The Sources of Randomness in Smartphones with Symbian OS

Jan Krhovják, Petr Švenda, Vašek Matyáš, Luděk Smolík

{xkrhovj, xsvenda, matyas}@fi.muni.cz, 169680@mail.muni.cz

Faculty of Informatics
Masaryk University
Brno, Czech Republic

Abstract

This paper is focused on the analysis of randomness sources available in current mobile phones. We discuss specifics of the mobile devices, identify potential sources of randomness, and define types of possible attackers. An analysis of a camera and a microphone input noise as a source of randomness has been performed and amount of available entropy estimated. We then also statistically tested the obtained data using NIST test battery. Preliminary version of entropy extraction function was implemented using cryptographic hash function and its performance tested on several mobile phones.

Keywords: CCD, CMOS, entropy, mobile device, smartphone, source of randomness, Symbian.

1 Introduction

This paper deals with issues related to the generation of truly random data (i.e., bits, numbers, and sequences) in mobile computing environments. We will describe the expected application requirements in terms of amount and speed of random data generation in such environments, and then we will focus on available sources of randomness in mobile devices with the Symbian OS. The main goal of this work is identification of such sources, evaluation of their acquisition speed, statistical testing of their quality, and estimation of the amount of available entropy in a given time period. Some issues described in this paper were partially discussed in Czech article [1].

Our work will cover the possibilities of random data generation from both external and internal environmental characteristics. The examined external sources will be audio and video streams that can be captured using a mobile device. Additional external sources of location-based information like actual strength of the network signal or fluctuation in current GPS position will also be examined. With respect to the internal sources, we will investigate the information accessible through the application programming interfaces of the Symbian OS like the actual battery charge level and other accessible system statistics.

All identified sources of randomness are tested on the Nokia N73 smartphone with respect to performance and statistical quality of the output. The performance tests should reveal a practical usability of particular sources of randomness for the concrete applications. Statistical testing of randomness should then reveal whether the output sequences does not suffer from known statistical defects and have appropriate statistical properties. We expect that entropy extraction or at least digital post-processing techniques will be necessary to gather a good quality (and unpredictable) truly random data, as some of the generators may produce a large amount of data with only a small amount of entropy.

2 Basics of random number generation

The overwhelming majority of commonly used cryptosystems is designed in accordance with the Kerckhoff's principle – i.e., the security of cryptosystem must be based on secrecy of the cryptographic keys and not on the cryptosystem itself. This allows anybody to perform independent analysis of particular cryptosystems (ideally also alongside with the verification of their source codes) and on the contrary, the complete security is reduced to secrecy of private/secret keys. The quality and unpredictability of random data (i.e., bits, numbers, and sequences) that are the basis for cryptographic keys is therefore critical. Random data are in addition to generating of cryptographic keys also used for other cryptographic operations – e.g., as initialization vectors, various pad values (padding), random challenges (nonces) in cryptographic protocols, in process of digital signing (per-message secrets), and for masking data and prevent dangerous side-channel attacks.

In general, two kinds of generators can be distinguished – the true random number generator and the pseudorandom number generator. The former is typically based on nondeterministic physical phenomena (e.g., radioactive decay or thermal noise), while the latter is only a deterministic algorithm where all randomness of the output is fully dependent on the randomness of the input (often called seed). Getting truly random data in the deterministic environment of computer systems is extremely hard and slow (i.e., only a small amount of good quality random data can be generated in a reasonable time), therefore we often restrict ourselves to the use of deterministically generated pseudorandom data. Generating pseudorandom data is typically (in most environments) faster and truly random data are used in this process only as an input. Since the whole generating process is deterministic, the randomness of the output is fully dependent on the randomness of the input. Many classes of pseudorandom number generators (designed, e.g., for simulation purposes or randomized algorithms) exist, but the goal of a pseudorandom number generator in cryptography is production of pseudorandom data that are computationally indistinguishable from truly random data. The goal of cryptanalysis is to prove the converse.

Randomness is a probabilistic property, therefore verification of the statistical quality of (pseudo)random data by detecting deviations from true randomness (known a priori) is based on statistical testing. These tests may be useful as the first step in determining whether or not a generator is suitable for a particular cryptographic application. However, no set of statistical tests can absolutely certainly point out a generator as appropriate for usage in a particular application, i.e., statistical testing cannot serve as a substitute for cryptanalysis. In addition, the results of statistical testing must be interpreted with some care and caution to avoid incorrect conclusions about a specific generator. Most commonly used statistical test suites are DIEHARD [2] and NIST [3].

2.1 True random number generators

When designing true random number generators for cryptographic purposes, the problem of good randomness source selection must be addressed (see [4] or [5]). Excellent sources (with respect to unpredictability) are based on some kind of physical phenomena and in a typical environment of general purpose computer systems utilize some additional hardware – for example on-chip built-in true random number generator (solutions from Intel [6] or VIA [7]) or specially designed add-on card with true random number generator (solutions from ID Quantique [8]). Good sources of randomness include, e.g., the exact timing of keystrokes and exact movements of mouse. Other possible good sources are video camera (focused for example on a lava lamp [9]), microphone, or fluctuations in hard disk access time [10].

