CPSC/PMAT 669

Key Management

Mike Jacobson

Department of Computer Science
University of Calgary

Topic 10

Outline

Random Numbers in Cryptography

There are many uses of **random numbers** in cryptography:

-
-
-
-

It is critical that these values be

-
-

How to Obtain Randomness?

The only source of true randomness is the real world.

- Find a regular but random event and monitor. Examples:
  -
  -
  -
  -
  -
- Need special hardware in general (e.g. radiation counters)
- Can be slow and cumbersome
- Problems of bias or uneven distribution — have to compensate or use noisiest bits from each sample). One possibility: pass data through a cryptographically secure hash function.

Pseudo-Randomness

Published collections of random numbers also exist, but they are too limited and well-known for most uses.

In practice, one uses **pseudo-randomness**.

Definition 1 (Pseudorandom Number/Bit Generator (PRNG, PRBG))

An algorithmic technique to create sequences of statistically random numbers/bits, initialized with a random **seed**.

Statistical Randomness Tests

NIST FIPS 140-1 is a standard for randomness for PRBGs. It includes (among others) the following statistical tests, which take as input 20000 PRBG produced bits:

1. **Monobit test** — the number of '1's should be strictly between 9654 and 10346
2. **Poker test** — divide the sequence into 5000 non-overlapping blocks of length 4 and determine whether the frequencies of each length 4 block are as expected
3. **Runs test** — B_i (number of runs of '1's of length i) and G_i (number of gaps of length i) must fall into the expected intervals for $1 \leq i \leq 6$ (runs of length greater than 6 are counted as being length 6)
4. **Long run test** — no runs of length greater than 34

Cryptographically Secure PRBGs

Simple PRNG example: *linear congruential generator*

$$X_{n+1} = aX_n + c \pmod{m} .$$

- Advantage: outputs long statistically random sequences
- Disadvantage: fails unpredictability — it is too easy to reconstruct entire sequence given only a few values

Definition 2 (Cryptographically secure PRBG (CSPRBG))

Must pass the **next-bit test**: there is no polynomial time algorithm that, on input of the first k bits of an output sequence, can predict the $(k+1)$-st bit with probability significantly greater than $1/2$.

For all practical purposes, a CSPRBG is unpredictable.

Examples of CSPRNG

Non-Example: linear congruential PRNG

Simple Examples (idea: output of a strong hash function or block cipher is statistically random)

- $X_i = H(X_{i-1})$ where X_0 is a random seed (predictable, but good for distilling random bits from another source).
- $X_i = E_{K_m}(C+1)$ where K_m is a protected master key and C is a counter of period N. Seems to be computationally infeasible to predict next X_i if K_m is secret.

More Complicated Examples:

- ANSI standard X9.17 — three separate applications of 3DES $(E_{K_1} D_{K_2} E_{K_1})$ to generate a 64-bit random number
- Blum-Blum-Schub PRBG — satisfies the next-bit test under the assumption that the QRP is intractable

ANSI X9.17

Inputs:

DT_i — 64-bit representation of the current date and time
V_i — 64-bit seed value

Keys: two 56-bit DES keys K_1, K_2

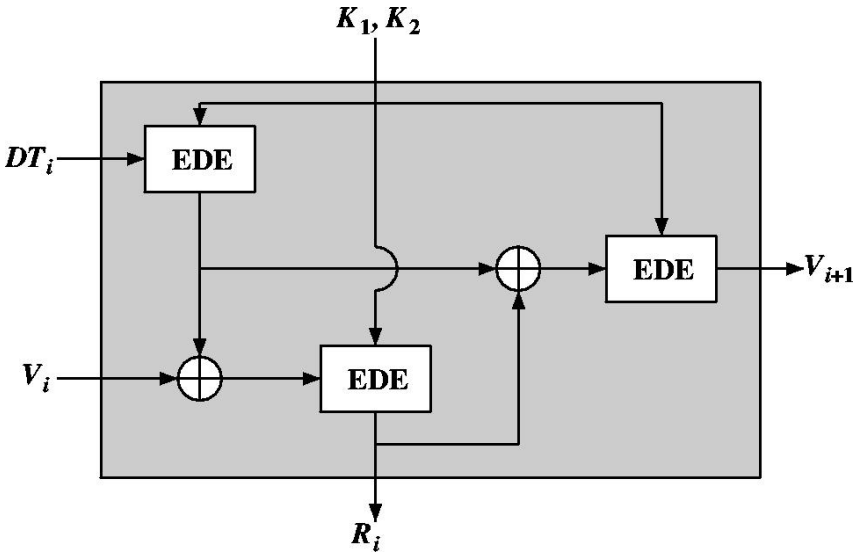
Outputs:

