Cryptographic random and pseudorandom number generators

Jan Krhovják

Department of Computer Systems and Communications
Faculty of Informatics, Masaryk University
Brno, Czech Republic

Outline

- Random and pseudorandom data in cryptography
- Generating truly random data
  - in general purpose computer systems
  - in mobile environments
- Generating pseudorandom data
  - in general purpose computer systems
  - in mobile environments
- Statistical testing of randomness

Random and pseudorandom data in cryptography

- Random data in cryptography
  - Cryptographic keys, padding values, nonces, etc.
  - Quality and unpredictability is critical
- Generating truly random data
  - Based on nondeterministic physical phenomena
    - Radioactive decay, thermal noise, etc.
  - In deterministic environments extremely hard and slow
    - Only a small amount of random data in a reasonable time
- Generating pseudorandom data
  - Typically (in many computational environments) faster
  - Generated by deterministic algorithm
    - Short input (often called seed) – truly random data
    - Output – computationally indistinguishable from truly random data
- Quality of particular generators
  - Statistical testing & cryptanalytical methods

Random and pseudorandom data in cryptography

- Random data in cryptography
  - Cryptographic keys, padding values, nonces, etc.
  - Quality and unpredictability is critical
- Generating truly random data
  - Based on nondeterministic physical phenomena
    - Radioactive decay, thermal noise, etc.
  - In deterministic environments extremely hard and slow
    - Only a small amount of random data in a reasonable time
- Generating pseudorandom data
  - Typically (in many computational environments) faster
  - Generated by deterministic algorithm
    - Short input (often called seed) – truly random data
    - Output – computationally indistinguishable from truly random data
- Quality of particular generators
  - Statistical testing & cryptanalytical methods

Random and pseudorandom data in cryptography

- Random data in cryptography
  - Cryptographic keys, padding values, nonces, etc.
  - Quality and unpredictability is critical
- Generating truly random data
  - Based on nondeterministic physical phenomena
    - Radioactive decay, thermal noise, etc.
  - In deterministic environments extremely hard and slow
    - Only a small amount of random data in a reasonable time
- Generating pseudorandom data
  - Typically (in many computational environments) faster
  - Generated by deterministic algorithm
    - Short input (often called seed) – truly random data
    - Output – computationally indistinguishable from truly random data
- Quality of particular generators
  - Statistical testing & cryptanalytical methods

Random and pseudorandom data in cryptography

- Random data in cryptography
  - Cryptographic keys, padding values, nonces, etc.
  - Quality and unpredictability is critical
- Generating truly random data
  - Based on nondeterministic physical phenomena
    - Radioactive decay, thermal noise, etc.
  - In deterministic environments extremely hard and slow
    - Only a small amount of random data in a reasonable time
- Generating pseudorandom data
  - Typically (in many computational environments) faster
  - Generated by deterministic algorithm
    - Short input (often called seed) – truly random data
    - Output – computationally indistinguishable from truly random data
- Quality of particular generators
  - Statistical testing & cryptanalytical methods

Generating truly random data in computer systems

- Quality of TRNG is strongly dependent on source of randomness
  - Excellent sources – hardware-based
    - Built-in on-chip (Intel) or special add-on cards (Quantis)
  - Good sources – almost any input
    - Exact timing of keystrokes or exact movements of mouse
    - Microphone, video camera, or fluctuations in HDD access time
  - Acceptable sources – software-based
    - Process, network, or I/O completion statistics
  - Bad sources – easily predictable (with insufficient entropy)
    - System date and time, process ID, process runtime
- Problem of breaking or influencing TRNG
  - Partially solved by using digital postprocessing and statistical testing
- Problem of estimating entropy from particular physical event
  - No satisfactory solution exist