Still acceptable sources of randomness are process statistics, network statistics, I/O completion statistics, etc. However, these could be under certain circumstances subjects of influence (i.e., could be predictable) and for cryptographic purposes are thus inappropriate – they should be always combined with sources described above. Almost worthless sources of randomness are then for example system date and time or process runtime. A cautionary example is the true random number generator in the Netscape implementation of SSL protocol that used as randomness sources only the time of a day, the process ID, and the parent process ID. Thus, an adversary who can estimate these three values can simply apply the well-known MD5 hash algorithm to compute the generated seed for the pseudorandom number generator [11].

All problems relevant to random sources and randomness are more serious in considerably different mobile computing environments – this area is currently most technically challenging. In mobile computing environments many sources described above do not exist at all, and as far as we know, some kind of hardware true random number generator located inside the integrated chip is typically used for generation of truly random data. It is also the only solution for all single-chip devices such as smartcards, microcontrollers, or RFID tags (utilized, e.g., in electronic passports). Unfortunately, a design of these generators is mostly kept secret due to classified nature of most research in this field. The situation here is in fact very similar to the situation of concealing cryptographic algorithms in the seventies of the last century. These generators are, according to the unofficial information, based typically on the frequency instability of several free running oscillators.

On the contrary, mobile devices bring also a possibility to utilize new potential sources of randomness – for example information about current location, strength of transmitted signal, or battery level. Suitability of these and other possible sources of randomness for cryptographic purposes and their influencing are a subject of our ongoing research.

2.2 Pseudorandom number generators

The problem of insufficient speed of true random number generators is typically solved by a pseudo-random number generator, whose output should be for cryptographic purposes computationally indistinguishable from truly random data. The pseudorandom number generator is in fact a deterministic finite state machine, which implies that it is at any point of time in a certain internal state. This pseudorandom number generator state is secret (since the output must be unpredictable) and during the generation of pseudorandom data is repeatedly updated (as different outputs must be produced). Since the whole generating process is deterministic, the measure of entropy in output sequences is limited to the entropy in truly random input sequence (seed). Compromised seed or internal state typically leads to degradation of security of output sequence.

Of course, careful design of generators should expect that a secret state compromise may occur. Recovering a pseudorandom number generator with a compromised state is a difficult task which is typically based on mixing data with small amounts of entropy to the secret state. However, if the amount of entropy between two requests for pseudorandom data is limited, the attacker can simply make frequent requests for pseudorandom data and by exhaustive search obtain a secret state. The best solution how to prevent this problem is to pool (i.e., collect) incoming entropy to sufficient amount, and then to mix it to the secret state [12]. The generator should stop producing outputs if there are no data with sufficient amount of entropy in the pool.

Basic types of pseudorandom number generators utilize linear feedback shift registers (e.g., their non-linear combination), hard problems of number and complexity theory (e.g., a problem of factorization), and typical cryptographic functions/primitives (e.g., DES, SHA-x). Only in the last category of generators mechanisms to recovery from state compromise are used (good examples of such generators are Yarrow-160 and Tiny). However, existing pseudorandom number generators are often very slow (even if some cryptographic functions are hardware accelerated) due to the lack of resources in mobile computing environments.

2.3 Application requirements

The applicability of commonly used pseudorandom number generators (designed for general purpose computer systems) in mobile devices due the speed and quality issues can be quite problematic or nearly impossible. Namely their speed is for truly random number generators strongly related to the quality and for pseudorandom number generators is dependent on particular type of a mobile device. Unfortunately, the amount of required (pseudo)random data is also strongly dependent on particular applications and need not to be as small as one could expect.

Consider, for example, a user that uses only classical data services of mobile phones or personnel digital assistants. This user can use a special application to perform a login to remote services (e.g., mobile banking) that require confidentiality and integrity assurance of all subsequent communication. Application requirements on speed and the quality of generated (pseudo)random data are dependent on their expected usage. The high-quality (pseudo)random data should be used as basis for creating long-term cryptographic keys. The advantage is that these keys are typically generated once upon a time (definitely not in intervals shorter then one month) and their generating process is not limited in time. The length of these long-term keys is typically in the order of hundreds or thousands bits, and is dependent on particular cryptosystems. The speed of a generator can be thus less than one hundred bytes per second and the generating can take several seconds (or even minutes).

Considerably different situation (mainly with respect to the speed) arises in the case of initialization vectors and random padding values. Their length is also in the order of hundreds of bits, but achieving a permanent secure communication requires their frequent change (and thus re-generation). Moreover, the requirements on random data are rising up with increasing transfer rates – in the case of mobile devices their maximum is typically limited by the maximum uplink transfer rate. In practice the transfer rates of EDGE (with configuration of slots 3+2) are approximately 200 kb/s for downlink and 100 kb/s for uplink, the transfer rate of Bluetooth is approximately 3 Mb/s (in both directions), and similarly for WiFi up to 108 Mb/s.

