Key Management

Prof.Dr. Ferucio Laurenţiu Ţiplea

Department of Computer Science
Alexandru Ioan Cuza University of Iaşi
Iaşi, Romania
E-mail: fltiplea@info.uaic.ro

Outline

Outline

Cryptographic Keys

- Classification by algorithm type
  - symmetric keys : for symmetric cryptography
  - (public key, private key) : for asymmetric cryptography
- Classification by intended use
  - for confidentiality (encryption)
  - for data origin authentication (signature or MAC)
  - for entity authentication (signature or MAC)
  - for key agreement
- Classification by duration of use
  - long-term keys
  - short-term keys

“A chain is only as strong as its weakest link”

The security of the system is dependent on the security of the keys !

A security system without strong key management procedures has no security !

Key Management

Key management goals:

- Generation of keying material
- Distribution of keying material
- Storage (including backup, recovery, archival) of keying material
- Usage of keying material
- Update, revocation and destruction of keying material

Standards

- International standards: ANSI X9.17 / ISO 8732, ANSI X9.24, ISO 11166, ISO 11568
  - defines both the manual and automated management of keying material used for financial services such as point-of-sale (POS) transactions (debit and credit), automated teller machine (ATM) transactions, messages among terminals and financial institutions, and interchange messages among acquirers, switches and card issuers
- National standards: ETEBACS (France), AS2805.6.xx (Australia), APACS 40 & APACS 70 (UK)

Adherence to standards does not guarantee security !

Key Management Systems

Examples of key management systems:

- Cryptomathic (http://www.cryptomathic.com/)
- KeyConductor (http://www.capturetech.com/)
- keyAuthority (http://www.thales-esecurity.com/) etc.

Choice of a key management system usually determined by:

- network topology
- cryptographic services (confidentiality, authentication etc.)
- cryptographic mechanism (digital signature, MAC etc.)

S. Bellovin (RFC 4107, June 2005): Key management schemes should not be designed by amateurs !

Outline

Outline

Key Generation

Key generation techniques:

- Random bit generator (RBG)
- Derivation from source (initial) keying material (such as from another key or from a password)

NIST SP 800-133: Recommendation for Cryptographic Key Generation

Key Length

Key lengths for confidentiality:

duration	symmetric	RSA	ECC
days/hours	50	512	146
10-20 years	103	2048	206
30-50 years	141	4096	282

Figure: From http://www.ecrypt.eu.org/documents/D.SPA.20.pdf

Assumptions: no quantum computers; no breakthroughs; limited budget

Outline

Definitions for Random Sequences

- An infinite sequence is random if the quantity of information it contains (in the sense of Shannon's information theory) is also infinite
  - not very useful : it is not possible in practice to produce and process infinite sequences
  - formally impossible to verify whether a finite sequence is random or not (it is only possible to check that it shares the statistical properties of a random sequence)
- [Knuth, D.: The Art of Computer Programming, 1981] : a sequence of random numbers is a sequence of independent numbers with a specified distribution and a specified probability of falling in any given range of values
- [Schneier, B.: Applied Cryptography: Protocols, Algorithms, and Source Code in C, 1996] : a sequence of random numbers is a sequence that has the same statistical properties as random bits, is unpredictable and cannot be reliably reproduced

General Requirements

Two general requirements a random sequence should fulfill:

- (R1) Random sequences should have “good” statistical properties
  - (R1) is usually checked by applying a particular statistical test suite, such as the NIST 800-22 Statistical Test Suit or Diehard Tests of Randomness;
  - (R1) may be sufficient for some applications (e.g., for challenge-and-response protocols) but may be insufficient for others (e.g., generation of session keys).
- (R2) The knowledge of subsequences of random sequences shall not allow one to practically compute predecessors or successors or to guess these numbers with non-negligible larger probability than without knowledge of these subsequences

Classification

Random number generators (RNG) can be classified as follows:

- deterministic random number generators (PRNG) (also called pseudo-random number generators (PRNG)).
  PRNGs generate random number sequences algorithmically, starting with a seed. They may be pure or hybrid;

- non-deterministic random number generators (also called true random number generators (TRNG)). They may be:
  - physical. These generators use non-deterministic effects of electronic circuits (e.g., inherent semiconductor thermal noise) or physical experiments (e.g., radioactivity, quantum processes);
  - non-physical. These generators use non-deterministic events (e.g., system time, hard disk seek time, user interaction).

  Each of the physical or non-physical random number generators may be pure or hybrid.

Pure PRNG

Definition 1

A pure PRNG is a 4-tuple $G = (S, O, \delta, g)$, where:

- S is a finite set of states;
- O is a finite set of outputs;
- $\delta : S \to S$ is the transition function;
- $g : S \to O$ is the output function.