Generating truly random data in computer systems

- Quality of TRNG is strongly dependent on source of randomness
  - Excellent sources – hardware-based
    - Built-in on-chip (Intel) or special add-on cards (Quantis)
  - Good sources – almost any input
    - Exact timing of keystrokes or exact movements of mouse
    - Microphone, video camera, or fluctuations in HDD access time
  - Acceptable sources – software-based
    - Process, network, or I/O completion statistics
  - Bad sources – easily predictable (with insufficient entropy)
    - System date and time, process ID, process runtime
- Problem of breaking or influencing TRNG
  - Partially solved by using digital postprocessing and statistical testing
- Problem of estimating entropy from particular physical event
  - No satisfactory solution exist

Generating truly random data in computer systems

- Quality of TRNG is strongly dependent on source of randomness
  - Excellent sources – hardware-based
    - Built-in on-chip (Intel) or special add-on cards (Quantis)
  - Good sources – almost any input
    - Exact timing of keystrokes or exact movements of mouse
    - Microphone, video camera, or fluctuations in HDD access time
  - Acceptable sources – software-based
    - Process, network, or I/O completion statistics
  - Bad sources – easily predictable (with insufficient entropy)
    - System date and time, process ID, process runtime
- Problem of breaking or influencing TRNG
  - Partially solved by using digital postprocessing and statistical testing
- Problem of estimating entropy from particular physical event
  - No satisfactory solution exist

Generating truly random data in mobile environments

- Considerably different mobile computational environments
  - Mobile phones, personal digital assistants, cryptographic smartcards
  - New resources of randomness (information about current location)
- TRNG is typically located inside the integrated chip
  - For many one-chip devices (such as cryptographic smartcards) it is also the only solution
    - Design principles of these generators kept secret :-(
  - Mostly based on direct amplification of a noise source, jittered oscillator sampling
    - Certain level of correlation due to physical limitations, influencing
- Statistical testing of TRNG on Gemplus GemXpresso smartcards
  - Standard conditions – all selected statistical tests passed
  - Next research – influencing generators and new tests

Generating truly random data in mobile environments

- Considerably different mobile computational environments
  - Mobile phones, personal digital assistants, cryptographic smartcards
  - New resources of randomness (information about current location)
- TRNG is typically located inside the integrated chip
  - For many one-chip devices (such as cryptographic smartcards) it is also the only solution
    - Design principles of these generators kept secret :-(
  - Mostly based on direct amplification of a noise source, jittered oscillator sampling
    - Certain level of correlation due to physical limitations, influencing
- Statistical testing of TRNG on Gemplus GemXpresso smartcards
  - Standard conditions – all selected statistical tests passed
  - Next research – influencing generators and new tests

Generating truly random data in mobile environments

- Considerably different mobile computational environments
  - Mobile phones, personal digital assistants, cryptographic smartcards
  - New resources of randomness (information about current location)
- TRNG is typically located inside the integrated chip
  - For many one-chip devices (such as cryptographic smartcards) it is also the only solution
    - Design principles of these generators kept secret :-(
  - Mostly based on direct amplification of a noise source, jittered oscillator sampling
    - Certain level of correlation due to physical limitations, influencing
- Statistical testing of TRNG on Gemplus GemXpresso smartcards
  - Standard conditions – all selected statistical tests passed
  - Next research – influencing generators and new tests

Generating pseudorandom data in computer systems

- PRNG is deterministic finite state machine =>
  at any point of time it is in a certain internal state
  - PRNG state is secret (PRNG output must be unpredictable)
  - PRNG (whole) state is repeatedly updated (PRNG must produce different outputs)
- Secret state compromise may occur – recovering is difficult
  - Mixing data with small amounts of entropy to the secret state
  - Problem is limited amount of entropy between two requests for pseudorandom data (solution is pooling)
- Basic types of PRNGs utilize
  - LFSR, hard problems of number and complexity theory, typical cryptographic functions/primitives
- A lot of different (and possibly unsecure) modifications of PRNGs are implemented in many applications

Generating pseudorandom data in computer systems

- PRNG is deterministic finite state machine =>
  at any point of time it is in a certain internal state
  - PRNG state is secret (PRNG output must be unpredictable)
  - PRNG (whole) state is repeatedly updated (PRNG must produce different outputs)
- Secret state compromise may occur – recovering is difficult
  - Mixing data with small amounts of entropy to the secret state
  - Problem is limited amount of entropy between two requests for pseudorandom data (solution is pooling)
- Basic types of PRNGs utilize
  - LFSR, hard problems of number and complexity theory, typical cryptographic functions/primitives
- A lot of different (and possibly unsecure) modifications of PRNGs are implemented in many applications

