


Cryptography Basics

An overview of cryptography used in trusted computing


Notes for CSC 495 / 680
September 30 – October 1, 2010



Slide 1

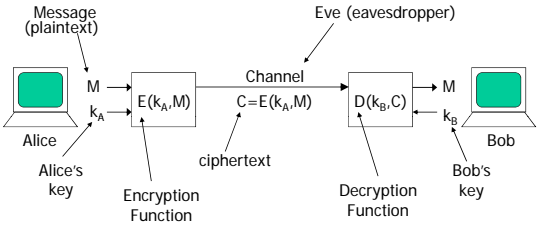
What You Should Learn

- Fundamental tools / building blocks:
 - Encryption algorithms/types
 - Digital signature techniques
 - Cryptographic hash functions
 - Message authentication codes
- Properties of algorithms used in TPMs
 - RSA encryption and signatures
 - SHA/1 hashing
 - HMAC
- Higher-level functionality
 - PKI and user credentials
 - Authentication




Slide 2

Confidentiality-oriented model




Goal: Eve can't learn anything about M



Slide 3

Basic Concepts

- Kirckhoff's Principle: Security depends only on strength of algorithm and secrecy of keys
 - **Not** secrecy of algorithm (security through obscurity)
 - Allows standardization of algorithms
 - TPMs: Specification/algorithms are public – keys protected
- What keys are secret?
 - Symmetric Algorithms: $k_A = k_B$ is secret
 - Asymmetric algorithms: $k_A \neq k_B$, and only k_B secret
 - Also called "public key algorithms"
 - Key Management is vital to maintaining security!
 - Business issues: Key generation, key certification, key escrow, rekeying, key revocation, ...



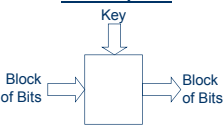
Slide 4

Symmetric Encryption

Brief Overview (TPMs do not use symmetric encryption)

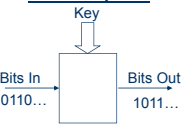
- Typically much faster than public key algorithms
 - Fast AES (symmetric): ~1.2 GBit/sec on Intel Core 2 6700 (1 thread)
 - Fast RSA (public key): ~247 kBit/sec on Intel Core 2 6700 (1 thread)
- Two main types of symmetric algorithms:

Block Ciphers




Blocks typically 64 or 128 bits.

Stream Ciphers




"Immediate": One bit in gives one bit out.



Slide 5

Symmetric Encryption – Cont'd

- Important parameters:
 - Block size (bs)
 - Key size (ks)
- Security:
 - Brute force attacks
 - "Keyspace": number of possible keys (2^{ks})
 - Brute force time proportional to keyspace size
 - Cryptanalysis attacks
 - Note: Keyspace for simple substitution = $26! \approx 10^{26}$
(approximately 10^{13} years to break at 10^6 trials/second)



Slide 6

Symmetric Encryption

Widely-used algorithms

Algorithm	Block Size (bits)	Key Size (bits)
DES	64	56
RC4	Stream	8 to 2048
Blowfish	64	32 to 448
AES	128	128, 192, or 256

Brute force keyspace search time:

Key Length (bits)	10 ⁶ tests per sec	10 ⁹ tests per sec	10 ¹² tests per sec
56	2283 years	2.28 years	20 hours
80	10 ¹³ years	10 ¹⁰ years	10 ⁷ years
128	10 ²⁶ years	10 ²³ years	10 ¹⁰ years

Cryptography Basics



Slide 7

How Big is a 128-bit Key?

- 2004 Estimate: \$100k machine breaks DES key in 6 hours
- What about a 128-bit key?
 - \$100k machine takes >10¹⁸ years [the earth is <10¹⁰ years old]
- What if we spent \$100,000,000,000?
 - Would take >10¹² years
- What about Moore's law saying that in 20 years machines will be about 16,000 times faster?
 - Would take >10⁸ years
- OK, what about in 40 years (machines 100 million times faster)?
 - Would still take >30,000 years
 - Do you really think Moore's law will last this long?
- What about improvements in cryptanalysis or super-duper quantum computers?
 - This could change everything....

Cryptography Basics



Slide 8

Public Key Algorithms

- Idea due to Diffie and Hellman in 1976
 - Maybe not the first! British declassified documents showing they were using this in the early 1970's!
- Different encryption and decryption keys
 - Decryption key difficult to compute from encryption key
 - Relationship between keys depends on secret knowledge ("trapdoor") known only to key generator
 - Public key can be widely published
- Security based on some explicitly-stated mathematical problem which is assumed hard
 - Note: Like all crypto, based on assumptions!

