T-110.5211 Cryptosystems

Summary and review of lectures

4.12.2008

Outline

- Summary and review of lectures
  - Lecture 1 Basics
  - Lecture 2-7 Crypto
  - Lecture 8-9 Implementation issues
- Preparing for the exam

General about the course

- This is the fifth time the course was arranged
  - We need a course covering practical cryptographic topics
  - Security is becoming more important and cryptosystems are an important part of improving security
- It's important to improve the course for next year
  - We want the students to enjoy the course ! (Really :-)
  - The assignment should support learning in a useful way
- To do this, we appreciate constructive feedback
  - How can we make the course **more useful for you** ?
  - How could we make the topics **more concrete and motivating** ?
  - What topics would you skip, and what new ones would you add ?
  - => You can use **e-mail** or **fill in a form** during the exam; or
  - **http://www.cs.hut.fi/Opinnot/Palaute/kurssipalaute-en.html** [Dec 8 to Jan 7]

Course assignment

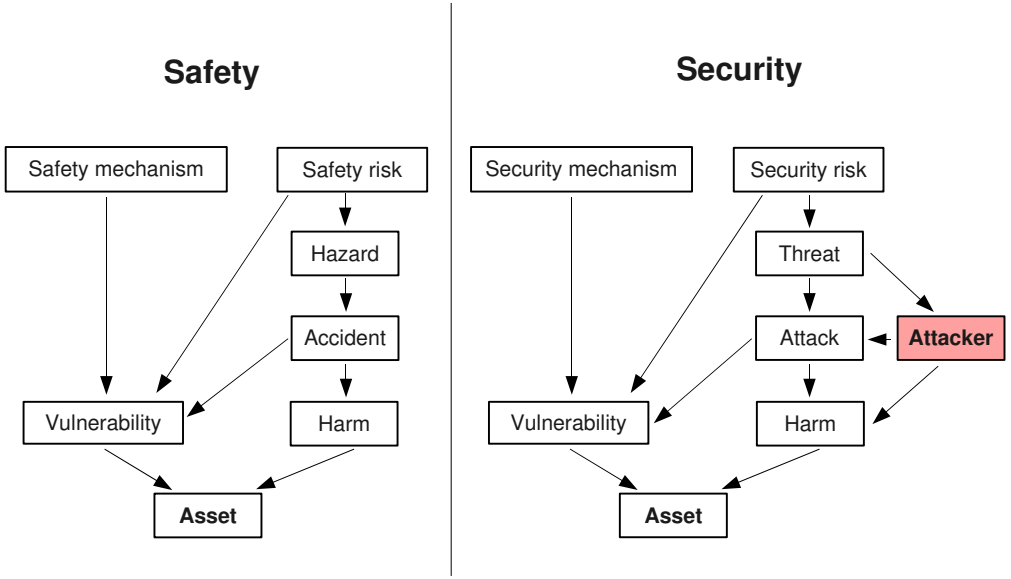
- Assignment structure
  - Weekly deliverables, “crypto library” => was this good?
  - Did you feel you learned from the assignment?
  - => Feedback very much appreciated, we'll improve next year
- Almost everyone passed
  - Quite many extras

Summary – Introduction

Course context

- Course scope
  - Basics of cryptography
  - Implementation issues
  - Practical software issues throughout SW life cycle
- Safety, security, and survivability
  - Quality factor <= Dependability <= Security
- Assets-based security model
  - Assets, vulnerabilities, threats, security mechanisms
  - Active attacker
  - Threat modeling – e.g. Dolev-Yao for crypto protocols

Safety and security compared



Assets can include people, financial assets, information, reputation, availability of resources, ...

Cryptosystems

- Creating secure systems
  - Specific methods at each step of software life cycle
  - Sound threat model => good requirements, refinement, correspondence
- Cryptology = cryptography + cryptanalysis
  - Cryptography: the study of mathematical techniques related to information security (e.g. encryption)
  - Cryptanalysis: the study of mathematical techniques to defeat cryptography
  - => Adversarial process in developing cryptographic primitives
- Cryptographic primitive
  - Mathematical entity which meets specified security requirements
  - Unkeyed, secret-key, public-key
  - Kerckhoff's principle – algorithms are public, key(s) are secret

