New developments in cryptology

Prof. Bart Preneel
COSIC
Bart.Preneel(at)esat.DOTkuleuven.be
http://homes.esat.kuleuven.be/~preneel

Outline

• 1. Cryptology: protocols
  – identification/entity authentication
  – key establishment
• 2. Public Key Infrastructures
• 3. Secure Networking protocols
  – Internet Security: email, web, IPSEC, SSL
• 4. Using cryptography well
• 5. New developments in cryptology

How to use cryptographic algorithms

• Modes of operation
• The hash function disaster
• How to encrypt using RSA
• Algorithm: secure design and implementation
• Obfuscation
• SPAM fighting

How NOT to use a block cipher: ECB mode

P1
block cipher
C1

P2
block cipher
C2

P3
block cipher
C3

An example plaintext
Encrypted with substitution and transposition cipher

Encrypted with AES in ECB and CBC mode

How to use a block cipher: CBC mode

CBC mode decryption

What if IV is constant?

CBC with incomplete plaintext (1)

Repetition in P results in repetition in C: ⇒ information leakage  
need random and secret IV
CBC with incomplete plaintext (2)

Plaintext length in bytes

\[ P_1 \quad P_2 \quad P_3 || 1000..0 \]

\[ + 1100110011||0000….000 \]

\[ \text{IV} \quad \text{AES}^{-1} \quad \text{AES}^{-1} \quad \text{AES}^{-1} \]

\[ C_1 \quad C_2 \quad C_3 \]

\[ + 1100110011||0000….000 \]

CBC with incomplete plaintext (3)

Plaintext length in bytes

\[ P_1 \quad P_2 \quad P_3 || 1000..0 \]

\[ + 1100110011||0000….000 \]

- If the first 10 bits of P3 are equal to 1100110011 then after the modification P3’ will be equal to 0.
- The decryption will then produce an error message because the plaintext length field is incorrect.
- Conclusion: information on 1 byte of P3 can be obtained using on average 128 chosen ciphertexts.
- Protection: random padding or authenticated encryption.

Modes of Operation

- CTR mode allows for pipelining
  - Better area/speed trade-off
- authentication: E-MAC
  - CBC-MAC with extra encryption in last block
- authenticated encryption:
  - most applications need this primitive (ssh, TLS, IPsec, …)
  - for security against chosen ciphertext this is essential

Authenticated encryption

Inefficient solution: encrypt then MAC
We can do better

Issues:
- associated data
- parallelizable
- on-line
- patent-free
- provable security

Example: CCM: CTR + CBC-MAC

<table>
<thead>
<tr>
<th>SN</th>
<th>Length</th>
<th>CBC IV</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>...</th>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>SN 1</td>
<td>SN 2</td>
<td>SN n</td>
<td>SN n+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

<table>
<thead>
<tr>
<th>SN</th>
<th>Packet sequence number (WEP &quot;IV&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A5FD41234567890</td>
<td>SN = packet sequence number (WEP &quot;IV&quot;)</td>
</tr>
</tbody>
</table>

Hash functions

- MDC (manipulation detection code)
- Protect short hash value rather than long text
- collision resistance
- preimage resistance
- 2nd preimage resistance
**New Developments in Cryptography**

**Bart Preneel**

**February 2007**

---

**MDx-type hash function history**

- **MD4**
- **MD5**
- **HAVAL**
- **RIPEMD**
- **SHA-1**
- **SHA-256**
- **SHA-512**

---

**MD5**

- Advice (RIPE since '92, RSA since '96): stop using MD5
- Largely ignored by industry (click on a cert...)
- Collisions for MD5 are within range of a brute force attack anyway ($2^{64}$)
- [Wang+’04] collision in 15 minutes
- Today: collisions in seconds

---

**SHA-1**

- SHA designed by NIST (NSA) in ’93
- Redesign after 2 years (’95) to SHA-1
- Collisions found for SHA-0 in $2^{51}$ [Joux+’04]
- Reduced to $2^{39}$ [Wang+’05]
- Collisions for SHA-1 in $2^{80}$ [Wang+’05]
- Structured collisions for SHA-1 found for 64 out of 80 rounds [De Cannière-Rechberger’06]
- Prediction: collision for SHA-1 in 2007

---

**Impact of collisions (1)**

- Collisions for MD5, SHA-0, SHA-1
  - Two messages differ in a few bits in 1 to 3 512-bit input blocks
  - Limited control over message bits in these blocks
  - But arbitrary choice of bits before and after them
- What is achievable?
  - Two colliding executables
  - Two colliding postscript documents and gif files [Lucks, Daum ’05]
  - Two colliding RSA public keys – thus with colliding X.509 certificates [Lenstra, Wang, de Weger ’04]
  - Two arbitrary colliding files (no constraints) for 100K$"
Other properties?