R_i — 64-bit pseudorandom bit string
V_{i+1} — seed value for next iteration

Formulas:

$$R_i = 3DES_{K_1,K_2}\left(V_i \oplus 3DES_{K_1,K_2}(DT_i)\right)$$
$$V_{i+1} = 3DES_{K_1,K_2}\left(R_i \oplus 3DES_{K_1,K_2}(DT_i)\right)$$

Diagram of ANSI X9.17



Properties of ANSI X9.17

ANSI X9.17 is one of the strongest PRNGs:

- 112-bit key
- Each round depends on the current date/time DT_i and the seed value V_i which is distinct from the previous pseudorandom number R_i produced
- Even if R_i is compromised, can't derive V_{i+1} without "'inverting" $3DES$

Appears unpredictable in practice (no proof).

Can substitute 3DES with any secure block cipher (eg. AES)

Blum-Blum-Shub CSPRBG

Satisfies the next-bit test under the assumption that the QRP is intractible.

Algorithm (system paramters $n = pq$, $p \equiv q \equiv 3 \pmod 4$, p, q prime):

1.
2.
    1.
    2.
    3.

Note: can also securely use the least significant $\lfloor \lg \lfloor \lg n \rfloor \rfloor$ bits of the x_i

Security

Can prove that any algorithm that computes the *previous* bit in the sequence can be used to solve the QRP.

- Thus, if the QRP is hard, BBS passes the previous bit test.
- Can show that this is equivalent to the next bit test.

Also the basis of the semantically-secure Blum-Goldwasser PKC

- XOR the pseudorandom sequence with the plaintext to encrypt, also send (with the ciphertext) the next x_i value.
- Given p and q (the private key), can recover the seed X_0 by computing square roots modulo p and q and applying the CRT.

See handout for more details.

Common Mistakes with PRNGs

The seed must have sufficient **entropy** to make it unpredictable.

The following are all real life (bad) examples!

1. Generating a random 512-bit prime using a 32-bit seed for the random number generator.
2. Generating a random 512-bit prime by calling a system PRNG that produces 32-bit random numbers, padding with 0s to 512 bits, and looking for the smallest prime greater than the number.
3. Instead of padding with zeros, call the system PRNG 16 times to generate 16 32-bit random numbers.

Fourth Bad Example — Kerberos 4

Kerberos 4 generates DES session keys by using a PRNG, seeded with a 32-bit value, to produce two 32-bit random numbers.

Problem: only 32 bits of entropy (should be 56)

Bigger problem: seed is the XOR of 5 random 32-bit numbers:

- time of day in seconds since Jan. 1, 1970
- fractional part of the current time in microseconds
- process ID of Kerberos server process
- cumulative count of session keys produced so far
- host id of the machine on which Kerberos is running

Possible Fix to Kerberos 4

Compute a hash on the concatenation of the 5 values.

- Every bit of randomness contributes to every bit of the session key
- Successive applications of the hash function will produce further pseudorandom bits (but with no more total entropy than the seed)

See the Internet Engineering Task Force's (IETF) Request For Comments RFC 1750 "Randomness Recommendations for Security" for more information about guidelines for deploying random number generators. Section 6 covers recommendations for software-based strategies for example.