Cryptosystems

- Cryptosystem (or cryptographic system)
  - **Provides** information security services (e.g. “CIA”)
  - **Through** a combination of cryptographic primitives, protocols, operational procedures, documentation, user training, etc...
- Cryptographic security concepts
  - Perfect secrecy
  - Computational security
  - Provable security
- Role of cryptosystem in the whole system
  - A tool in providing security (also safety, even survivability)
  - Decrease vulnerability to data communication threats
  - Decrease vulnerability to local threats
  - May also create new threats (e.g. loss of an important key)

Cryptosystems

- Attacks and failures of cryptosystems
  - Cryptosystem environment
  - Cryptosystem implementation (hardware or software)
  - Cryptographic protocol / cryptosystem (protocol failure)
  - Cryptographic primitive
- Implementation vulnerabilities very important
  - Typically caused by unsafe languages (C, C++)
  - The second half of the course deals with implementation issues
  - Software life cycle model used as a reference

Summary – Cryptography

Crypto mathematics

- Review of crypto mathematics
  - Binary arithmetic, finite groups & fields
  - One-way trapdoor functions, number theory and complexity theory
- Number theory basics
  - Primes, co-primes, gcd, lcm (Euclidean algorithm)
  - Modular arithmetic (mod n), modular inverses and exponentiation
  - Euler's totient function, Euler's theorem, Fermat's theorem
  - Generators and primitive roots (elements)
  - Chinese Remainder Theorem (CRT), primality testing
- Groups, rings, fields
  - Groups => rings and fields
  - Finite fields – **GF(p^n)**, example of a field used in AES

Ciphers

- Classical ciphers
  - Substitution (Caesar), permutation (reorganize letters), polyalphabetic
  - Running key, Hill cipher, one-time pad = theoretically fully secure cipher
- Modern cipher structure
  - Product ciphers – iterated permutation and substitution
  - Kerckhoff's principle – public algorithm, secret key(s)
- Basic symmetric / unkeyed primitives
  - Block ciphers and stream ciphers (symmetric = shared key)
  - Message Authentication Codes (MACs) (symmetric)
  - Hash functions (unkeyed)
- Threats – typically retrieve message or key (also forgery, etc)
  - Effective key length and brute force

Symmetric crypto

- Block ciphers
  - Plaintext encrypted in blocks (typically 64/128 bits), padding
  - Each block uses the same key
  - Mode of operation alters encryption process – ECB, CBC, CTR,..
  - Design – Feistel networks & S-P networks
  - Security – resistance against known attacks; mode of operation

Symmetric crypto

- Stream ciphers
  - Plaintext processed in small blocks (typically 1 or 8 bits), no padding
  - Keystream = separate key for each block
  - Trivial combinator (e.g. XOR) for keystream and plaintext
  - Keystream is security critical; initialized using non-repeating IV
  - Synchronous stream cipher vs. self-synchronizing stream cipher
  - Design – LFSRs + combinators
  - Security – high quality keystream, avoid IV repeat
- A block cipher can be converted into a stream cipher
  - E.g. use encrypted counter to generate keystream

Symmetric crypto

- Hash functions
  - “Fingerprint” or “compress” an arbitrary size message into a fixed size hash value
  - Often used in protocols for e.g. digital signatures, message identification, random number bias normalization, etc.
  - Design – Merkle-Damgård structure (e.g. MD5 and SHA1)
  - Security – preimage, second preimage, collision resistance
  - MD5 and SHA1 popular, but currently security in doubt
    - Known collisions and extension attacks
- MAC functions
  - Protect integrity of data – like hash but requires a key
  - Security – resist existential forgery (any new valid MAC)
  - HMAC and CBC-MAC typical, but more modern constructs exist

Symmetric crypto

- Symmetric ciphers
  - DES (3DES), Feistel structure
  - AES, S-P network, algebraic
  - Alleged RC4, simple stream cipher
- Hash functions and MACs
  - MD5, Merkle-Damgård
    - Pad to 512-bit blocks, rounds using 512-bit blocks, 128-bit output
  - SHA1 + SHA-{224,256,384,512}, Merkle-Damgård
    - 160-bit output
  - HMAC, uses e.g. MD5/SHA1 internally
  - Poly1305-AES MAC, example of a modern MAC