Given an initial state s_0 (called seed), G generates a sequence of outputs

$$r_1, r_2, \cdots$$

where $r_i = g(s_i)$ and $s_i = \delta(s_{i-1})$, for all $i \geq 1$.

Pure PRNG

Remark 1

1. *The seed is generated outside G.*
2. *In order to meet (R2), the transition and output functions should be sufficiently complex, and the entropy of the seed should be large;*
3. *The least $p > 0$ with $s_{n+p} = s_n$ for all $n \geq 1$ is called the period of G. Large periods are better.*

Examples of Pure PRNGs

Example 2 (Linear congruential generator)

A linear congruential generator is characterized by:

- $S = \mathbf{Z}_m$ $(m > 0)$;
- $\delta(x) = ax + c \ mod \ m$, where $a \in \mathbf{Z}_m - \{0\}$ (the multiplier) and $c \in \mathbf{Z}_m$ (the increment);
- the output function may have the form $g(x) = \frac{x}{m}$ or $g(x) = \frac{x}{m-1}$ or $g(x) = \frac{x+1/2}{m}$.

When $c = 0$, it is called Lehmer generator.

It can be shown that the period is maximum if and only if

- $(c, m) = 1$;
- $a - 1$ is divisible by all prime factors of m;
- $a - 1$ is a multiple of 4 if m is a multiple of 4.

Examples of Pure PRNGs

Example 3 (Multiple recursive generator)

A multiple recursive generator is characterized by:

- $S = \mathbf{Z}_m^k$ ($m > 0$ and $k > 0$);
- $\delta(x_1, \ldots, x_k) = (x_2, \ldots, x_k, x_{k+1})$, where $x_{k+1} = \sum_{i=1}^{k} a_i x_i \ mod \ m$ $(a_1, \ldots, a_k \in \mathbf{Z}_m)$;
- the output function may have the form $g(x_1, \ldots, x_k) = \frac{x_1}{m}$.

Examples of Pure PRNGs

Example 4 (PRNG from block cryptosystems)

Assume that $\mathscr{S}$ is a block cryptosystem which works on n-bit blocks with m-bit keys and the result is an n-bit block. Then, we can define a PRNG as follows:

- $S = \{0,1\}^n \times \{0,1\}^m$;
- $\delta(x,K) = (\{x\}_K, K)$;
- $g(x,K) = x$.

If no statistical weaknesses are known for $\mathscr{S}$, then (R1) should be fulfilled. If we assume now that $r_i, \ldots, r_{i+j}$ are known, then finding r_{i-1} or r_{i+j+1} is as hard as a chosen plaintext attack on $\mathscr{S}$. Assuming that K is kept secret and cannot be guessed with non-negligible probability, we conclude that this PRNG should fulfill (R2).

Requirement (R3)

If an attacker get knowledge of the current internal state of the PRNG from the previous example, then (R2) does not hold anymore. As (R2) is important in many applications, it is desirable to add one more requirement to PRNG:

> (R3) The knowledge of the internal state shall not allow one to practically compute "old" random numbers or even a previous internal state or to guess these values with non-negligible larger probability than without knowledge of the internal state.

(R3) demands a one-way transition function!

Examples of pure PRNGs

Example 5 (PRNG from hash functions)

Assume that h_1 and h_2 are two distinct hash functions on n-bit blocks (yielding n-bit blocks). Then, we can define a PRNG as follows:

- $S = \{0,1\}^n$;
- $\delta = h_1$;
- $g = h_2$.

If h_1 and h_2 are one-way functions, then the above PRNG fulfills (R3).

Remark 2

If $h_1 = h_2$ in the above example, then the PRNG does not fulfill (R2) because one can easily obtain the successor of a random value from the current random value.

Examples of pure PRNGs

Example 6 (Blum-Blum-Shub PRNG (BBS))

Let p and q be two large distinct m-bit primes with $p \equiv 3 \equiv q \ mod \ 4$ and $n = pg$. Then, we can define a PRNG as follows:

- $S = \mathbf{Z}_n$;
- $\delta(x) = x^2 \ mod \ n$;
- $g(x) = x \ mod \ 2$ (the last significant bit of x).

The generator is seeded with a quadratic residue x_0 modulo n (that is, $x_0 = r^2 \ mod \ n$, for some $r \in \mathbf{Z}_n^*$).