Fifth Bad Example — Factoring RSA Moduli

1. Scrounge the internet for lots of RSA public keys with moduli $n_1, n_2, \ldots$
2. Compute $\gcd(n_i, n_j)$ for lots of $i \neq j$

You'd be surprised how many of the moduli you can factor!

Problem:

NIST Recommendations

Moral: The number of bits of entropy must correspond to the overall bit security of the system.

Example: 1024-bit RSA provides 80 bits of security, so the seed material for the PRNG must have at least 80 bits of entropy.

NIST's Recommendations for Security levels:

RSA modulus (in bits)	1024	2048	3072	8192	15360
Hash function size (in bits)	160	224	256	384	512
Security level (in bits)	80	112	128	192	256

Authenticity of Keys

Secure communication requires proper mechanisms for managing keys and ensuring their **authenticity**.

Mechanisms for ensuring authenticity of keys:

-
  -
  -
-
-

Symmetric Key Distribution

Symmetric schemes require both parties to share a common, secret key.

Possible distribution mechanisms:

- A selects a key and physically delivers to B. Secure, but cumbersome.
- Third party selects and physically delivers key to A and B. Also secure, but cumbersome.
- A and B can use a previous key to encrypt a new key. If one key is compromised, all subsequent keys are compromised.
- A commonly-trusted third party called a *key distribution center* (KDC) can relay the key between A and B via encrypted links (commonly used solution).

Key Distribution Centres

Idea:

- Each user holds a shared symmetric **master key** with the KDC
- Master key is used for distributing one-time **session keys**
- Encryption is performed with a session key that is destroyed at the end of the session

Advantages:

-
-

Key Distribution Centres: Issues

Issues:

-
-
-

PKC Solutions

There are three main contributions in PKC:

- **Digital signatures** — for data origin authentication and non-repudiation
- **Key agreement protocols** — both parties contribute to the generation of a session key (eg. Diffie-Hellman)
- **Key transport** via hybrid encryption — party A generates a session key, encrypts and sends to B using a PKC (B has no control over the session key)

Main problem — user's public keys must be *authenticated* in order to prevent active attacks such as man-in-the-middle and impersonation.

Public-Key Distribution, I

1. **Point-to-point delivery over a trusted channel** such as personal exchange, registered mail, courier, etc.
   Problems:
2. **Direct access to a trusted public file** (public-key repository).
   Advantage: no user interaction.
   Problems:
   -
   -

Public Key Distribution, II

3. An **on-line trusted server** dispenses public keys on request. The server signs the transmitted keys with its private key.
   Problems:
   -
   -
   -
   -
4. **Off-line server and certificates** (certification authorities).
5. Use of **systems implicitly guaranteeing authenticity** of public parameters (ID-based systems).

Option 5 is feasible, but has its own problems. We will focus on Option 4.

Public-Key Infrastructures

Definition 3 (Public-Key Infrastructure (PKI))

A set of techniques and procedures supporting authenticated key management for PKC. Specifically, a PKI supports:

-
-
-
-
-

Public-Key Certificates

Definition 4 (Public-Key Certificate)

A data structure consisting of a **data part** (containing at least the user ID and the corresponding public key) and a **signature part** consisting of the digital signature of a *certification authority* over the data part.

A certificate should also include information such as:

-
-
-
-

Certification Authorities

Definition 5 (Certification Authority (CA))

A trusted third party whose signature on a certificate vouches for the authenticity of the public key bound to the subject entity.

Idea: CA issues public key certificates that may be verified off-line. Users may exchange authentic public keys without having to contact the CA.

Example 1

X.509 is an IETF (Internet Engineering Task Force) standard for certificate-based authentication schemes (used in S/MIME, IPsec, SSL). **VeriSign** is an online CA service that uses X.509 certificates.