Asymmetric crypto

- Asymmetric cryptography
  - Early work classified, first public Diffie-Hellman and RSA (Merkle)
  - Encr. & decr. use separate (mathematically related) keys
    - Example benefit – public key can be distributed, so that multiple senders can send messages to you (but won't be able to decrypt)
  - Can be used for e.g. public key encryption and digital signatures
  - Requires **one-way trapdoor functions**
- Motivation for asymmetric crypto
  - Key management – **O(n²)** becomes **O(n)** (**n** = # parties)
  - Differentiates between capability to encrypt/decrypt, and sign/verify
  - Improved authentication properties (e.g. non-repudiability)
- Important early work (RSA, D-H), still very much in use

Asymmetric crypto

- Public key encryption
  - Public key operation (encr.), private key operation (decr.)
  - One-way trapdoor function
- Hybrid encryption
  - Because asymmetric encryption is slow, we typically use hybrid encryption
  - E.g. master key encrypted using RSA, actual data using AES
  - Prevailing model – IPsec, TLS, PGP, SSH, ...
- Key management (asymmetric crypto suits KM well)
  - Generate, store, protect, transfer, use, destroy keys
  - Key exchange protocols to generate symmetric keys
  - Certificates, various trust models

Asymmetric crypto

- Diffie-Hellman (g^{xy} (mod p))
  - Establish a shared secret without prior secrets, even with eavesdroppers
  - Requires additional authentication to prevent man-in-the-middle
  - Several security benefits – e.g. keys are independent
  - Security related to discrete logarithm (DLP)
  - Most key exchange protocols use D-H (e.g. TLS, IPsec)
- RSA
  - First public key encryption system, computation mod **n=pq**
  - Provides public key encryption and digital signatures
  - Message requires padding – security critical
  - Security related to factoring **n**
  - Most widely used public key encryption / signature system

Asymmetric crypto

- Digital signatures
  - Private key operation on plaintext data => signature
  - Typically applied over a hashed version of message M
  - Verification = public key operation + compare hash
- Public key certificates
  - Subject S associated with public key P + attributes (such as validity period), signed by a trusted party
  - Certificates can be chained => certificate hierarchy
  - Simplifies key and trust management, but complex in practice
  - Certificate Revocation List (CRL)
  - Certificate validation – path building, path validation; trust roots
- ASN.1 format used for X.509 certificates
  - Abstract notation for structured data + alternative concrete encodings

Some mathematical crypto protocols

- Secret sharing and splitting
  - Allows sharing of secret with zero information leak until enough participants decide to reproduce the secret
  - **(m,n)**-threshold scheme; Shamir's scheme, polynomial coeffs
- Bit commitment
  - Allows Alice to commit to a certain bit sequence without revealing it
  - E.g. two-phase symmetric protocol
- Zero-knowledge proofs
  - Proofs where other party becomes convinced but obtains no information
- Secure multiparty computation
  - Computing arbitrary functions of data held by participants
  - Participants only get final value, no data from other participants

Some real-world protocols

- Kerberos
  - Symmetric key distribution & authentication system
  - Requires a trusted third party – Kerberos servers
  - Successful authentication results in a **ticket**, which encapsulates service parameters, validity period, and the crypto key
- PGP
  - Signing, encryption, compressions, ASCII conversion (hybrid crypto)
- Hard drive encryption
  - Specific requirements, e.g. random access & escrow
  - E.g. block cipher => per-block CBC or CTR
- SSH2
  - MAC, encryption, compression, mutual auth., similarities with TLS