However, the discussion above does not cover actual requirements on a generator speed – only gives us an upper bound. These requirements are dependent on the application that can be categorised as interactive (e.g., transfer of voice or video), semi-interactive (e.g., web-based services), and non-interactive (e.g., one-time file transfer or sending e-mail). The data can be thus encrypted and transferred (according to these categories) immediately after creating/filling outgoing packet (to prevent unwanted delays), or after reaching other pre-specified sizes. A new initialization vector (and often also padding) is required for each encryption. Splitting data to several independent pieces thus imply higher requirements on initialization vectors (and padding). The consequence is higher requirements on (pseudo)random data.

2.4 Specifics of the mobile devices

Mobile devices are different from general purpose computers and this also influences the process of generating of (pseudo)random data. The possibility to change environment of mobile devices is definitely a great advantage given by the mobility nature of the device. The existence of several embedded input devices as microphone, radio receiver, video camera, or touchable display is another great advantage. On the contrary, the mobility and the small physical size of device brings also a higher possibility of a theft or (temporary) lost with potential compromise of generator. The important assumption of high-quality secure generator design is thus the impossibility of deducing the inner state of generator and fast recovering of its entropy level. Other disadvantages can be low performance (CPU frequency is in the best smartphones roughly 200MHz) and restricted random access memory size (in the order of tens of MB).

A well-designed generator must also have a sufficient amount of entropy – shortly after turning device on, after letting device out of sight, and after intensive generating of random numbers. This non trivial task may require employment of the energetically costly sources of randomness (e.g., video camera) and/or the user contribution. These sources can be required to assure higher security of generated data – e.g., for mobile banking purposes. It can be expected that the user will be more tolerant to the delay in such special cases (caused by his involving to the generating process) than in the case of normal voice encryption (where is necessary the short response time).

2.4.1 Available sources of true randomness

There are several good potential sources of randomness in the mobile devices. All of them can be used for generating truly random data and categorised as follows:

1. The external environment – the sources located in the proximity of a user. They can be used to generate random data and their characteristics (samples) can be obtained by the different phone input devices. The typical examples of this characteristic can be the location (Location API JSR-179), photography, sound/video recording (Mobile media API JSR-135), or strength of incoming signal from accessible base transceiver stations (BTSs).
2. The internal environment – the sources inside the mobile device that are inaccessible for a remote attacker. The example is the execution of computational operations, access time to memory, or true random number generator on the SIM card (typically cryptographic chip card Gemplus, Philips, Schlumberger …). Clearly, the processes performed inside the phone will be always (to a certain level) dependent on the external environment. We will expect that this dependency (that can lead to influencing of this kind of sources) is very complex and cannot be by the attacker observed or even misused. Typical example is the noise originated in the microphone, CCD chip or A/D converter. However, it is extremely important to make precise categorisation of particular sources. For example, the microphone in hands-free sets should not be considered as an internal source. A poorly performed

Bluetooth pairing process can lead to the eavesdropping of digitalised voice communication including the noise of such microphone.

3. Interaction with the surrounding environment – the sources that are highly influenced by the feedback from the external environment. Typical example is sound reverberation, the periodical changes in battery level (depends also on temperature or strength of signal), or fluctuations in the incoming signals.

Important characteristics of all these sources are: availability in all environments, dynamic in the time, unpredictability by the attacker, and uninfluenceability by the attacker. The characteristic from the internal environment (e.g., some kind of noise) cannot be definitely influenced by an attacker as much as in other cases (e.g., external signals or images). In the following part we focus on the opportunities of an attacker to influence the generator.

2.4.2 Attacker models

Attacker capabilities can be categorised as follows:

1. Weak attacker – is expected to have the same equipment as the legitimate user (including phone model, type of SIM card, etc.). Moreover, he knows the current state of user's environment (he stays very close to the user or records user's environment to the camera). This type of attacker is also capable to slightly influence the user's generator (e.g., by disturbing the signal, blind a phone camera by intensive ray of light, or force the microphone membrane to resonating on the same frequency).
2. The average attacker – has the same capabilities as the weak attacker. Moreover, we also expect that he had temporary access to the user's device and thus he knows the actual inner state of a device. These facts imply that he is able to construct the device that is exactly same (including the inner state) as the user's mobile device.
3. The strong attacker – has the same capabilities as the average attacker. Moreover, he is (with the exception of phone interaction with the surrounding environment) capable to fully control and modify the user's environment.

Since the strong attacker can try influencing almost all these sources, then the following fundamental questions arise: Is the user able to observe an ongoing attack? Can he stop the generator during the observed attack? Can be the generator protected against malicious modification of generator (or software) in the device? Such difficult tasks cannot be definitely solved easily. However, the integrated Trusted Platform Module [13] and support for Digital Rights Management (OMA DRM API) services could be utilized in modern smartphones at least to protect installed firmware/software integrity.

3 Analysis of selected sources of randomness

This part of our paper deals with issues related to the practical experiments performed on smartphone Nokia N73. The goal of these experiments was to assess quality of selected randomness sources in one particular mobile device and to estimate the amount of randomness (entropy) in these sources.

Due to the API restrictions, we were forced to drop sources like battery level, signal strength or GPS position. Currently, it is not possible to obtain source output with a sufficient precision (e.g., battery and signal provides scale into the integer between 0 and 10) or frequency (e.g., external GPS provides only one measurement per second with fluctuations typically only in two least significant bits). So far, we did not focus on internal sources like fluctuations in the execution speed, time to accomplish a drawing task, etc.