Cryptography Basics



Slide 9

Public Key Algorithms – Cont'd

- RSA (Named after Rivest, Shamir, Adelman):
 - First generally useful public key algorithm (1977)
 - Security based on assumed difficulty of factoring
 - Used in TPMs
- El Gamal
 - Based on original ideas of Diffie and Hellman
 - Security based on assumed difficulty of discrete log
- Elliptic Curve Cryptography (ECC)
 - Newer construct – El Gamal over different group
 - Seems to have comparable security to RSA and El Gamal with much fewer bits of key

Cryptography Basics



Slide 10

Public Key Algorithms – Cont'd

- Unlike symmetric algorithms, breaking related to well-defined mathematical problems
 - Breaking typically more efficient than brute force
- RSA "challenge key" of 768 bits factored
 - Completed December 2009
 - Took over two years using "many hundreds of machines"
- "Safe" key sizes:
 - RSA and El Gamal: 1024 bits OK for casual use - 2048 if paranoid
 - ECC: 160 bits (211 if paranoid)
- Warning: Algorithmic improvements very hard to predict!
 - Example: In 1977 Rivest predicted that a 129 digit (approx 430 bits) would take 40 quadrillion (40 x 10¹⁵) years to break – but broken in 1994!

Cryptography Basics



Slide 11

Public Key Algorithms – RSA

- RSA is the most widely-used public key algorithm
- Based on doing modular arithmetic ("mod n ")
 - Recall: "mod n " means remainder when divide by n
 - Examples:
 - $24+27 \text{ mod } 35 = 16$
 - $15 \cdot 10 \text{ mod } 35 = 10$
 - $2^6 \text{ mod } 35 = 29$
- In RSA, n is the product of two primes ($n=pq$)
 - 2048-bit RSA has p and q each about 1024 bits
 - These are huge numbers!
 - n is around 2^{2048} – remember how big 2^{128} is!?!?
 - Number of atoms in the observable universe is about 2^{265}

Cryptography Basics



Slide 12

Public Key Algorithms – RSA

- We don't need all the math behind RSA
 - It's really not that difficult, and available from many sources...
- **Fact:** If e is relatively prime to $(p-1)(q-1)$ then there exists a d such that:
 - For any M , if $C = M^e \bmod n$, then $M = C^d \bmod n$
 - So (e,n) is the public key, d is the private key
 - $E(M) = M^e \bmod n$ and $D(C) = C^d \bmod n$
 - There is an efficient algorithm to compute d from e, p , and q
 - Seems intractable to compute d from e and n without p and q
 - So... it seems that computing d is equivalent to factoring n
 - Not *known* to be equivalent
 - People have studied factoring for centuries – but no fast algorithms!

Cryptography Basics



Slide 13

Public Key Algorithms – RSA

- An example with small numbers: $n = 7 \cdot 11 = 77$
 - $(p-1)(q-1) = 6 \cdot 10 = 60$, and $e=7$ is relatively prime to $(p-1)(q-1)$
 - Corresponding d is 43
- An encryption/decryption example
 - $M=5 \rightarrow 5^7 \bmod 77 = 78125 \bmod 77 = 47$
 - $47^{43} \bmod 77 = 5$
 - Note that 47^{43} is a 71 digit number!!! Use efficient algorithms!
- Think about with RSA-size numbers (2048 bit key)
 - d is typically also 2048 bits
 - So C^d would be 2^{2059} bits?! Could you store this?

Cryptography Basics



Slide 14

Public Key Algorithms – RSA

- In practice...
 - Don't use simple RSA formula for encryption
 - Randomly pad in special ways to increase security
 - OAEP: Optimal Asymmetric Encryption Padding
 - PKCS #1: An older padding method
- Efficiency issues:
 - If n is chosen appropriately, can always use $e=3$ or $e=65537$
 - These values make encryption much faster
 - $M^{65537} \bmod n = M^{2^{16}+1} \bmod n = M \cdot (M^{16}) \bmod n$
 - This takes only 17 modular multiplications
 - Decryption (or using random e) requires on avg 3072 mod mults
 - TPMs use (by default) $e = 65537$

Cryptography Basics



Slide 15

Public Key + Symmetric

- Problem: Public key systems are powerful but slow, while symmetric systems are inflexible but fast
- Solution: A hybrid system!
 1. Sender generates random symmetric session key
 2. Sender encrypts session key using PK crypto
 3. Sender encrypts message using session key (and symmetric cipher)
- Result: A fast, flexible system

Cryptography Basics



Slide 16

Digital Signatures

- Remember: Only Bob (with secret key) can compute decrypt function
- Idea: Run plaintext through decryption function!
 - Property: Everyone can verify with the encryption function and public key!
 - Problem: plaintext and ciphertext domains might be different, making this impossible
 - Works for RSA where $E(k, D(k, P)) = P = D(k, E(k, P))$
- Some signature-specific algorithms (non encryption)
 - El Gamal signatures (related to El Gamal encryption, but a different algorithm)
 - DSA/DSS (Digital Signature Algorithm/Standard)
 - One advantage: Compact signatures (320 bits)