Transport Layer Security 1.0

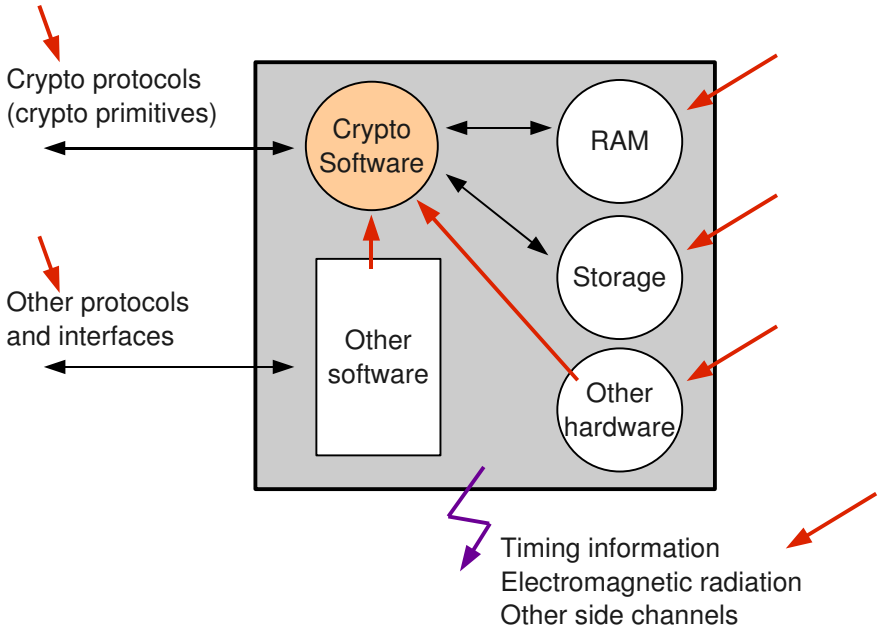
- Basics
  - Secure sockets model – one encrypted application stream
  - MAC, encryption, compression, mutual auth. using certificates
- Protocol layers
  - TLS Record Layer – symmetric crypto & compression
  - Protocols running on top of Record Layer
    - handshake, alert, change_cipher_sec = setup protocols
    - application = application data protocol
- Security
  - Diffie-Hellman or RSA key exchange
  - Mutual certificate auth., optional for both, typically server only
  - X.509 certificates for public key representation
  - Block and stream cipher encryption, HMAC-based MACs

Comparing TLS to IPsec

- Basic differences in application
  - TLS secures streams, IPsec secures packets
  - TLS relies on a transport layer, while IPsec works as part of IP
- Key exchange
  - Bulk encryption (ESP/AH) separated from key exchange (IKE)
  - IPsec (IKE) uses authenticated Diffie-Hellman, like TLS
  - RSA / DSA signatures, pre-shared key; legacy authentication
- IPsec bulk encryption
  - IPsec has to deal with packet loss – thus CBC IVs sent explicitly
  - IPsec packets are also numbered to deal with replays
  - Algorithms similar, e.g. 3DES + HMAC-SHA1-96 (truncated hash)
- Compression – no support in IPsec (IPcomp)

Summary – Implementation issues

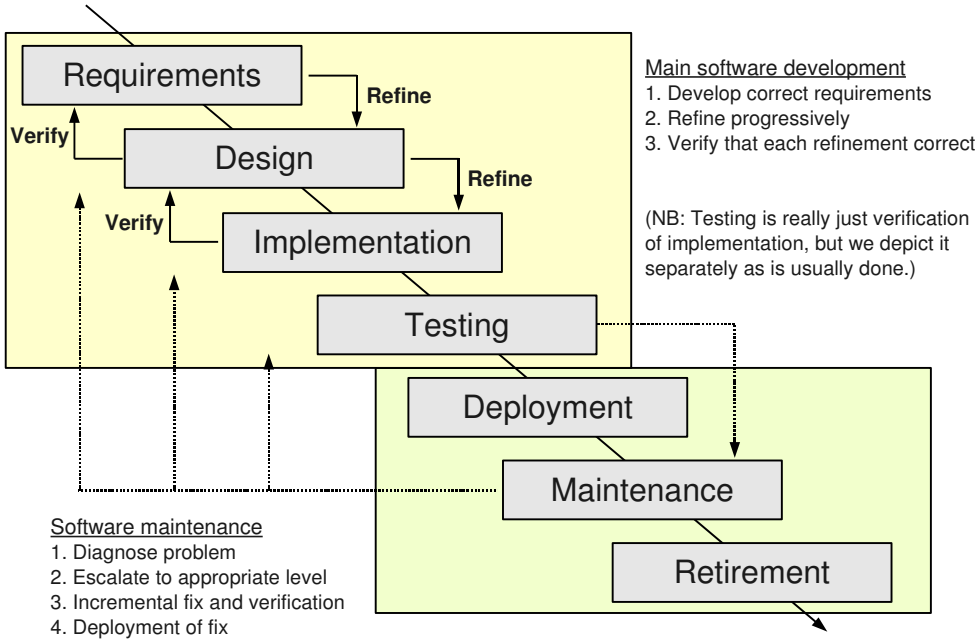
Crypto implementation environment



Software security

- Security depends on protocols and implementation
  - “Mathematical”, software, and hardware security
  - Other components of hardware and software potential problems as well
- Security is different from functionality in many ways
  - The attacker always has the advantage
  - Security is invisible, security failures are difficult to notice
- Penetrate-and-patch development model dominates today
  - Programs have (security) flaws, but we fix them ASAP
  - Measures to limit impact (defense-in-depth)
- Security is relative and often subjective
  - Relative to threat model
  - Computational security => always breakable

