Towards True Random Number Generation in Mobile Environments

Jan Bouda, Jan Krhovjak, Vashek Matyas, and Petr Svenda

Faculty of Informatics, Masaryk University
Botanicka 68a, 602 00 Brno, Czech Republic

Abstract. In our paper, we analyze possibilities to generate true random data in mobile devices such as mobile phones or pocket computers. We show how to extract arguably true random data with a probability distribution $\epsilon = 2^{-64}$ close to the uniform distribution in the trace distance. To postprocess the random data acquired from the camera we use a randomness extractor based on the Carter-Wegman universal$_2$ families of hashing functions. We generate the data at the bit rate approximatively 36 bits per second – we used such a low bit rate only to allow statistical testing at a reasonable level of confidence.

Keywords: min-entropy, random number generator, randomness extractor.

1 Introduction

A good generator of random numbers for cryptographic purposes must guarantee good statistical properties and unpredictability of its output. The whole generation process is not a trivial task and it must utilize nondeterministic sources of randomness together with high-quality digital postprocessing. Without physical randomness the output and all derived cryptographic key material could be (easily) predictable. Digital postprocessing by randomness extractors or pseudorandom number generators is necessary to improve statistical properties of generated data and to offer protection against limited malicious manipulation of the source of randomness.

Many documented flaws in implementations of random or pseudorandom number generators exist – e.g., in an early version of Netscape SSL [GW96], OpenSSL FIPS Object Module [OS07], Java 2 ME [KMC07], or Sun's MIDP Reference Implementation of SSL [SMH05]. They have a critical impact to the security of many common services. For example, a recently published flaw (resulting in predictability of the generator) in the Debian OpenSSL caused that SSH keys, SSL certificates, DNSSEC keys, OpenVPN keys, and DH/DSA session keys generated in the past two years should be considered compromised [DS08].

In our paper, we study options for true random number generation in mobile devices that contain no special purpose hardware designed especially for this task. We examine and describe randomness sources available to the application programmer in current mobile phones and other mobile devices, e.g., the

Nokia N73 with the Symbian OS that we used for our tests. The main criteria for true random number generation in our setting are provable security against active adversaries and sufficient bit rate of the source.

The paper is structured in the following way: In Section 2 we discuss possible sources of randomness we can use and in Section 3 we describe how to process the acquired (biased) random data to obtain (almost) uniformly distributed randomness. In Section 4 we review randomness extractors, choose extractors suitable for our purposes and describe one particular extractor used in our tests. The tests performed to verify quality of randomness that we input to the extractor and obtain from the extractor are described in Section 5. Finally, in Section 6 we discuss possible ways to improve the bit rate and computational efficiency of the extractor.

2 Sources of Randomness

We performed several practical experiments on two Nokia N73 smartphones (with the Symbian OS) and two similar PDA phones E-Ten X500 and E-Ten M700 (with the Windows Mobile OS) [KSM07]. The goal of our experiments was to identify (and assess the quality of) sources of randomness in these mobile devices and to estimate the amount of randomness (min-entropy) in these sources. We used two identical Nokia N73 devices since we wanted to verify the correctness of our results or to detect unexpected behavior of the smartphone – in a case when one device suffers, e.g., by some manufacturing defect.

Due to the API restrictions (especially in the Symbian OS), we were forced to drop sources of randomness like the battery level, signal strength or GPS position as measurements over these sources do not provide output (at the API level) with a sufficient precision (e.g., battery and signal values are available in the form of an integer between 0 and 10) or frequency (e.g., external GPS provides only one measurement per second). On the contrary, microphone and digital camera perform a high-rate sampling of physical sources, yielding high volumes of data. Since we can never guarantee the quality of a physical source, our analysis is concentrated on the microphone and the camera noise that arises, e.g., in the CCD/CMOS chip or the A/D converter, and is always present in the output data.

Unfortunately, digitalized microphone input is slightly correlated, and captured camera frames contain, due to low-level postprocessing and optical sensor technology (such as row-dependent readouts), several systematic defects. These defects result in a decrease of the entropy of probability distribution supplied by these sources. Moreover, all external sources of randomness can be influenced (gamma rays, temperature, etc.) and protection against such attacks requires advanced defence techniques. Changes of temperature have an impact to the noise level that slightly influences the probability distribution, but we have not discovered any regularities that could make the output more easily predictable. On the contrary, direct illumination of CCD/CMOS chip can result in generation of highly biased values – overexposed pixels yield always the value 255. Detailed discussion of these issues can be found in [KSM07].