Remark 3

- *The BBS generator is slow due to the modular multiplication it uses.*
- *It can be shown that BBS is cryptographically secure in the sense that the next outputted bit is unpredictable. Moreover, this property holds true even if BBS outputs not only the least significant bit but also the least* $\log\log m$ *significant bits.*

Hybrid PRNGs

In a hybrid PRNG, the update process of the current internal state takes into consideration not only the current state but also an additional input from some finite set $E_\infty = E \cup \{\infty\}$. When the element ∞ is used, this means that the update of the current state is only performed on the basis of the current state. Therefore, if $E_\infty = \{\infty\}$ then the hybrid PRNG is in fact a pure PRNG.

Definition 7

A hybrid PRNG is a 5-tuple $G = (S, O, E, \delta, g)$, where S, O, and g are as in the case of a pure PRNG, E is a finite set (not including ∞), and δ is a function from $S \times E_\infty$ into S with the property $\delta(s, \infty) = s$, for any $s \in S$.

The generation of a sequence of elements by G follows the same line as in the case of pure PRNG but with the difference that a state s is updated by the following procedure:

- generate, according to some probability distribution, an element $e \in E$;
- compute the new current state $s' = \delta(s, e)$.

Requirement (R4)

In the case of pure PRNG, the knowledge of an internal state may compromise the generator. This might not be the case of hybrid PRNG because the generation of a new state takes into account external inputs. But, it may be desirable to add one more requirements:

> (R4) The knowledge of the internal state shall not allow one to practically compute the next random numbers or to guess these values with non-negligible larger probability than without knowledge of the internal state.

Whether (R4) is met depends on the randomness of the external input.

Examples of hybrid PRNGs

Example 8 (ANSI X9.17 PRNG)

We consider the cryptosystem two-key EDE 3DES and define the following PRNG:

- $S = \{0,1\}^{64} \times \{0,1\}^{128}$;
- $E = \{0,1\}^{64}$;
- $\delta((x,K),t) = \{\{x \oplus \{t\}_K\}_K \oplus \{t\}_K\}_K$;
- $g(x,K) = \{x \oplus \{t\}_K\}_K$.

The external input t is the 64-bit representation of the current (date and) time (the (date and) time just before the generation of a new state).

This PRNG does not fulfill (R3) if the adversary knows the exact time when the random numbers are generated (that is, if the adversary knows t).

Outline

Key Derivation Functions

- A key derivation function (KDF) takes a source of initial keying material and derives from it one or more cryptographically strong secret keys
- The main difficulty in designing a KDF relates to the form of the initial keying material
- Two types of initial keying material:
  - "large" initial keying material
  - passwords (Password-based KDF (PBLDF))

More on this in class !

Outline

Outline

Key Establishment

Key establishment = protocol whereby a shared secret becomes available to two or more parties (for subsequent cryptographic use)

Key establishment may use:

- pre-shared secrets or keys
- centralized or trusted parties (trusted third party (TTP), trusted server, authentication server, key distribution center (KDC), key translation center (KTC), certification authority (CA))

  They may be:

  - in-line
  - on-line
  - off-line

- symmetric or asymmetric cryptography

Key Establishment

Requirements:

- (mutual) entity authentication
- (mutual) data origin authentication
- implicit key authentication
- (mutual) key confirmation
- explicit key authentication
- key confidentiality, key freshness, joint key control

Key Establishment

General classification:

- key distribution
  - point-to-point techniques - communicating parties involved directly
  - centralized techniques - trusted (third) party involved
- key agreement - mainly used to establish keying material

Some techniques may benefit of keying material issued in advance on initialization of the system - these are called pre-distribution techniques

Outline

Needham-Schroeder Shared-key Protocol

- Notations: T is a KDC, encryption is symmetric, and K_{XT} is the key shared by X and T
- Protocol:

$$
\begin{array}{lllll}
1. & A \to T: & (N_A) & A, B, N_A \\
2. & T \to A: & & \{N_A, B, K, \{K, A\}_{K_{BT}}\}_{K_{AT}} \\
3. & A \to B: & & \{K, A\}_{K_{BT}} \\
4. & B \to A: & (N_B) & \{N_B\}_K \\
5. & A \to B: & & \{N_B - 1\}_K
\end{array}
$$

- Goal: A requests from T a communication key with B, and then transports the key to B
- Properties: key authentication (due to the trusted server T), key confirmation, and entity authentication of A to B
- Remark: no longer recommended (important primarily for historical reasons); it is the basis for many server-based authentication and key distribution protocols such as Kerberos and Otway-Rees

Needham-Schroeder-Lowe Public-key Protocol

- Notations: K_X is the public key of X
- Protocol:

$$\begin{array}{lllll} 1. & A \to B: & (N_A) & \{N_A, A\}_{K_B} \\ 2. & B \to A: & (N_B) & \{N_A, N_B, B\}_{K_A} \\ 3. & A \to B: & & \{N_B\}_{K_B} \end{array}$$