- 2nd preimage attack close to feasible for MD4; not a problem for MD5/SHA-1
- HMAC
  - HMAC-MD4 is broken
  - HMAC-MD5 is questionable
  - HMAC-SHA1 seems ok
- Many other issues have been identified with all our hash functions

The future

- RIPEMD-160 seems more secure than SHA-1
- use more recent standards (slower)
  - SHA-256, SHA-512
  - Whirlpool
- Upgrading MD5 and SHA-1 in Internet protocols:
  - it doesn’t work: algorithm flexibility is much harder than expected
- NIST will probably run an open competition from 2007 to 2011

How to encrypt with RSA?

- Assume that the RSA problem is hard
- … so a fortiori we assume that factoring is hard
- How to encrypt with RSA?
  - Hint: ensure that the plaintext is mapped to a random element of [0,n-1] and then apply the RSA Encryption Permutation (RSAEP)

How (not) to encrypt with RSA?

- Non-hybrid schemes
  - RSA-PKCS-1v1_5 (RSA Laboratories, 1993)
  - RSA-OAEP (Bellare-Rogaway, 1994)
  - RSA-OAEP+ (Shoup, 2000)
  - RSA-SAEP (Johnson et al., 2001)
  - RSA-SAEP+ (Boneh, 2001)
- Hybrid schemes
  - RSA-KEM (Zheng-Seberry, 1992)
  - RSA-KEM-DEM (Shoup, 2001)
  - RSA-REACT (Okamoto-Pointcheval, 2001)
  - RSA-GEM (Coron et al., 2002)

RSA PKCS-1v1_5

- Introduced in 1993 in PKCS #1 v1.5
- De facto standard for RSA encryption and key transport
  - Appears in protocols such as TLS, S/MIME, ...

RSA-PKCS-1v1_5 Diagram

- Diagram showing the encryption process with RSA-PKCS-1v1_5
- Random nonzero bytes
- Padding
- Message
- Source: RSA Labs
- Public Key
- RSAEP
- C
RSA-PKCS-1v1_5 Cryptanalysis

- Low-exponent RSA when very long messages are encrypted [Coppersmith+ '96/Coron '00]
  - large parts of a plaintext is known or similar messages are encrypted with the same public key
- Chosen ciphertext attack [Bleichenbacher '98]
  - decryption oracle: ciphertext valid or not?
  - 1024-bit modulus: 1 million decryption queries
- These attacks are precluded by fixes in TLS

Bleichenbacher’s attack

- Goal: decrypt c
  - choose random s, 0 < s < n
  - computer c' = c se mod n
  - ask for decryption of c': m'
  - compute m as m'/s mod n
- but m’ does not have the right format!
- idea: try many random choices for s:
  - if no error message is received, we know that
    \[ 2^B < (m s \mod n) < 3^B \]
  - with \( B = 2^{8(k-2)} \) (k length in bytes of the modulus)

RSA-OAEP

- designers: Bellare and Rogaway 1993
- enhancements by Johnson and Matyas in 1996 ("encoding parameters")
- already widely adopted in standards
  - IEEE P1363 draft
  - ANSI X9.44 draft
  - PKCS #1 v2.0 (PKCS #1 v2.1 draft)
  - ISO 18033-2 working draft 2000

RSA-OAEP Diagram

- Source: RSA Labs

RSA OAEP - security

[BR '93] RSA-OAEP is IND-CCA2 secure under RSA assumption in ROM

Shoup '00: the proof is wrong

[POPS 01] RSA-OAEP is IND-CCA2 secure under partial domain one-wayness RSA assumption in ROM
  - for RSA: partial domain one-wayness ⇔ one-wayness

Reduction is very weak

ROM assumption is questionable

RSA OAEP - security

- Improved chosen ciphertext attack [Manger, Crypto '01]
  - requires a few thousand queries (1.1 log n)
  - opponent needs oracle that tells whether there is an error in the integer-to-byte conversion or in the OAEP decoding
- overall conclusion: RSA Inc. is no longer recommending the use of RSA-OAEP

if it’s provable secure, it probably isn’t
How to encrypt with RSA

- RSA-KEM
  - encrypt 2 session keys with RSA
  - encrypt and MAC data with these 2 keys
- Recommended in NESSIE report (http://www.cryptonessie.org) and to be included in ISO 18033
- Similar problems for signatures: ISO 9796-1 broken, PKCS#1 v1.0 questionable