3.1 Theoretical entropy estimation

The basic measure for randomness is in information theory often called *uncertainty* or *entropy* and is typically defined [14] as:

$$H_1(X) = -\sum_{x \in X} P_X(x) \log P_X(x),$$

where the sample *x* is drawn from random distribution *X* with probability *P_X(x)*. The logarithm base typically correspondents to the unit of measured information – in information theory base 2 is often used and that implies that the unit will be bits. This entropy measure is often referred as Shannon entropy or alternatively information entropy.

Unfortunately, Shannon entropy may be inappropriate for our purposes, since it is in fact only average case entropy. We cannot do any assumptions about distributions formed by our sources of randomness. The problem is that the attacker can simply force the source of randomness to produce the most probable values that contains minimum entropy. To cope with this situation the min-entropy measuring the worst case entropy is often used (especially in the theory of randomness extractors). It is defined as:

$$H_\infty(X) = \min_{x \in X}(-\log P_X(x)) = -\log(\max_{x \in X} P_X(x)),$$

where the sample *x* is drawn from random distribution *X* with probability $P_X(x)$. It can be easily seen that min-entropy is always less then or equal then Shannon entropy (the tight example is for uniform distribution).

These two entropy measures are only special cases of generalised so-called Rényi entropy [15] that is defined as:

$$H_\alpha(X) = \frac{1}{1-\alpha} \log \sum_{x \in X} (P_X(x))^\alpha,$$

where the sample *x* is drawn from random distribution *X* with probability $P_X(x)$ and α is a parameter. Evidently, the higher α implies the smaller resulting entropy. Rényi entropy converges to Shannon entropy for α limitary close to *1* and to min-entropy for α limitary close to ∞ (if such limits exist).

3.2 Microphone input

An embedded or hands-free microphone is used as a voice input device in mobile devices. Almost all commonly used microphones are typically based on an oscillating membrane and some mechanisms that transform the oscillation to the voltage representing particular signal elements. Supported sampling frequency, modulation method, and number of bits used for representing the value of one sample are the most important parameters of such devices. The Nokia N73 smartphone uses 16-bit pulse coded modulation (a signed PCM) at the frequency 8000 Hz for sampling a sound wave – the data throughput is thus 16000 B/s. We restrict ourselves only to the small number of 204800 samples that corresponds to 25.6 seconds of sound due to memory restrictions of the device.

Of course, each microphone has a different characteristic (e.g., due to different solidity of membrane or other manufacturing differences). Our goal is to estimate the amount of entropy in the input sound signal captured by the microphone. We focus mainly on measuring min-entropy of the noise originated in the microphone.

We used the fast Fourier transformation (FFT) algorithm to compute a discrete Fourier transform (DFT) for analyzing the basic frequency components present in the noise. We performed this analysis of the embedded microphone on the sound sample of both recorded music (figure 1) and noise (figure 2). Moreover, we analysed also the hands-free microphone on the sound sample of noise (figure 3). We are obviously interested only in the spectrum of frequencies between 0 and 4000 Hz (for sampling rate 8000 Hz it is so-called Nyquist frequency). Ideal noise is expected to have all frequencies uniformly presented. This preliminary results shows that there are significant differences in the observed spectrums of the noise. The hands-free microphone appears to be more sensitive than the embedded one.

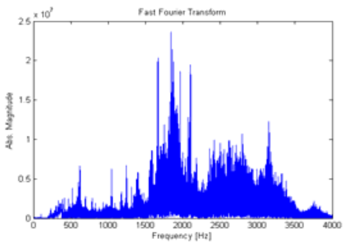

Figure 1: Frequency spectrum of the recorded music sample (embedded microphone).

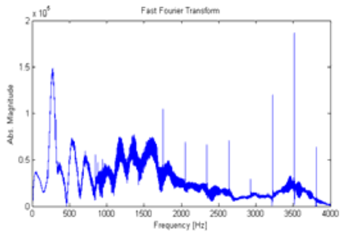

Figure 2: Frequency spectrum of the recorded noise sample (embedded microphone).

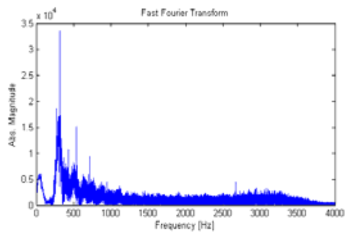

Figure 3: Frequency spectrum of the recorded noise sample (hands-free microphone).

Finally, we analysed histograms of all these recordings. They have approximately normal distribution and the actual minimum/maximum values are –32764/32767, –14220/8856, and –347/574, respectively. However, especially in the case of noise, these numbers are strongly influenced by the peak on the beginning of each recording. Much better values –12/13 and –9/8 can be obtained after removing this peak. We can make a very preliminary estimation that at most $\log_2 8=3$ bits of entropy can be presented in each sample.

However, these samples are definitely not independent and the detailed analysis of this source and the statistical quality testing will be the subject of our next research.