Digital camera provides higher data rate than the microphone and this implies also a higher entropy (and min-entropy) yield in the same time period. We used "view finding" (resolution 180×240, a standard function of Nokia N73 smartphone) rather then a full-size image to get the raw image data without almost any postprocessing. Such raw data were captured at temperatures of 8, 20 and 45 degrees Celsius to observe the influence of environment temperature on the camera noise arising from chip. Unless explicitly mentioned all results in this paper refer to data obtained from the Nokia N73 digital camera at 8 degrees Celsius – i.e., data with the smallest amount of noise (and min-entropy).

In order to use our sources of randomness for cryptographical purposes, we need to obtain a sequence of random bits distributed according to the uniform distribution. The problem to solve is that our device not only outputs (in general) a biased distribution, but this distribution in not known in advance since it might be influenced by an adversary. To postprocess our data and to get an output with some provable guarantees of the probability distribution we have to use some randomness extractor.

3 Processing Randomness

The straightforward way to construct the postprocessing software would be to estimate the probability distribution of camera output first ($180 \cdot 240 \cdot 8 \cdot 3 = 1\,036\,800 \approx 2^{20}$ sample points), or at least the min-entropy of the distribution, and then design a suitable randomness extractor. The obvious obstacle to this method is the size of sample space that prevents reasonable statistical testing (the number of samples would be practically unreachable).

We will limit the amount of data used from each frame to 4 bits only to allow statistical testing. In order to use more information from each frame at least to increase expected min-entropy we would not take 4 raw bits directly obtained from the camera, but rather as a function of a number of bits.

To extract random bits from a CCD/CMOS chip we adopted the following general approach (see Figure 1):

1. Acquire a picture (frame) from a CCD/CMOS chip of the camera (when the lens is closed).
2. Apply a function f (see below) that distills few (in our case four) random bits from the input frame.
3. Repeat step 1. and 2. until a sufficiently long output sequence is accumulated. Then input this sequence to the randomness extractor to obtain the final random sequence.

The role of function f is to preprocess the random data in the way that regardless of the (limited) actions of the adversary the probability distribution of sequences of outputs of function f has with high probability min-entropy sufficient for successful randomness extraction (for chosen extractor). The main problem when designing function f is the limited knowledge of specification of the CCD/CMOS chip and the postprocessing done by the camera. Moreover, these details change

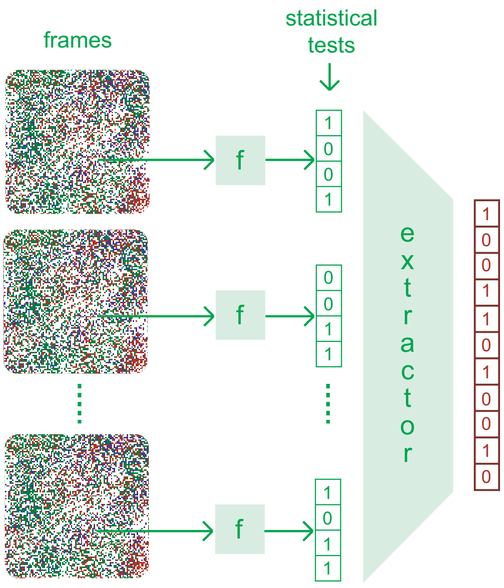

Fig. 1. Scheme of the Data Postprocessing

with both model and producer of the mobile device. We use the function f to separate the extractor from these technical assumptions.

Following the aforementioned motivation we designed the function f to be an XOR (parity) of least significant bits (LSBs) of selected (colors of) pixels of the CCD/CMOS chip. The choice of LSBs has the obvious motivation that this bit is hard to predict and to flip this bit a very accurate (physical) attack is needed[1].

We make one important assumption in our analysis – that bits obtained from different frames are independent. To assure this, we cyclically change pixels used as inputs for the function f. Also, we limit the sampling frequency to 12 frames per second.

Let us concentrate on the noise arising from the CCD/CMOS chip in the Nokia N73 smartphone. The generator we considered takes 12 pictures per second from the view finding (resolution 180×240 function of the cell phone), while the lens of the camera is shut and therefore the influence of the output by external lights is minimized. Randomness contained in pictures comes mainly from the heat inducing electrical charge in cells of the CCD/CMOS chip. As we mentioned before, we use a function f that extracts 4 bits from each picture. We proceed with sampling sufficient number of 4-bit sequences to estimate the probability distribution our source delivers with reasonable confidence. We run the same tests in various external conditions that may influence the source (cold, heat – 8, 20 and 45 degrees Celsius) and obtain limitations on probability distributions an adversary can achieve. More details on performed tests are presented in Section 5.

[1] As a further improvement we later discuss XORing all bits of particular (colors of) pixel instead of LSBs only.