Cryptography Basics



Slide 17

Cryptographic Hash Functions

- Given an arbitrary-length message M , produces a fixed-length "message digest" $h(M)$
- Desired properties:
 - Given y , can't find an M such that $h(M)=y$ (one-way function)
 - Given M_1 , can't find an M_2 such that $h(M_1)=h(M_2)$
 - (Weak collision resistance)
 - Can't find any two M_1 and M_2 such that $h(M_1)=h(M_2)$
 - (Strong collision resistance)
- Widely used:
 - MD5 (128 bit digest) – as of August 2004: *Don't use this!* (doesn't exhibit strong collision resistance)
 - SHA1 (160 bit digest) – used in TPMs
 - SHA-256 (256 bit digest) – also SHA-384 and SHA-512

Cryptography Basics



Slide 18

Examples of Hash Function Usage

- File integrity (or file identification)
 - If a file changes, its hash should change
 - For accidental changes, checksums have done this
 - For malicious changes, need a cryptographic hash
 - Collision resistant properties says can't practically tamper and not disrupt hash value
 - However: if strong collision resistance isn't provided, then no guarantees (so don't use MD5)
 - Used by almost all anti-virus systems
 - Identification: Hash values are unique for real files, so there are databases (built in to many forensics tools) that identify files based on hash value
- Key derivation
 - People remember English phrases
 - Cryptographic functions need keys that are binary strings
 - A hash function does the mapping!

Cryptography Basics



Slide 19

Use of Hash Functions in TPMs

Passphrase to secret mapping

- "Authentication Secrets" (owner auth, key auth) are 160-bits
- Mapping of a string passphrase to 160-bit secret:

$$\text{secret} = \text{SHA1}(\text{passphrase})$$
- Simple, right? But... What is "passphrase"??
 - Infineon uses a UTF16LE (little endian) string
 - TSS says use UTF16BE (big endian) string
 - IBM TPM Tools use an ASCII string
 - Include or don't include null terminator when hashing?
- So: passphrase created using one software library might not work in applications using a different library...

Cryptography Basics



Slide 20

Use of Hash Functions in TPMs

Platform Configuration Registers (PCRs) for Integrity Measurement

- "Platform Configuration Registers" contain 160-bit measurements
 - 16 PCRs (PCR0 .. PCR15) can only be reset by system reset
 - Can only be extended with new values: $\text{PCR}_i \leftarrow \text{SHA1}(\text{PCR}_i || \text{newValue})$
- So:
 - System first extends PCR0 with initial BIOS code
 - BIOS extends this with measurement of POST code, platform extensions, ...
 - Measures other boot code (boot sector, OS, etc.) into other PCRs
- What can malicious code do?
 - Can't reset PCRs to load with faked values (hardware protection)
 - Can't extend PCR to get a faked value
 - Would require finding value such that $\text{DesiredPCR}_i = \text{SHA1}(\text{CurrentPCR}_i || \text{value})$ – impractical due to weak collision resistance

Cryptography Basics



Slide 21

Hash Functions + Signatures

- Problem: Don't want to run long message through (slow) digital signature algorithm
- Solution:
 1. Compute digest of message: $y = h(M)$
 2. Sign digest: $s = \text{sign}(k, y)$
 3. Transmit triple: (M, y, s) [y redundant, so optional]
- Key Property:
 - s is signature for any message with $h(M) = y$, but if hash function is secure, can't find such an M
 - Note: if strong collision avoidance fails, can find two M 's with the same hash (so same signature!)
 - So remember: Don't use MD5!

Cryptography Basics



Slide 22

Message Authentication Codes (MACs)

Keyed Hash Functions

- Idea: How to ensure a message comes from someone you share a key with?
 - Anyone can compute a hash (or re-compute)
 - Must use a secret key in a meaningful way
- Common techniques:
 - HMAC: MAC based on a hash function (TPMs use HMAC)
 - Can "plug in" any cryptographic hash
 - If one turns out to be weak, just replace with another
 - Using a symmetric cipher in a chained mode (like CBC)
 - Standardized in ANSI X9.9: "American National Standard for Financial Institution Message Authentication" (also ISO 8730)

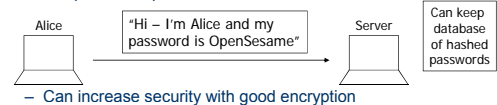
Cryptography Basics



Slide 23

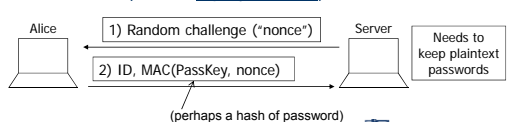
MACs for Authentication

- Basic (insecure):