3.3 Camera input

Digital optical input devices (e.g., cameras, microscopes, or scanners) can be based on several different silicon sensors [16]. All of them use the array of semiconductor photo-sensors to transfer an accumulated electric charge to a voltage. The older and the most famous is the charge-coupled device (CCD) chip invented in late sixties. This chip utilizes only one amplifier for the whole array and the sequential processing in parallel register is thus quite slow. The complementary-metal-oxide-semiconductor (CMOS) chip uses more amplifiers and allows faster parallel processing. Unfortunately, each amplifier is typically slightly different and this results in the higher level of amplifier noise. The electron multiplying charge coupled device (EMCCD) and the image intensified CCD (ICCD) chips are even more sophisticated and provide in the former case better sensitivity and in the latter case shorter exposure times.

Digital cameras based solely on CCD (e.g., Sony-Ericsson S700i) or CMOS (e.g., Nokia N* series) sensors (for simplicity will be denoted as *optical sensors*) are used in current computer systems and mobile phones. As we will mention in next paragraphs – these optical sensors are influenced by thermal noise, they have problems with vignetting, blooming, sensitivity to some colours, etc. Some of these problems are solved only by purely software means (that are kept secret) and this make the entropy estimation (fortunately not extraction) harder.

3.3.1 Camera view finding noise

The significant part of the current mobile devices (cell phones, smart phones and PDA) is equipped with the build-in camera that can be used for an entropy gathering. For camera input, we reason, that entropy should be extracted from an optical sensor noise during view finding rather than from the high-resolution picture of the surrounding environment.

The lower-quality optical sensors (often used in mobile devices) have generally higher noise presence than sensors in high-end cameras. The noise is typically unwanted for almost all applications – with the exception of the random number generation process. The optical sensor white noise should be always present, but its actual level may depend on physical conditions – namely the temperature. We performed the practical experiment with Nokia N73 camera within temperatures 5 °C to 45 °C and concluded that an inside decrease of the noise towards lower temperature can be detected. However, this noise is still significantly present to provide enough entropy.

The data output from view finding is more suitable source than a high-resolution post-processed output (after software noise reduction and compression process). Data acquisition is also much faster – commonly between 10–15 frames per second – and has more suitable size for the additional processing: a single frame with 240×180 pixels can be fully stored and processed in RAM memory (~130kB) which is unlikely for high-resolution picture.

The input source is available even when the camera cover is closed or lens is covered with a finger. This is both convenient and useful – it serves as an important defence against the active attacker that illuminate the sensor. An overexposure of the sensor can be caused, for example, by an intense light like halogen lamp (see figure 6). All colour components within exposed area are stir up to maximum value (255) and thus all possible entropy obtainable from (not only white) noise is effectively removed. A similar effect can be also caused by direct sunlight. The closed lense cover should serve as a prevention of such unintentional degradation during random data generation.

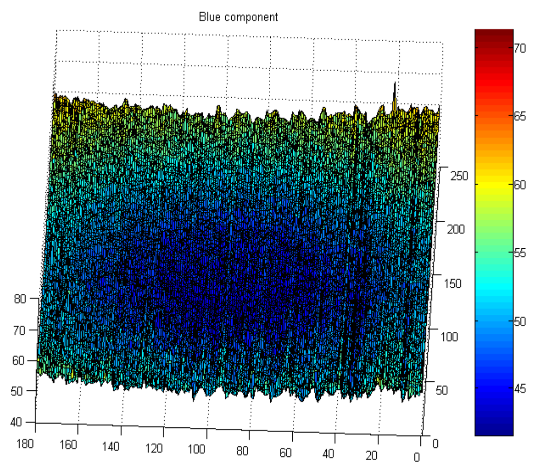

Figure 4: The average value of the blue color component over the whole camera's frame with closed cover.

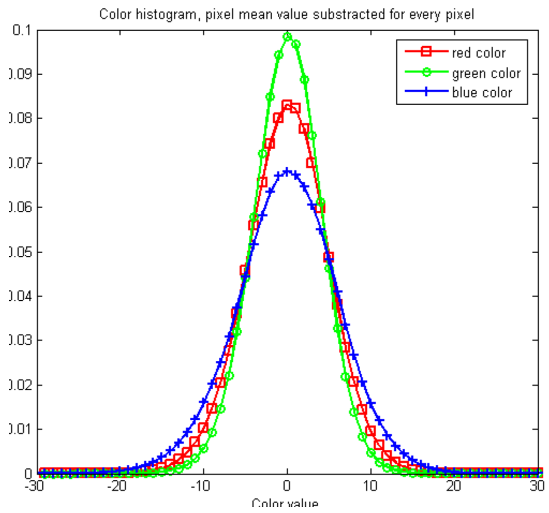

Figure 5: The intensity histogram for each color component after remove of the pixel post-processing effects by normalization.