Attack on PKCS #1 v1.5 implementations (1) [Bleichenbacher06]

- Consider RSA with public exponent 3
- For any hash value H, it is easy to compute a string “Magic” such that the above string is a perfect cube of 3072 bits
- Consequence:
  - One can sign any message (H) without knowing the private key
  - This signature works for any public key that is longer than 3072 bits
- Vulnerable: OpenSSL, Mozilla NSS, GnuTLS

Attack on PKCS #1 v1.5 implementations (2) [Bleichenbacher06]

- Fix
  - Write proper verification code (but the signer cannot know which code the verifier will use)
  - Use a public exponent that is at least 32 bits
  - Upgrade – finally – to RSA-PSS

Cryptographic algorithm selection

- Standards?
- Public domain versus proprietary
- Upgrades

Cryptographic standards

- Algorithms historically sensitive (e.g., GSM)
- Choices with little technical motivation (e.g., RC2 and MD2)
- Little or no coordination effort (even within IETF)
- Technically difficult

A.S. Tanenbaum: “The nice thing about standards is there’s so many to choose from”

Major Standardization Bodies in Cryptography

- International
  - ISO and ISO/IEC International Organization for Standardization
  - ITU: International Telecommunications Union
  - IETF: Internet Engineering Task Force
  - IEEE: Institute of Electrical and Electronic Engineers
- National
  - ANSI: American National Standards Institute
  - NIST: National Institute of Standards and Technology
- European
  - CEN: Comité Européen de Normalisation
  - ETSI: European Telecommunications Standards Institute
- Industry
  - PKCS, SECG
  - W3C, OASIS, Liberty Alliance, Wi-Fi Alliance, BioAPI, WS-Security, TCG
  - GP, PC/SC, Open Card Framework, Multos
**Independent evaluation efforts**

- **EU-funded IST-NESSIE Project (2000-2003):** new cryptographic primitives based on an open evaluation procedure (http://www.cryptonessie.org)
- **ECRYPT eSTREAM (2004-2007):** stream cipher competition

**Proprietary/secret algorithms**

- No “free” public evaluations
- Risk of snake oil
- Cost of (re)-evaluation very high
- No economy of scale in implementations
- Reverse engineering
- Fewer problems with rumors and “New York Times” attacks
- Extra reaction time if problems
- Fewer problems with implementation attacks
- Can use crypto for IPR and licensing

---

**Many insecure algorithms in use**

- Do it yourself (snake oil)
- Export controls
- Increased computational power for attacks (64-bit keys are no longer adequate)
- Cryptanalysis progress - including errors in proofs
- Upgrading is often too hard by design
  - cost issue
  - backward compatibility
  - version roll-back attacks

**Upgrade problem**

- GSM: A5/3 takes a long time
- Bluetooth: E0 hardwired
- TCG: chip with fixed algorithms
- MD5 and SHA-1 widely used
- Negotiable algorithms in SSH, TLS, IPsec,…
- But even then these protocols have problems getting rid of MD5/SHA-1

Make sure that you do not use the same key with a weak and a strong variant (e.g. GSM A5/2 and A5/3)

---

**And the good news**

- Many secure and free solutions available today: AES, RSA,…
- With some reasonable confidence in secure
- Cost of strong crypto decreasing except for “niche applications” (ambient intelligence)

**What to use (generic solutions)**

- Authenticated encryption mode (OCB, CWC, CCM, GCM) with 3-key 3-DES or AES
- Hash functions: SHA-512 or Whirlpool
- Public key encryption: RSA-KEM or ECIES
- Digital signatures: RSA-PSS or ECDSA
- Protocols: TLS, SSH, IKE(v2)

---

In spite of all the problems, cryptography is certainly not the weakest link in our security chain
Secure implementations of cryptography

- Error messages and APIs (cf. supra)
- Side channels
  - Timing attacks
  - Power attacks
  - Acoustic attacks
  - Electromagnetic attacks
- Fault attacks

Power analysis tools for smart cards

Software: constant time is crucial

- PIN verification
- Square and multiply for RSA
- Variable rotations in RC5 and RC6
- Swaps in RC4
- Problems with cache misses in ciphers with S-boxes such as DES and AES

PIN verification

input (PI N_U[0..k-1], PI N[0..k-1])
i = 0;
while (i < k) do {
  if (PI N_U[i] != PI N[i]) return (0);
  i = i + 1;
} return(1);

Problem?