- Can increase security with good encryption

- MAC-based (avoids replay attacks):



(perhaps a hash of password)



Slide 24

MACs in TPMs

- TPMs use HMAC with SHA1 to authenticate commands
- Example: Consider an owner-authenticated command
 - Along with command pass HMAC(ownerSecret, command)
 - So: Can be computed only w/knowledge of ownerSecret
 - And: Command can't be modified between owner and TPM
- Not quite this simple...
 - Not all fields of command are hashed
 - Combined with MAC from previous commands to chain commands

Cryptography Basics



Slide 25

Example from a TPM Command

- Sample command from spec (input data):

Offset	HMAC	Type	Name	Description
1	2	TPM_TAG	tag	TPM_TAG_NULL_AUTH_COMMAND
2	4	UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	TPM_COMMAND_CODE	cmdCode	Command code: TPM_CMD_LOAD_KEY
4	4	TPM_AUTH_HANDLE	authHandle	Handle of parent key
5	10	TPM_KEY	key	Including key structure, with encrypted private and one public portion, wrapped by TPM_TAG_KEY
6	4	TPM_AUTH_HANDLE	subAuthHandle	The authorization session handle used for parent-handle authorization
7	20	TPM_NONCE	nonce	Nonce previously generated by TPM to cover inputs
8	20	TPM_NONCE	nonceOut	Nonce generated by system associated with sub-handle
9	1	BOOL	certifyAuthSession	The continue use flag for the authorization session handle
9	20	TPM_AUTH_DATA	parentAuth	The authorization session digest for private and parent-handle HMAC key: parentHandle * paramSize



Slide 26

A Problem with Signatures

- What does a public-key signature verification tell you?
Verification parameters include public key, and successful verification says "Only someone holding the corresponding private key could have made this signature."
- What do you want a signature verification to tell you?
Probably something like "Joe Smith signed this."
- Problem: What assurance do you have that the public key really belongs to Joe Smith?



Slide 27

What is a Digital Certificate?

- Associates an identity/properties with a public key
 - Identity can be person's name, website, e-mail, ...
 - Properties can be valid key uses, age of individual, access rights granted, ...
- Signed by someone you trust
 - Signature is trusted party vouching for ID/key pair
 - Role is similar to a notary public
- Some typical properties of certificates:
 - Good for a set time (validity period)
 - Must get a new certificate after expiration
 - Certificates may be revoked



Slide 28

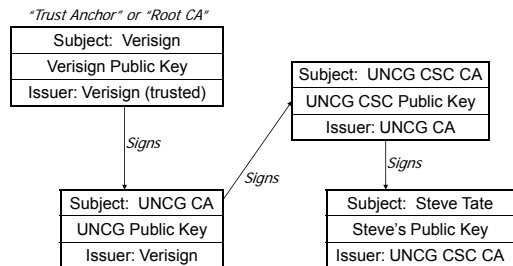
More on Certificates

- Common types of certificates:
 - X.509 standard (version 3)
 - PGP certificates
- Who signs certificates? Several possibilities:
 - Independent "Certification Authority" organization
 - Disinterested third party – company or government
 - Examples: Verisign, Deutsche Telekom, Entrust, AOL, ...
 - Internal (organizational) certification authority
 - Organization controls certificates for employees or clients
 - Could be just an individual you trust
 - This is how PGP certificates are typically certified



Slide 29

Certificate Chains



Slide 30

Uses of Certificates in a TPM

Certifying the Endorsement Key

- If TPM specification is open, how can we tell if we are interacting with a real TPM or a software simulator (or a “Trojan TPM”)?
- Each TPM contains a special “Endorsement Key” (EK)
 - Unique to each TPM
 - So if one is revealed, only that one TPM is compromised
 - Private part of EK never leaves TPM
 - Public part certified by manufacturer as a valid TPM-bound key
 - So: Manufacturers are CAs for integrity of TPM EKs

Cryptography Basics



Slide 31

Uses of Certificates in a TPM

Certifying Attestation Identity Keys (PrivacyCAs)

- Privacy Issue: Doesn't EK identify a system?
 - Yes, it does!
- Solution:
 - TPMs can create Attestation Identity Keys (AIKs)
 - Key is created in cooperation with a “PrivacyCA”
 - EK is used in transaction, and PrivacyCA verifies EK certificate
 - Resulting AIK is issued a certificate by PrivacyCA
 - Idea: Only a valid TPM can do this (uses EK), and an honest PrivacyCA will only certify properly-constructed keys
 - Problems:
 - PrivacyCA must be known and trusted by all
 - PrivacyCA can link all “pseudonyms” together by EK
 - In version 1.2: Direct Anonymous Attestation (DAA) has stronger privacy guarantees

Cryptography Basics



Slide 32