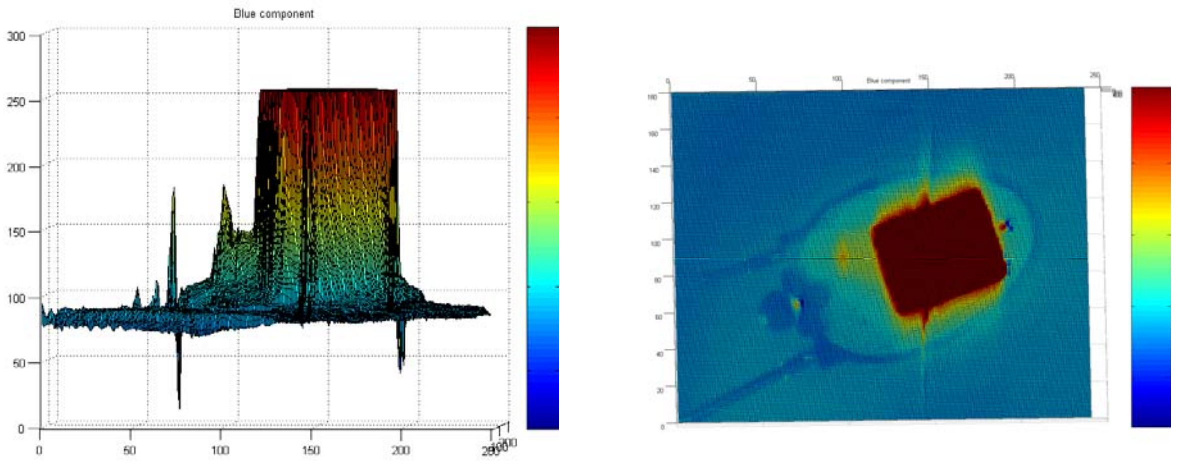

Figure 6: A visualized overexposure caused by the halogen lamp (blue color component).

3.3.2 Camera input entropy estimation

In this section, we would like to estimate the amount of entropy extractable from a camera input. We proceed as follows:

1. A custom application for storing large amount of frames produced by mobile device's camera into removable memory card was developed. The systematic defects of the input due to post-processing were examined using visualization and statistical tools with captured data transferred to PC.
2. A noise dependency on the surrounding temperature was experimentally measured and evaluated.
3. The correlation between neighbour pixels, pixels in the same row and column was computed as well as autocorrelation and fast Fourier transformation of values from single pixel in time (subsequent frames) was computed.
4. The distribution of values for separate colour components was computed for temperature with the least noise within device operation value (5 °C) and entropy was estimated based on Shannon entropy and min-entropy formulas.
5. A simple technique for entropy extraction (with least possible post-processing) was implemented and resulting binary stream was tested with suite of NIST randomness test to challenge its statistical properties.
6. Practical tests of maximal data throughput of SHA-1 hash function on mobile devices with Symbian OS and JavaME were performed. Preliminary entropy extraction function using SHA-1 was implemented.

The systematic defects introduced into camera frame due to post-processing and optical sensor technology (like row-dependent readouts) are clearly visible from figure 4 when the average camera input with a closed lense cover is visualized. There are hot pixels around borders, significant rips in the rows, centred circle rips and significantly different intensity towards centre of the frame. Especially, blue colour component shows visible row-dependent rips (caused by the read-out technology) and red colour component has significantly increased value towards chip borders. This effect is probably caused due to camera post-processing (out of our control) to balance light drop due to different lens mass towards borders and may lead to suspicion, that some pixels are systematically correlated. However, when values for each single pixel are normalized by subtracting mean of the pixel in longer time period, most of the systematic effects vanish out. The colours histograms of normalized input taken over 2000 subsequent frames are depicted on figure 5 (histograms are centred to 0). All colours exhibits Gauss-like distribution, blue colour providing more entropy (in Shannon-sense) than other colour components. The presented frames were taken with temperature around 5 °C.

Well documented property of the optical sensors is dependency of noise on temperature – the noise component should be reduced by lower temperatures (e.g., under-cooling of chip by liquid nitrogen). As the common recommended temperature operation range for mobile devices with LCD display is between –15 to 45 °C, we choose to test the noise presence when a plastic cover of lens has a temperature of 5 °C and 45 °C (precise measurement of optical sensor was not possible). The detected differences are not too dramatic and we can expect slightly decreased (but still comparable) amount of entropy for the lower temperatures. For entropy estimation, we used noise obtained from measurement with 5 °C. In the common camera devices the raw data from the optical sensor are not directly accessible to the user but at first they underlay to a few correction steps which are usually incorporated into the firmware of the optical sensor. Such corrections take care of couple of different systematic effects which are partially inherent to the optical sensor working principle and partially occur as fabrication tolerances during the manufacturing.

For an overall entropy estimation, we have to know the number of independent (uncorrelated) pixels in one frame and whether values of single pixel are independent between two or more subsequent frames (or which frames has to be discarded to obtain uncorrelated values (e.g., pixel value from every 3rd frame) and finally estimate the expected entropy provided by single independent pixel.

The Matlab *corrcoef* cross-correlation function was used to verify whether neighbouring pixels and pixels in the same row and column are independent. No statistically significant correlation was found on significance level set to 0.05. To further verification of results obtained from cross-correlation, we generated streams of a binary data from red, green and blue colour component and tested those streams with the NIST battery. See section 3.2.3 for details.