Timing attack on RSA

- “square and multiply” algorithm
- Exponent bits scanned from MSB to LSB (left to right)

Let k = bitsize of d (say 1024)
Let s = m
For i = k-2 down to 0
  Let s = s*s mod n (SQUARE)
  If (bit i of d) is 1 then
    Let s = s*m mod n (MULTIPLY)
  End if
End for

Example:

init (MSB 1) s = m
round 2 (bit 0) s = m^2
round 1 (bit 0) s = (m^2)^2 = m^4
round 0 (bit 1) s = (m^4)^2 * m = m^9

Cache attack on crypto algorithms with S-boxes (DES, AES,…)

- Cache misses influence execution time
- Uses HyperThreading to monitor the encrypting process in real time and observe its use of shared resources.

- [Osvik-Shamir-Tromer 05] Cache Attacks and Countermeasures: the Case of AES, RSA CT 2006
- [Bernstein 05] Cache-timing attacks on AES
Some crypto libraries

- OpenSSL: http://www.openssl.org/
- Cryptlib: http://www.cs.auckland.ac.nz/~pgut001/cryptlib/
- SSLeay: http://www2.psy.uq.edu.au/~ftp/Crypto/
- IAIK Java: http://jce.iaik.tugraz.at/products/index.php
- COSIC crypto library (contact B. Preneel)
- See also http://www.ssh.fi/support/cryptography/online_resources/practical.html

Novel applications of cryptography

- Whitebox crypto
- SPAM fighting

Protection of software against whitebox attacks

- Software
  - Confidential information
  - Secret keys
  - Proprietary code
- Software and content distribution
- White-box setting
  - Complete access to implementation
  - Decompilation, reverse engineering, …

Protection of software against whitebox attacks

- “sandboxing” protect host against malware
- malicious hosts protect software against malicious hosts

Techniques

- White-box cryptography
  - Extra input and output coding of encryption
- Code obfuscation
  - Obfuscate code and program flow
- Other techniques:
  - Integrity checks + error detection
    - Tamper resistant software (TRS)
  - Code encryption + ‘on-the-fly’ decryption

White Box Cryptography

- Mathematical technique to hide keys in code
  \[ E'_K = G \circ E_K \circ F^{-1} \]
- With:
  - \( E_K \): encryption function, key \( K \)
  - \( F \): arbitrary input coding
  - \( G \): arbitrary output coding
Pro and Cons

- Unique object code
  - Choose $F$ and $G$
  - Integrate key
- Protect key
  - No function that computes $E_K$ for an arbitrary key $K$
- Flexible
- Fast updates

- Increased memory
  - Tables for input and output coding and for function
- Increased execution time
- Security: very strong attack model
  - Trade-off with performance
- Fast key update open problem

Example

- DES
  - 16-round Feistel
  - 8 S-boxes
  - 56-bit key
- White-box DES
  - General structure
  - 12 "T-boxes"
  - Key built in code

The SPAM problem: it is about economics, stupid

- list of $10^7$-$10^8$ "good" names
- cost per message: $\sim 10^{-5}$ €; total cost 100-1000 €
- hit ratio: $10^{-6}$ to $10^{-4}$: 10-10000 responses

- Cost to society
  - Ruining e-mail as communication tool
  - Time and attention
  - ISP fees
  - Storage and bandwidth

AND…

"The right to be left alone - the most comprehensive of rights, and the right most valued by civilized men."

- Supreme Court Justice Louis Brandeis

Fighting SPAM

- Filtering
  - Make sender pay
  - Ephemeral email addresses
  - Data/Sender Authentication

Fighting SPAM (2)

- Filtering
  - Everyone: text-based
  - Brightmail: decoys; rules updates
  - Microsoft Research: (seeded) trainable filters
  - SpamCloud: collaborative filtering
  - SpamCop, Osirusoft, etc: IP addresses, proxies, …
- Make Sender Pay
  - Computation (CPU and/or memory)
  - Human attention
  - Cash, bonds, stamps (PennyBlack)
Fighting SPAM (3)

