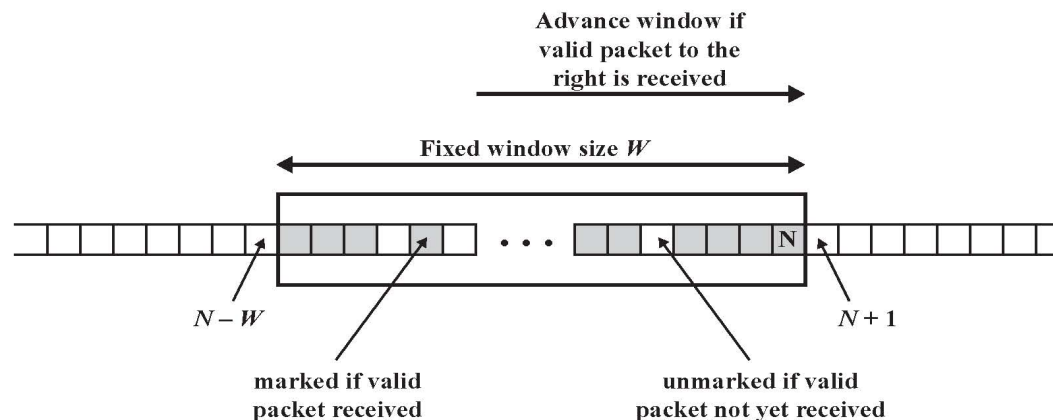
```
left:/usr/src # ipsec auto --statusall
000 interface lo/lo ::1:500
000 interface lo/lo 127.0.0.1:500
000 interface eth0/eth0 192.168.1.5:500
000 %myid = (none)
000 debug none
000
000 "host-host": 192.168.1.5[@left.auburn.edu]...192.168.1.6[@right.auburn.edu]; erouted; eroute owner: #3
000 "host-host":   ike_life: 10800s; ipsec_life: 3600s; rekey_margin: 540s; rekey_fuzz: 100%; keyingtries: 3
000 "host-host":   policy: RSASIG+ENCRYPT+TUNNEL+PFS+UP; prio: 32,32; interface: eth0;
000 "host-host":   newest ISAKMP SA: #1; newest IPsec SA: #3;
000 "host-host":   IKE algorithms wanted: 7_256-6-18,
000 "host-host":   IKE algorithms found: 7_256-6_512-18,
000 "host-host":   IKE algorithm newest: AES_CBC_256-SHA2_512-MODP8192
000 "host-host":   ESP algorithms wanted: 12_256-5,
000 "host-host":   ESP algorithms loaded: 12_256-5_256,
000 "host-host":   ESP algorithm newest: AES_256-HMAC_SHA2_256; pfsgroup=<Phase1>
000
000 #3: "host-host" STATE_QUICK_I2 (sent QI2, IPsec SA established); EVENT_SA_REPLACE in 2574s; newest IPSEC; eroute owner
000 #3: "host-host" esp.c0c4d88e@192.168.1.6 (0 bytes) esp.d8db2295@192.168.1.5 (0 bytes); tunnel
000 #2: "host-host" STATE_QUICK_R2 (IPsec SA established); EVENT_SA_REPLACE in 3318s
000 #2: "host-host" esp.af8342b7@192.168.1.6 (0 bytes) esp.1e03205c@192.168.1.5 (0 bytes); tunnel
000 #1: "host-host" STATE_MAIN_R3 (sent MR3, ISAKMP SA established); EVENT_SA_REPLACE in 10517s; newest ISAKMP
000
left:/usr/src # █
```

- Screen Shot from Dr. Wu's lab



Prevention of Replay Attacks

- When SA is established, sender initializes 32-bit counter to 0, increments by 1 for each packet
 - If wraps around $2^{32}-1$, new SA must be established
- Recipient maintains a sliding 64-bit window
 - If a packet with high sequence number is received, do not advance window until packet is authenticated



Attacks to Pre-shared Key

- Crack pre-shared key by brute force dictionary attack
- Free attacking tools:
 - IKECrack: <http://sourceforge.net/projects/ikecrack/>
 - Cain: <http://www.oxid.it/cain.html>
 - IKEProbe: <http://www.ernw.de/download/ikeprobe.zip>
 - IKE-scan: <http://www.nta-monitor.com/ike-scan/>
 - FakeIKeD: <http://linux.softpedia.com/get/Security/FakeIKeD-7926.shtml>.
- Solution:
 - Do not use pre-shared key and IKE aggressive mode
 - Use Public-key encryption or signature