The camera view finding mode on Nokia N73 provides us with 12 frames per second and thus 12 different values per single pixel (10–15 frames for other cameras). The key question for entropy estimation is the independency of consequent values of that pixel. The auto-correlation tests were performed with the vector containing the values taken in time from single pixel. A statistically significant deviations from the characteristics of white noise (where significant correlation value is present only for *lag = 0* and can be omitted otherwise) were not detected. This implies that the consequent frames should be independent. All view finding pixels (180×240) were separately tested within a sequence of 2100 frames. Fast Fourier transformation was applied on the same vector (values of single pixel in time) to detect dominant frequencies, if there are any. FFT provides almost uniform output (no dominant frequency) for all tested pixels and thus confirms independence of values between frames from auto-correlation test. Note that as view finding provides 12 frames per second, only frequencies between 0 to 6 Hz can be tested by FFT.

Final step is to compute the amount of entropy carried by the noise in a single (independent) pixel. As it can be seen from figure 5, histograms of all colour components follow the common Gauss distribution without significant deviations. Overall entropy (see table 1) of camera view finding source can be computed using simple formula:

$$E = \mathit{IndependentPixelsPerFrame} * \mathit{IndependentFramesPerSecond} * \mathit{EntropyPerPixel}\ [\mathit{bits/s}]$$

	Single pixel (Shannon)	Single pixel (Minimal)	Entropy tested by NIST battery (bits/sec)
Red color	3.9203	3.1979	50400
Green color	4.0373	3.3277	50400
Blue color	4.7608	3.9276	50400

Table 1: An entropy estimation according to Shannon and min-entropy formulas for a single independent pixel. Note, that extrapolation to whole input (180×240 pixels/frame, 12 frames/second) is valid only if the pixels and frames are independent (see 3.3.2 for detailed discussion and performed tests).

3.3.3 Statistical testing with NIST battery

As described in previous section, we also wanted to verify the statistical properties of binary stream extractable from input. We tested colour components separately, every frame from view finding input was taken (values from consequent frames were not proved to be correlated). The pixels values were concatenated into the binary string, based on simple extraction technique. For testing with the NIST battery, we liked to add as little post-processing as possible to prevent situations when low-entropy source is processed using high-quality mixing function and then pass all NIST tests even when should not. Note, that only one bit per frame is extracted from the single pixel. We are aware of various extraction techniques that try to overcome this problem (stimulating discussion can be found in [17]) and can extract more, possibly close to theoretical estimation given by the Shannon formula. However, as we liked to perform the NIST test on data with as little post-processing as possible, we decided to use only this simple technique.

For a given frame, all pixels are processed row by row:

1. For every even pixel do: if actual pixel value modulo 2 = 0, then set output bit b to 0 otherwise to 1.
2. For every odd pixel do: if actual pixel value modulo 2 = 0, then set output bit b to 1 otherwise to 0.
3. Bit b is appended at the end of bit stream.

Set of the possible values from the range $\langle 0, 255\rangle$ is divided into two groups. A first group contains only even values and second group contains only odd values. According to histogram, both groups should have almost same sum of probabilities. Values from first group will result in bit value 0, second group in value 1. Unfortunately, separation into two groups with the same probability is not possible. As groups will not have the exactly same probability, fixed bit assignment rule may result in significant difference between number of ones and zeroes. We choose to invert this bit value for the half of pixels thus balancing more probable value 0 for one pixel by higher probability of 1 for the next one.

The independent binary streams were constructed for each colour component using described extraction technique with all pixels within the frame (2100 frames were used, gathered in burst of 7 consequent frames then approximately 5 second delay needed to save data to removable memory). All streams were tested with the NIST battery (100 sequences per 1 Mb) on significance level 0.01. One sequence failed for red colour, all passed for green colour and two sequences failed for blue colour (non-overlapping template tests). We are aware of possible impact of relatively short length of sequences, but we were restricted by the camera memory, acquisition speed (especially time need to store the captured frame on removable card) and 1 bit per pixel by simple extraction technique. Additional tests are definitely needed here, but initial results show considerable amount of entropy (the green colour was always able to pass tests).

4 Preliminary design of practical generator

In this section we focus on practical aspects of random data generation in mobile devices. At first, suitable sources of randomness and expected entropy have to be available. Second, entropy extraction process must be computationally feasible on such devices. Preferably, all data from the source is processed by entropy extraction function (for practical application, cryptographic hash function like SHA-1 or SHA-256 may be used). A careful reduction of the whole data stream can be performed when real-time processing of the whole data stream is not possible (e.g., due to limited computation power or memory restriction). However, as an attacker can possibly exploit fixed selection rule so this rule should be also dynamic (e.g., using only pseudo-randomly selected blocks of pixels rather than only green color component).

To estimate acceptable amount of raw data from sources, we performed practical performance experiments on mobile phones Nokia N73 (Symbian OS v9.1), Nokia 6021 (JavaME), and Sony-Ericsson k750i (JavaME), using SHA-1 as an entropy extraction/mixing function. See table 2 for results. Both execution environments (native C and Java) of Nokia N73 phone provide enough computation power for real-time extraction from view finding (camera input 180x240 pixels ~ 380kB/sec) and microphone (16 bit mono PCM 8000Hz ~ 16kB/sec). Unfortunately, this is not the case for Nokia 6021 (not equipped with camera) where even full audio input cannot be hashed in real-time.