- Goal: A and B agree on (N_A, N_B) as the source keying material
- Properties: entity authentication of B to A, assurance that B knows N_A, assurance that A knows N_B

Diffie-Hellman Protocol

- Notations: p is a (public) large prime and α is a (public) primitive root modulo p. All operations below are $\bmod p$
- Protocol:

$$\begin{array}{lll} 1. & A \to B: & (x \leftarrow \{2,\ldots,p-2\}) \quad \alpha^x \\ 2. & B \to A: & (y \leftarrow \{2,\ldots,p-2\}) \quad \alpha^y \end{array}$$

- Goal: A and B agree on α^{xy} as the source keying material
- Properties: vulnerable to man-in-the-middle attack

Station-to-Station Protocol

- Notations: p is a (public) large prime, α is a (public) primitive root modulo p, and $sig_X(m)$ is X's RSA signature on a hash value of m. All operations below are $\bmod p$
- Protocol:

$$\begin{array}{lll} 1. \quad A \to B: & (x \leftarrow \{2,\ldots,p-2\}) & \alpha^x \\ 2. \quad B \to A: & (y \leftarrow \{2,\ldots,p-2\}) & \alpha^y,\ \{sig_B(\alpha^y,\alpha^x)\}_K \\ 3. \quad A \to B: & & \{sig_A(\alpha^x,\alpha^y)\}_K \end{array}$$

  where $K = \alpha^{xy}$
- Goal: A and B agree on α^{xy} as the source keying material
- Properties: mutual entity authentication and mutual explicit key authentication
- Remark: there are several variations of this protocol such as full STS (certificates included) and STS-MAC ($\{\cdot\}_K$ replaced by MAC_K)

Outline

Certifying Public-keys

- Public keys for encryption or signature verification must be certified !
- A Public-key Certificate binds an identity to a particular public key value
- Core elements of a public-key certificate:
  - name of owner
  - public-key value
  - validity time period
  - signature (of the creator of certificate)
- Examples: X.509 v3

Managing Certificates

- Certification Authority (CA)
- Certifying the certifiers:
  - cross certification
  - certificate hierarchies
  - certificate chains
- Revocation
  - certificate revocation list (CRL)
  - on-line certification status protocol (OCSP)

Public Key Intrastructures (PKIs)

1. PKIs are currently the primary means for practical deployment of public key encryption (PKE) schemes
2. PKIs support the management of public-keys and certificates
3. Some PKI problems:
    1. Difficulty in retrieving keys and certificates
    2. Questionable value of certified key representations
    3. Certificate processing complexity
    4. Costly certificates
    5. Problematic cross-domain trust management
    6. Naming semantics
    7. Use with insecure clients
    8. Privacy compromises

PKI: Usability Studies

1. Why Johny Can't Encrypt: A Usability Case Study of PGP 5.0, by A. Whitten and J.D. Tyger, 1999
2. Johny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express, by S.I. Garfinkel and R.C. Miller, 2005
3. Why Johny Still Can't Encrypt: Evaluating the Usability of Email Encryption Software, by S. Sheng, L. Broderick, C.A. Koranda, and J.L. Hyland, 2006
4. Why (Special Agent) Johnny (Still) Can't Encrypt: A Security Analysis of the APCO Project 25 Two-Way Radio System, by S. Clark, T. Goodspeed, P. Metzger, Z. Wasserman, K. Xu, and M. Blaze, 2011

Outline

Other Techniques

- Techniques based on secret sharing
- Quantum key distribution
- Identity-based encryption !
  - Voltage security: `http://www.voltage.com/`

Outline

Key Storage

Techniques for storing secret keys:

- Inside a tamper-resistant hardware security module (HSM)
- On a smart card or other token encrypted with another key
- Stored on a database

Local Master Key (LKM):

- Used to encrypt other keys
- It is usually a strong key
- Stored inside a HSM

Outline

Key Update, Revocation, and Destruction

- In all cryptographic systems there should be the facilities to change keys on a regular basis (updates) or for the compromised keys (revocation)
- Keys, when no longer needed, must be destroyed in a secure manner

  ANSI X9.17: "Paper-based keying materials shall be destroyed by crosscut, shredding, burning or pulping. Keying material stored on other media shall be destroyed so that it is impossible to recover by physical or electronic means."

Outline

Key Use

Techniques for controlling the use of keys:

- Key tags: a simplified method for specifying allowed uses of keys
- Key variants: keys derived from a base key (for instance, $K \oplus v$ or $\{r\}_K$, where v and are random)
- Key notarization: prevents key substitution by requiring explicit identification of the parties
- Control vectors: usually combine key tags and key notarization

On-going research: use access control models