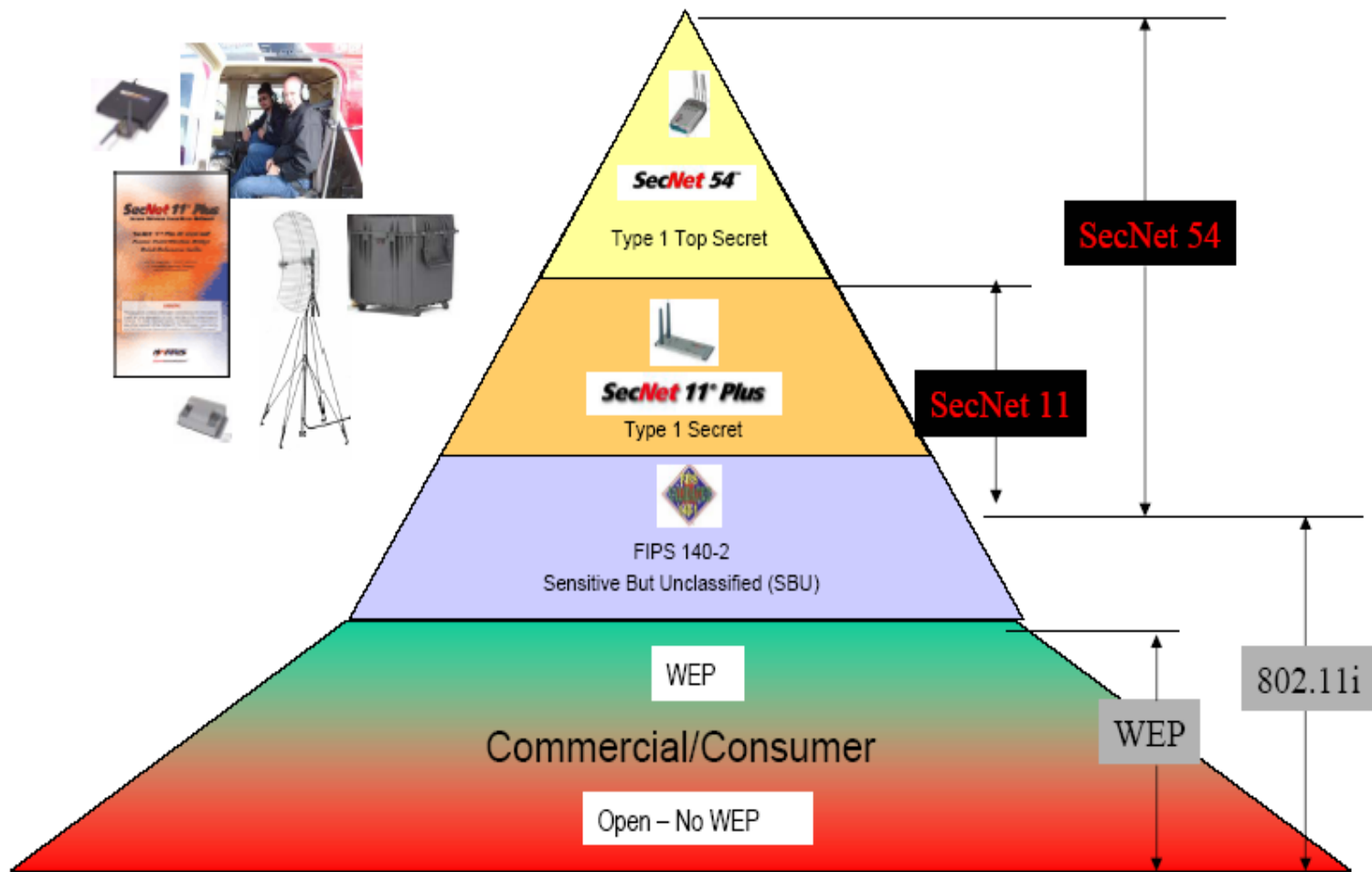


Denial of Service (DoS) Attack

- **IKEv1 suffers from DoS attacks**
 - Although cookies are used, the responder keeps state from the first message for crypto algorithms
 - Not stateless
- **IKEv2 prevents DoS attacks**
 - Ratified in December 2005
 - Use cookies
 - An optional additional exchange to request the feature in order for responder to ensure that the IKEv2 initiator can receive at the IP address it is claiming to be coming from before responder devotes state and significant computation to the exchange