More precisely, we used 180×240 pixels from each frame. Function f logically divides pixels in one frame to 300 squares of 12×12 pixels and uses exactly one pixel from each square. These pixels are selected deterministically in such a way that the neighbors are always from different row and column. The main motivation is that we get a pixels from different locations of CCD/CMOS chip and we mitigate possible dependencies of pixels from successive frames.

Finally, we extract only four bits from these 15×20 squares/pixels. Firstly, we apply XOR function to all pixels (and colors) in a column. Then we transform the resulting 20 bytes (1 byte per column) into 4 bytes by XORing together several different non-neighbor columns: $(1, 5, 9, 13, 17)$, $(2, 6, 10, 14, 18)$, $(3, 7, 11, 15, 19)$, and $(4, 8, 12, 16, 20)$. At the end we apply XOR also to all bits in each byte, or alternatively extract only the LSB from each byte. The output is in both cases exactly four bits per frame – to improve inter-frame pixel independence we took the resulting pixels in order $1, 3, 2, 4$, respectively. For statistical properties of the output see Section 5.

Note that a successful attack scenario defeating such preprocessing technique would require an attacker capable of repeatedly flipping or predicting all bits used for construction of all resulting four bits per frame. Aside from aforementioned need of very accurate physical attack to CCD/CMOS chip, flipping one particular bit in image captured from "view finding" will be very tricky, since algorithms used in mobile devices for downgrading the CCD/CMOS image resolution to image with a "view finding" resolution are kept secret.

4 Randomness Extractor

In this section we review the theory of randomness extractors, choose extractors suitable for our purposes and describe one particular extractor used in our implementation and corresponding tests.

4.1 Definition of Randomness Extractor

The randomness extractor is a function $e : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ that takes n bit random (biased) input, d bit auxiliary uniformly distributed random input and outputs m bit sequence that should be distributed according to the (almost) uniform probability distribution (see below). The processed data obtained from the camera are used as the first parameter, while the second parameter requires true random uniformly distributed data. This means that such data must be pregenerated, and it can be, e.g., stored in the memory and refreshed via network after a fixed number of extractor iterations (meanwhile the output of the extractor is used). This is an obvious drawback, but such an extractor allows us to process a wide range of input probability distributions, namely any probability distribution with sufficiently high (pre-chosen) amount of randomness.

Definition 4.1. *The* **min-entropy** *of a probability distribution* $(p_1, \ldots, p_{2^n})$ *is*

$$H_\infty(p_1, \ldots, p_{2^n}) = \min_{i=1,\ldots,2^n} -\log p_i = -\log \max_{i=1,\ldots,2^n} p_i. \tag{1}$$

Definition 4.2. *Let X and Y be random variables defined on the same sample space $\mathcal{S}$ with probability distributions p_X and p_Y, respectively. We say that X and Y are* **ϵ-close in the trace (or L_1)** *distance iff*

$$\frac{1}{2}\sum_{a\in\mathcal{S}} |p_X(a) - p_Y(a)| = \max_{A\subseteq\mathcal{S}} |P(X \in A) - P(Y \in A)| \leq \epsilon. \tag{2}$$

Definition 4.3. *The function*

$$e : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m \tag{3}$$

is a (non-deterministic) (k, ϵ) **randomness extractor** *if for every input distribution X on $\{0,1\}^n$ with $H_\infty(X) \geq k$ and uniform probability distribution on $\{0,1\}^d$ the probability distribution of the output is ϵ-close in the trace distance to the uniform probability distribution on m bit strings.*

The motivation why to use the min-entropy in the randomness extractor definition is straightforward – it is impossible to extract $d + k$ random bits using an extractor with d bit auxiliary input provided the min-entropy of the input is $k - 1$ or less. On the other hand, this bound can be approached asymptotically [Sha02].

The other possibility is to use a deterministic extractor $e : \{0,1\}^n \to \{0,1\}^m$ requiring no uniformly distributed input. Unfortunately, possibilities of such a source are strongly restricted – it partitions all inputs into fixed subsets, each of which is mapped to one fixed output bit sequence. Therefore, this extractor gives unbiased output only as long as the probability of each such subset is unchanged. As an extreme example, given a fixed randomness extractor with n-bit input and single bit output, there is always a probability distribution on n-bit strings with min-entropy $n - 1$, such that the output of the extractor on this distribution is deterministic, i.e. the distribution $(0, 1)$ [Sha02].

It may be impossible to explicitly construct one suitable deterministic extractor, however, non-deterministic extractor with random, but public and fixed auxiliary input can do very well [BST03] in some situations. For more details on randomness extractors see the survey paper [Sha02].