Life cycle model



Requirements and design issues

- Requirements issues
  - Threat model – assets, threats, attacker capabilities; tree model
  - Risk analysis => Security objectives & assumptions
- From requirements to design
  - Security objectives and assumptions => security mechanisms
- Common design principles
  - Software design “rules of thumb” to minimize design risks
- Assurance, trust, and certification
  - Semi-formal methods applied by a trusted party which assures a user that security has been implemented properly
  - E.g. Common Criteria, FIPS 140-1 and FIPS 140-2

Choosing a cryptosystem

- Choosing a cryptosystem
  - We first need security objectives and assumptions (e.g. Dolev-Yao)
  - Layer in networking stack affects choice
    - Packet-based vs. stream-based =>PGP, TLS, IPsec, WEP ?
    - => However, all layers underneath also a concern (e.g. DoS)
  - Organizational process and environment issues (e.g. PKI process)
- Should use an existing cryptosystem
  - Designing a cryptosystem from scratch is extremely difficult
  - It's better to use a subset of an existing cryptosystem
- Other concerns
  - Library support
  - Performance and code/data size

Implementation issues

- Random number generation
  - Critical to security => various requirements; Fortuna example
- Secure handling of storage media
  - How to store and erase secrets to/from RAM & magnetic media
  - Memory locking to avoid paging
  - Memory remanence – relevance; RAM encryption & Boojum
  - Magnetic media – encryption or destruction
  - => Difficult problem if threat relevant
- Cryptographic libraries
  - Layered services
  - Security problems caused by crypto libs – wooping (RSA example)

Implementation issues

- Side and covert channels – side effects of computation
  - Timing, power usage, E-M fields, cache interference, etc.
  - Can leak crypto secrets – D-H example, Bernstein AES key
  - Protection – constant time primitives, quality crypto library
  - => Difficult if threat relevant
  - => Especially local attacks with modern hardware architecture are problematic (hyperthreading, recent branch prediction attacks)

Preparing for the exam

Preparing for the exam

- Exam material
  - Exams are based on
    - (1) slides
    - (2) lectures
    - (3) assignment
  - Kaufman & Stallings are useful as supplementary material for course, but are **not** required reading for the exam
- What kind of questions ? Typically:
  - Multiple choice + justification
  - Essay answer, figures
  - Simple calculations

Preparing for the exam

- By default, all lecture topics are within exam scope
- However, we won't be asking extremely detailed questions, like
  - DES S-box contents or round function, or exact ARC4 algorithm
  - Detailed history of algorithms; Kerberos (or other) message contents
  - Exact **details** of Bernstein AES attack
- But you are required to remember basic definitions, such as
  - Diffie-Hellman and RSA computation (equations)
  - Overall structure of algorithms (e.g. DES is a Feistel network with 16 rounds, MD5 has Merkle-Damgård structure)
  - The services provided by Kerberos and its overall design
  - Why timing attacks work, how to eliminate them
  - Example – what side channel enables Bernstein's AES timing attack and why (answer: variable time table lookup with key related index => leaks key info.)
- You may need a calculator, though a basic one will do

Example questions

- Define stream cipher and block cipher with formulas, and explain how they differ in practice, from an application point of view
- Which modes of operation (of those covered in the course) have the property that a single bit change in ciphertext changes exactly one bit in the corresponding plaintext (when decrypting)? Justify.
- What is the effective key length (in bits) of 3DES? Why?
- Describe the Diffie-Hellman key exchange with formulas and a message sequence chart.
- Give two examples for sources of true randomness.
- Describe what a side channel is, and give an example of how a side channel might help in attacking a Diffie-Hellman key exchange.

Summary

- Course assignment review
- Summary and review of lectures
  - Lecture 1 Basics
  - Lecture 2-7 Crypto
  - Lecture 8-10 Implementation issues
- Preparing for the exam