Harris SecNet



Harris SecNet 54

- **HAIPE (High Assurance Internet Protocol Encryptor) is the United States Department of Defense's analog of IPsec**
- **inter-operate with a HAIPIS 1.3.5 compliant Inline Network Encryptor (INE)**
- **Sierra I cryptographic processor**
- **Pre-placed key - (NSA) large numbers of keys (perhaps a year's supply) that are loaded into an encryption device allowing frequent key change without refill**
- **Supporting 802.11a/b/g links up to 54 Mbps**
- **Channels:**
 - **802.11a/b/g: 3 nonoverlapping**
 - **802.11a: 12 nonoverlapping**
- **Range (outdoor, with included antennas):**
 - **54 Mbps: 500 ft**
 - **12 Mbps: 2,000 ft**
 - **1 Mbps: 3,000 ft**



Encryption

- **BATON: a Type 1 encryption algorithm, used broadly throughout the U.S. government to secure all types of classified information**
- **Type 1 certification is a rigorous process that includes testing and formal analysis of**
 - **cryptographic security,**
 - **functional security tamper resistance**
 - **emissions security**
- **Use pre-placed symmetric keys if both devices have been properly manually configured**



BATON

- **BATON has**
 - **a 96-bit or 128-bit block size and**
 - **a 320-bit key.**
 - **160 bits of the key are checksum material**
 - **prevent attackers from presenting modified keys to the device**
 - **they do not affect the security of the algorithm itself but rather prevent unauthorized keys from being loaded if a BATON device ends up in the hands of an adversary.**
 - **CBC, ECB or CTR mode**
 - **IV is 192 bits long, including a checksum**
- **Its speed and parallelizability were major factors in its selection**
- **BATON is an NSA Type I encryption method, implementation details are not in the public domain**



Conclusions

- **Effectively and securely using COTS-based wireless equipment in a tactical environment requires additional research.**
 - **Field environments are very different than tactical environments**
 - **Limited number of off-the-shelf items that are Common Criteria approved**
 - **Non-trivial performance issues in when using COTS hardware with NSA-approved commercial crypto**



Public Key Infrastructure

Domain 6 Cryptography & PKI

Reference:

Drew Hamilton Lecture Notes

Security+ Exam Guide, 5th ed.

Conklin, White, Cothren, Davis and Williams



Some Asymmetric Crypto Algorithms

- **Diffie-Hellman (DH)**
 - Developed for use as a key exchange protocol, Diffie-Hellman is used in Secure Sockets Layer (SSL) and IPsec encryption.
 - DH vulnerable to man-in-the-middle attacks, however, if digital signatures not used.
- **Elliptic Curve Cryptosystem (ECC)**
 - This uses points on an elliptical curve in conjunction with logarithmic problems, for encryption and signatures.
 - It uses less processing power than other methods, making it a good choice for mobile devices.



More Asymmetric Crypto Algorithms

- **El Gamal**
 - Not based on prime number factoring,
 - Uses the solving of discrete logarithm problems for encryption and digital signatures.
- **RSA**
 - Achieves strong encryption through the use of two large prime numbers.
 - Factoring these numbers creates key sizes up to 4096 bits.
 - RSA can be used for encryption and digital signatures and is the modern de facto standard.



Public-Key Cryptography Principles

- **The use of two keys has consequences in: key distribution, confidentiality and authentication.**
- **The scheme has six ingredients**
 - **Plaintext**
 - **Encryption algorithm**
 - **Public key**
 - **Private key**
 - **Ciphertext**
 - **Decryption algorithm**