	Nokia N73 (Symbian v9.1)	Nokia N73 (JavaME)	Sony-Ericsson k750i (JavaME)	Nokia 6230 (JavaME)	Nokia 6021 (JavaME)
SHA-1	2200kB	426kB	84kB	67kB	4.65kB

Table 2: Performance comparison of cryptographic hash functions in mobile devices.

We also performed some battery endurance tests for camera source with Nokia N73. Approximately, the camera will drain the battery out after 6 hours of uninterrupted view finding. As we are able to generate enough entropy for a single 128 bit key in order of hundreds of milliseconds (safe estimation entropy amount as described in 3.3.2), generation process should not significantly affect a normal battery lifetime.

5 Conclusions and future work

We have seen that the mobile device provides us with a several randomness sources that can be utilized for generating of true random numbers. Especially the microphone and camera input that are available on (almost) each mobile phone have a promising potential and may allow generating data with a sufficiently large amount of entropy that can be used for cryptographic purposes. These sources have a great potential and allows generating data with a sufficiently large amount of entropy that can be used for cryptographic purposes. A subject of our next research will be the better examination of these sources with more accurate entropy estimation, particularly the mask of independent elements from given source (pixels for camera, sampled values from microphone) to estimate an extractable random bits per second.

We would like also to implement an application for mobile devices that continuously pools available entropy from these sources and maintains an entropy pool. As quality of entropy sources may differ between different devices (e.g., less or more quality optical sensor or inability to bypass some post-processing), part of the work will be focused on creation of semi-automatic calibration software, that allows to perform basic entropy estimation tests inside the device (and without need of time-consuming data transfer to external PC). This calibration software can be used to verify correct work of source during its life-time as well (self-test phase).

References

[1] Krhovják J., Švenda P., Matyáš V.: *Generování náhodných dat v mobilních zařízeních.* Data Security Management (DSM), Vol. 2006, No. 6, ISSN 1211-8737.

[2] Marsaglia G.: *DIEHARD Statistical Tests.* 1995. URL: http://stat.fsu.edu/pub/diehard/ *(last check: 13/3/2007).*

[3] Federal Information Processing Standards Special Publication 800-22. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.* NIST 2001. Available at: http://csrc.nist.gov/publications/nistpubs/800-22/sp-800-22-051501.pdf *(last check: 13/3/2007).*

[4] Ellison C.: *IEEE P1363 Appendix E – Cryptographic Random Numbers.* 1995. Available at: http://theworld.com/~cme/P1363/ranno.html *(last check: 13/3/2007).*

[5] Eastlake D., Crocker S., and Schiller J. *RFC 4086 – Randomness Requirements for Security.* 2005. Available at: http://www.ietf.org/rfc/rfc4086.txt *(last check: 13/3/2007).*

[6] Jun B., Kocher P.: *The Intel Random Number Generator.* Cryptography Research, 1999. Available at: http://www.cryptography.com/resources/whitepapers/IntelRNG.pdf *(last check: 13/3/2007).*

[7] Cryptography Research, Inc. *Evaluation of VIA C3 Nehemiah Random Number Generator.* 2003. Available at: http://www.cryptography.com/resources/whitepapers/VIA_rng.pdf *(last check: 13/3/2007).*

[8] Id Quantique. *Random Numbers Generation using Quantum Physics.* Whitepaper, 2004. Available at: http://www.idquantique.com/products/files/quantis-whitepaper.pdf *(last check: 13/3/2007).*

[9] *LavaRND.* 2000. Available at: http://www.lavarnd.org/ *(last check: 13/3/2007).*

[10] Davis D., Ihaka R., Fenstermacher P.: *Cryptographic Randomness from Air Turbulence in Disk Drives.* Advances in Cryptology – CRYPTO'94, Vol. 839 LNCS, 1994, s. 114–120.

[11] Goldberg I., Wagner D.: *Randomness and the Netscape Browser.* Dr. Dobb's Journal, Special issue on Encoding: Encryption, Compression, and Error Correction, 1996.

[12] Kelsey J., Schneier B., Ferguson N.: *Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator.* Sixth Annual Workshop on Selected Areas in Cryptography. Springer, 1999. Available at: http://www.windowsecurity.com/uplarticle/4/yarrow-full.pdf *(last check: 13/3/2007).*

[13] *Trusted computing group.* Available at: https://www.trustedcomputinggroup.org/groups/mobile/ *(last check: 13/3/2007).*

[14] Shannon, C. E.: A Mathematical Theory of Communication. In The Bell System Technical Journal, 1948.

[15] Rényi A.: *On measures of information and entropy.* Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability, 1960, 547–561.

[16] Andor Technology. *Digital Camera Fundamentals.* Available at: http://www.andor.com/pdfs/Digital%20Camera%20Fundamentals.pdf *(last check: 13/3/2007).*

[17] Barak B., Shaltiel R., and Tromer E.: True Random Number Generators Secure in a Changing Environment. Available at: http://theory.csail.mit.edu/~tromer/papers/rng.pdf *(last check: 13/3/2007).*