Generating pseudorandom data in computer systems

- PRNG is deterministic finite state machine =>
  at any point of time it is in a certain internal state
  - PRNG state is secret (PRNG output must be unpredictable)
  - PRNG (whole) state is repeatedly updated (PRNG must produce different outputs)
- Secret state compromise may occur – recovering is difficult
  - Mixing data with small amounts of entropy to the secret state
  - Problem is limited amount of entropy between two requests for pseudorandom data (solution is pooling)
- Basic types of PRNGs utilize
  - LFSR, hard problems of number and complexity theory, typical cryptographic functions/primitives
- A lot of different (and possibly unsecure) modifications of PRNGs are implemented in many applications

Generating pseudorandom data in computer systems

- PRNG is deterministic finite state machine =>
  at any point of time it is in a certain internal state
  - PRNG state is secret (PRNG output must be unpredictable)
  - PRNG (whole) state is repeatedly updated (PRNG must produce different outputs)
- Secret state compromise may occur – recovering is difficult
  - Mixing data with small amounts of entropy to the secret state
  - Problem is limited amount of entropy between two requests for pseudorandom data (solution is pooling)
- Basic types of PRNGs utilize
  - LFSR, hard problems of number and complexity theory, typical cryptographic functions/primitives
- A lot of different (and possibly unsecure) modifications of PRNGs are implemented in many applications

Generating pseudorandom data in mobile environments

- Mobile environments are very different
  - Lack of resources as CPU speed, memory, energy
  - Design of PRNGs should be dependent on type of device
- PRNG implementations on GemXpresso JavaCards
  - PRNG 1 is ANSI X9.17/31; PRNG 2 is FIPS 186-2
  - Very slow (see below) – inappropriate for normal use
- Amount of generated data after one hour

	GXPLite-Generic	GXPPro-R3	GXP211 PK IS	GXP211 PK
TRNG	7973 KB	3619 KB	495 KB	492 KB
PRNG 1	333 KB	151 KB	43 KB	39 KB
PRNG 2	49 KB	33 KB	13 KB	13 KB

- All selected statistical tests also passed
  - Low performance of software implementations => tests only with PRNG 1 on GXPLite-Generic

Generating pseudorandom data in mobile environments

- Mobile environments are very different
  - Lack of resources as CPU speed, memory, energy
  - Design of PRNGs should be dependent on type of device
- PRNG implementations on GemXpresso JavaCards
  - PRNG 1 is ANSI X9.17/31; PRNG 2 is FIPS 186-2
  - Very slow (see below) – inappropriate for normal use
- Amount of generated data after one hour

	GXPLite-Generic	GXPPro-R3	GXP211 PK IS	GXP211 PK
TRNG	7973 KB	3619 KB	495 KB	492 KB
PRNG 1	333 KB	151 KB	43 KB	39 KB
PRNG 2	49 KB	33 KB	13 KB	13 KB

- All selected statistical tests also passed
  - Low performance of software implementations => tests only with PRNG 1 on GXPLite-Generic

Generating pseudorandom data in mobile environments

- Mobile environments are very different
  - Lack of resources as CPU speed, memory, energy
  - Design of PRNGs should be dependent on type of device
- PRNG implementations on GemXpresso JavaCards
  - PRNG 1 is ANSI X9.17/31; PRNG 2 is FIPS 186-2
  - Very slow (see below) – inappropriate for normal use
- Amount of generated data after one hour

	GXPLite-Generic	GXPPro-R3	GXP211 PK IS	GXP211 PK
TRNG	7973 KB	3619 KB	495 KB	492 KB
PRNG 1	333 KB	151 KB	43 KB	39 KB
PRNG 2	49 KB	33 KB	13 KB	13 KB