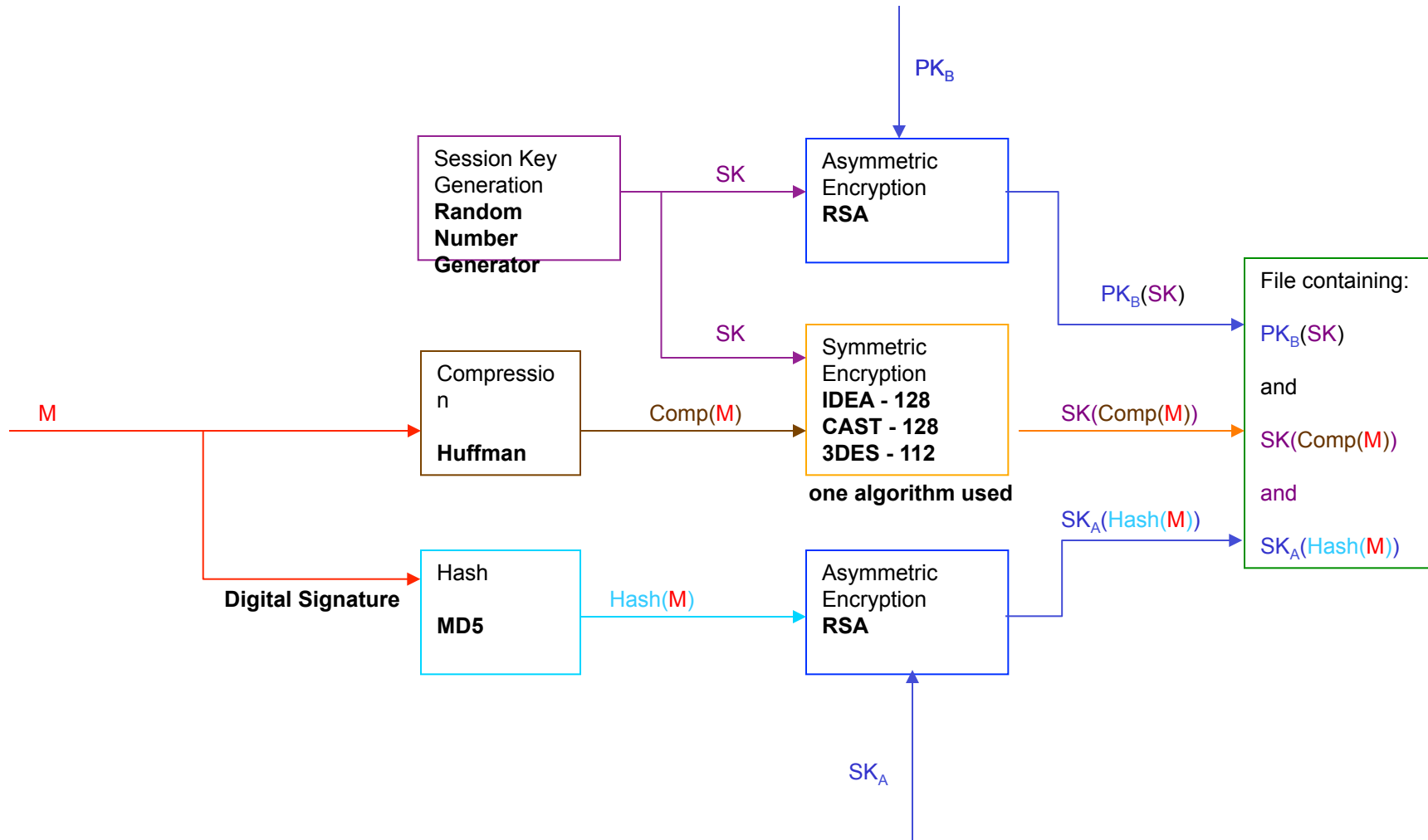


PGP

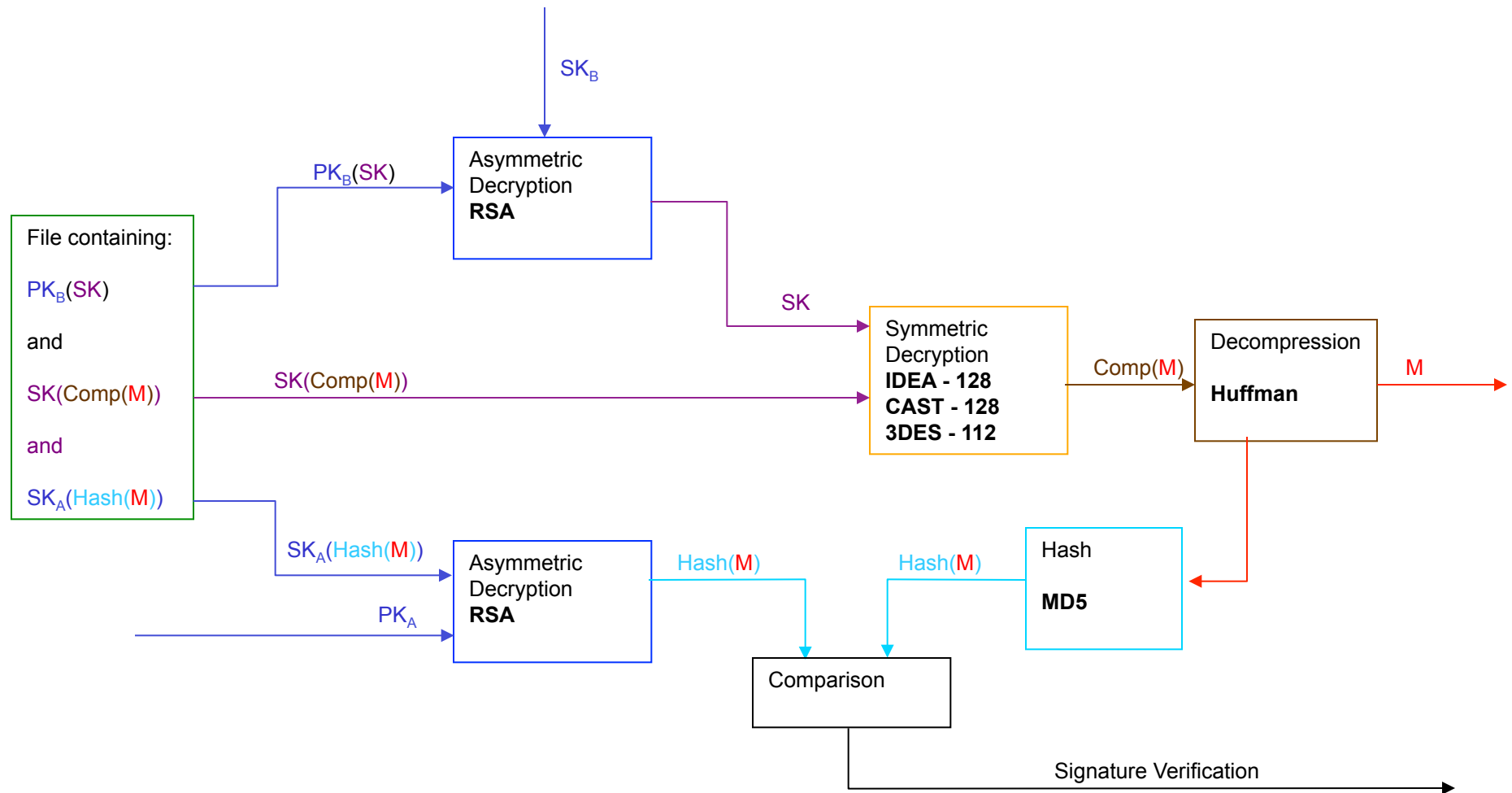
- **Freeware available from MIT (version 8) and Network Associates (version 8).**
- **Source code publicly available.**
- **PGP versions have been publicly available and tested since 1991.**
- **However, as with most reported results, insufficient information is provided to replicate the tests.**



PGP Sender



PGP Receiver



Endpoint Concerns

- **Physical Security**
- **Tempest**
- **Compromise (pass phrase, Session Key_a, secret keys)**
- **Public Key Tampering**
- **Viruses or Trojan Horses (PGP)**
- **Social Engineering**
- **Audio/Video Surveillance**
- **Key Management**



General Session Key Issues

- **Compromise of Keys (Stolen Keys)**
- **Destroying keys**
- **Seed Attacks**
 - **Seed stored on hard drive**
 - **Not truly random**
 - **short period**
 - **deterministic**
 - **Tested version of PGP:**
 - **changes seed for each session**
 - **new seed created from old seed and monitored keyboard activity**



Encryption Attacks

- **Symmetric Encryption**
 - **Brute force**
 - **Flaw in implementation**
- **Asymmetric Encryption**
 - **Brute force**
 - **Factoring (RSA)**
 - **Flaw in implementation**



Summary

- **6.1 Cryptographic Concepts**
- **6.2 Cryptographic Algorithms**
- **6.3 Wireless Security**
- **6.4 Public Key Infrastructure**