- Ephemeral e-mail addresses
  - E.g. SPA: Single Purpose Addresses
- Data(Sender) authentication
  - Sign all emails
  - Sender Permitted From (SPF): whitelist mail senders
  - Sign domain names (Yahoo’s DomainKeys)
  - Authenticated mail: AMTP (TLS)

Often bypass for friends on whitelist

Filtering: limitations

- Still high cost if too late in the chain
- Spammers generate more sophisticated emails…
  - “Daphnia blue-crested fish cattle, darkorange fountain moss, beaverwood educating, eyeblinking advancing, dulltuned amazons....”
  - FWD: Many On Stocks. VAl/i/m + V1codin+ ; V/@GRa + /Xanax/ ; Prnt.4.m.in ? Som|a| muKPs

Computational Approach

- If I don’t know the sender:
  - Prove sender spent 10 seconds CPU time,
  - just for me, and just for this message
- Checking proof by receiver:
  - automatically in the background
  - very efficient
- All unsolicited mail treated equally

Point-to-Point Architecture

Sender client

\[ m, f(S,R,t,nonce) \]

Recipient client

(Ideal Message Flow)

- Single-pass “send-and-forget”
- Can augment with helper to handle slow machines
- Can add post office / pricing authority to handle money payments
- Time mostly used as nonce for avoiding replays (cache tags, discard duplicates; time controls size of cache)

Economics

- 10 seconds CPU cost a few hundreds of a cent
- \((80,000 \text{ s/day}) / (10s/message) = 8,000 \text{ msgs/day}\)
- Hotmail’s billion daily spams:
  - 125,000 CPUs
  - Up front capital cost just for hardware: $150 million
- The spammers can’t afford it.

Cryptographic Puzzles

- Hard to compute; \(f(S,R,t,nonce)\) can’t be amortized
  - lots of work for the sender
- Easy to check “\(z = f(S,R,t,nonce)\)”
  - little work for receiver
- Parameterized to scale with Moore's Law
  - easy to exponentially increase computational cost, while barely increasing checking cost
- Can be based on (carefully) weakened signature schemes, hash collisions
- Can arrange a “shortcut” for post office
Idea: replace CPU by memory

- CPU speeds vary widely across machines, but memory latencies vary much less (20-100 vs 2-6)
  - 33 MHz PDA vs. 3 GHz PC
- design a puzzle leading to a large number of cache misses
- Concrete schemes: [ABMW02] and [DGN03]

Easy Functions [ABMW02]

- \( f: n \text{ bits to } n \text{ bits, easy} \)
- Given \( x_k \in \text{range}(f) \), find a pre-image with certain properties
- Hope: best solved by building table for \( f^{-1} \) and working back from \( x_k \)
- Choose \( n = 22 \) so \( f^{-1} \) fits in small memory, but not in cache
- Optimism: \( x_k \) is root of tree of expected size \( k^2 \)

Social Issues

- Who chooses \( f \)?
  - One global \( f \)? Who sets the price?
  - Autonomously chosen \( f \)’s?
- How is \( f \) distributed (ultimately)?
  - Global \( f \) built into all mail clients? (1-pass)
  - Directory? Query-Response? (3-pass)

Technical Issues

- Distribution lists
- Awkward introductory period
  - Old versions of mail programs; bounces
- Very slow/small-memory machines
  - Can implement “post office” (CPU), but:
    - Who gets to be the Post Office? Trust?
- Cache Thrashing (memory-bound)
- The Subverters or Zombies

Conclusions: cryptography

- Can only move and simplify your problems
- Solid results, but still relying on a large number of unproven assumptions and beliefs
- Not the bottleneck or problem in most security systems

To paraphrase Laotse, you cannot create trust with cryptography, no matter how much cryptography you use — Jon Callas.

Conclusions (2): cryptography

- Leave it to the experts
- Do not do this at home
- Make sure you can upgrade
- Implementing it correctly is hard
- Secure computation very challenging and promising: reduce trust in individual building blocks
New Developments in Cryptography

Bart Preneel

SPAM

- C. Dwork, M. Naor: Pricing via Processing or Combating Junk Mail; Crypto '92, LNCS 740, Springer-Verlag, Berlin 1992, 139-147.
- C. Dwork, A. Goldberg, M. Naor, On Memory-Bound Functions for Fighting Spam, Crypto 2003, 426-444.

Some books on cryptology


More books on Cryptology

- B. Schneier, N. Ferguson, Practical Cryptography, Wiley, 2003. A good short overview with strong focus on implementation aspects