- All selected statistical tests also passed
  - Low performance of software implementations => tests only with PRNG 1 on GXPLite-Generic

Generating pseudorandom data in mobile environments

- Mobile environments are very different
  - Lack of resources as CPU speed, memory, energy
  - Design of PRNGs should be dependent on type of device
- PRNG implementations on GemXpresso JavaCards
  - PRNG 1 is ANSI X9.17/31; PRNG 2 is FIPS 186-2
  - Very slow (see below) – inappropriate for normal use
- Amount of generated data after one hour

	GXPLite-Generic	GXPPro-R3	GXP211 PK IS	GXP211 PK
TRNG	7973 KB	3619 KB	495 KB	492 KB
PRNG 1	333 KB	151 KB	43 KB	39 KB
PRNG 2	49 KB	33 KB	13 KB	13 KB

- All selected statistical tests also passed
  - Low performance of software implementations => tests only with PRNG 1 on GXPLite-Generic

Statistical testing of randomness

- Based on statistical hypothesis testing
  - Each statistical test is based on some function of data (test statistic)
    - Expected value of test statistic is known for the reference distribution
    - Generated random stream is subjected to the same test
  - No set of such tests can be considered as complete
    - Some of them are accepted as the de facto standard
    - Well-known test batteries are NIST and DIEHARD
  - Correct interpretation of empirical results should be very tricky
- NIST test battery – two approaches of evaluation results
  - Examination of the proportion of sequences that pass the test
  - Check for uniformity of the distribution of P-values
    - Chi-square goodness-of-fit test
  - If either of them fails => new experiments with different sequences
- We tested always 50 MB sequence divided to 100/400/800 subsequences with level of significance 0.01

Statistical testing of randomness

- Based on statistical hypothesis testing
  - Each statistical test is based on some function of data (test statistic)
    - Expected value of test statistic is known for the reference distribution
    - Generated random stream is subjected to the same test
  - No set of such tests can be considered as complete
    - Some of them are accepted as the de facto standard
    - Well-known test batteries are NIST and DIEHARD
  - Correct interpretation of empirical results should be very tricky
- NIST test battery – two approaches of evaluation results
  - Examination of the proportion of sequences that pass the test
  - Check for uniformity of the distribution of P-values
    - Chi-square goodness-of-fit test
  - If either of them fails => new experiments with different sequences
- We tested always 50 MB sequence divided to 100/400/800 subsequences with level of significance 0.01

Statistical testing of randomness

- Based on statistical hypothesis testing
  - Each statistical test is based on some function of data (test statistic)
    - Expected value of test statistic is known for the reference distribution
    - Generated random stream is subjected to the same test
  - No set of such tests can be considered as complete
    - Some of them are accepted as the de facto standard
    - Well-known test batteries are NIST and DIEHARD
  - Correct interpretation of empirical results should be very tricky
- NIST test battery – two approaches of evaluation results
  - Examination of the proportion of sequences that pass the test
  - Check for uniformity of the distribution of P-values
    - Chi-square goodness-of-fit test
  - If either of them fails => new experiments with different sequences
- We tested always 50 MB sequence divided to 100/400/800 subsequences with level of significance 0.01

Conclusion

- Generators of random and pseudorandom data in cryptography
  - Acceptable designs in general purpose computer systems
  - Situation is more serious in mobile environments
- Three basic areas of our research (focused on mobile environments)
  - Identification and analysis of new sources of randomness
    - Information about location, signal characteristics, etc.
  - Design of principles of new pseudorandom number generators
    - With respect to particular mobile environments
  - Design/modification of password-based cryptographic protocols

Conclusion

- Generators of random and pseudorandom data in cryptography
  - Acceptable designs in general purpose computer systems
  - Situation is more serious in mobile environments
- Three basic areas of our research (focused on mobile environments)
  - Identification and analysis of new sources of randomness
    - Information about location, signal characteristics, etc.
  - Design of principles of new pseudorandom number generators
    - With respect to particular mobile environments
  - Design/modification of password-based cryptographic protocols