Obtaining Public Keys

User B uses a public-key certificate to obtain the authentic public key of user A as follows:

1. Acquires the authentic public key of the CA (done only once, eg. pre-loaded in web browsers)
2. Acquires a public key certificate corresponding to A over an insecure channel such as a central database, or even directly from A
3. Verify the authenticity of the public key:
   (a) Verifies the currency of the certificate using the time-stamp
   (b) Verifies the signature on A's certificate using CA's public key
   (c) Verifies that the certificate has not been revoked
4. If all the checks succeed, accepts the public key in the certificate as A's public key

Requirements for the Scheme

1. Any participant can read a certificate to determine the name and public key
2. Any participant can verify that the certificate originated from the CA and is not counterfeit
3. Only the CA can create and update certificates
4. Any participant can verify the currency of the certificate

Main Issue / Problem:

-
-

User Registration

Users must register with the CA in a secure manner (typically in person):

- The CA's public key (required for certificate verification) may be obtained at that time
- CA may generate user keys, or certify owner-generated keys (possibly without user revealing the private key)
- May store keys for backup

CA must verify the **binding** between the public and private keys.

CA Hierarchies

Large networks have hierarchies of CAs:

- Tree hierarchy — each node represents a principal whose public key is certified by its parent
- Leaf nodes — end users
- Non-leaf nodes — CAs at various levels and domains (e.g. country level has domains
  industry (.com)
  education (.edu)
  government (.gov)
  other organization (.org, .net)
- Two end users can obtain authentic public keys by finding a common ancestor node in the hierarchy

Certificate Revocation

Certificates may need to be revoked before they expire, for the following reasons:

- A user's private key is compromised
- A user is no longer certified by his current CA
- A CA's certificate is assumed to be compromised

Mechanisms for revocation:

- CA maintains a **certificate revocation list** (CRL), available online, signed by the CA
- Alternatively, incremental lists known as **Delta revocation lists** are disseminated through the hierarchy
- CA must time-stamp revocations — signatures issued prior to revocation date should be considered valid

Identity Based Cryptography

Motivation: an ideal e-mail system in which knowledge of a person's name (or e-mail address) alone is sufficient to

- send mail which can be read by that person only (secure),
- allow verification of signatures that could have only been produced by that person.

Idea (Shamir 1984): bind public keys directly to a user's identity

Definition 6 (Identity-based cryptosystem)

A PKC in which an entity's public identification information (unique name) plays the role of its public key.

Issues

Advantages:

-
-

If the wrong public user data is used, the cryptographic transformations fail.

Problem:

-

Private Key Generation

Answer:

-
-

Advantage:

Disadvantage:

Typical Application

- Users send encrypted messages using a public key derived from the recipient's ID and a **key validity period** (time stamp), using some publicly available function (e.g. converting concatenation of these two strings to appropriate length integer).
- Recipient requests the private key corresponding to a particular validity period from trusted authority

Incorporating a validity period into the public keys gives keys a lifetime, mitigating the problem of compromised keys (i.e. revocation).

Extreme solution: use a different public key for each message (unique time stamp instead of validity period).

Comparison to PKI

Both systems require a trusted third party. In ID-based cryptosystems, this authority always has access to the private keys.

In PKI, senders of messages / verifiers of signatures must obtain public keys of other users. In ID-based crypto, recipients / signers must obtain their own private keys.

Examples of ID Based Schemes

Signature Schemes

- Shamir (CRYPTO 1984) — based on RSA
- Feige, Fiat, and Shamir (J. Cryptology 1998) — based on computing square roots modulo pq (p and q large primes)

Encryption Schemes (good deal harder!)

- Boneh and Franklin (CRYPTO 2001) — uses the Weil pairing on elliptic curves.

  (first practical ID based encryption scheme)

Authentication

Today, **authentication** is arguably the most important application of cryptography. Three main classifications:

- **Data-origin authentication** (digital signatures) — covered previously
- **Entity authentication** (client-server, user-host, process-host)
- **Authenticated key establishment**

KDCs and PKIs make up a general key authentication framework.

- Protocols for ensuring *entity authentication* are required within these frameworks.
- See the Handbook of Applied Cryptography for some examples.