Finally, in this paper we use extractors that guarantee uniform distribution of concatenated extractor output and auxiliary input. Hence, the useful almost-uniform output of such an extractor has $d + m$ bits.

Definition 4.4. *The function*

$$e : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m \tag{4}$$

is a (k, ϵ) **strong randomness extractor** *if for every input distribution X on $\{0,1\}^n$ with $H_\infty(X) \geq k$ and uniform probability distribution Y on $\{0,1\}^d$ the joint probability distribution $\big(Y, e(X, Y)\big)$ is ϵ-close in the trace distance to the uniform probability distribution on $m + d$ bit strings.*

4.2 Extractors Based on Carter-Wegman Universal Hashing

To fight a general adversary we decided to use a (non-deterministic) randomness extractor, i.e., extractor requiring true randomness as an auxiliary input. This allows us to concentrate only on the analysis of the min-entropy of our randomness source. Due to a high min-entropy of our source (see Section 5) we can choose freely from a wide variety of extractors.

First parameter of extractor we should consider is the length d of the auxiliary random input. In our case this should be considered only in the context of the randomness efficiency, i.e., we should compare the length of the output sequence with the amount of randomness in the input, i.e., the length of the output should be approaching $d + k$. Possibly limiting factor of the initial sequence can be memory of the mobile device, but essentially all devices with a camera already have memory of size (at least) few megabytes (few hundreds for contemporary mobile phones) and therefore we are not limited by a reasonably large auxiliary input that has to be stored (we consider approx. 1000 bits for our purposes).

Since the absolute value of the true random input is not limiting and we have a high min-entropy, we can concentrate on the computational efficiency of our extractor. Computing power of mobile devices is still limited and intensive calculations increase battery exhaustion and thus the computational efficiency is of critical importance. The amount of true randomness is limited, and, therefore, we would like to reuse it or use the output of our extractor instead. A straightforward choice from this point of view are extractors based on Carter-Wegman universal classes of hash functions.

Definition 4.5 ([CW79]). *A class $H = \{h | h : \{0,1\}^n \to \{0,1\}^m\}$ of hash functions is* **universal**$_2$ *iff for every $x, y \in \{0,1\}^n$ there are exactly $|H|/2^m$ functions h such that $h(x) = h(y)$.*

Theorem 4.6 ([SZ99, IZ89, IL89]). *Let X be a random variable defined on the sample space $\mathcal{S} = \{0,1\}^n$ with probability distribution p having min-entropy $H_\infty(p) \geq k$, $H = \{h | h : \{0,1\}^n \to \{0,1\}^{k-2e}\}$ be a universal$_2$ class of hash functions. Let $x \in_R \{0,1\}^n$ be randomly chosen from $\{0,1\}^n$ according to p and h be randomly and uniformly chosen from H. Then the distribution of $(h, h(x))$ is 2^{-e} close to the uniform distribution in the trace distance, i.e. application of a function randomly chosen from H is a $k, 2^{-e}$ strong randomness extractor.*

As follows from the previous theorem, with each application of the extractor we loose $2e$ bits in contrast to the theoretic achievable rate. Note that this is fixed and independent of n, k and $|H|$ and, therefore, by using longer input sequence (with higher min-entropy) we may approach the theoretic bound arbitrarily. However, this may (depending on the particular hash function used) increase the computational complexity. Also, if we use a part of our output as the d auxiliary bits or we reuse original randomness, the quality of the output distribution decreases, but by a limited and well-defined amount.

Theorem 4.7 ([SZ99]). *Let $X_1, X_2, \ldots, X_l$ be independent random variables defined on the sample space $\mathcal{S} = \{0,1\}^n$ with a common probability distribution*

p having min-entropy $H_\infty(p) \geq k$, $H = \{h | h : \{0,1\}^n \rightarrow \{0,1\}^{k-2e}\}$ be a universal$_2$ class of hash functions. Let $x_i \in_R \{0,1\}^n$ be randomly chosen from $\{0,1\}^n$ according to X_i and h be randomly and uniformly chosen from H. Then the distribution of $(h, h(x_1), \dots, h(x_l))$ is $l\,2^{-e}$ close to the uniform distribution in the trace distance, i.e. l repeated applications of a fixed function randomly chosen from H is a $k, l\,2^{-e}$ strong randomness extractor.

4.3 Choosing Suitable Class of Hash Functions

A straightforward observation shows that any *XOR universal* class of functions [Sho96] is universal$_2$ as well, and, therefore, we may use any of the classes of hash functions designed for message authentication and heavily optimized towards computational efficiency. One should be careful when comparing computational efficiency, since in the case of message authentication only one (secretly) chosen hash function is being computed and therefore message authentication enables efficient precomputation we can rely upon only when the hash function is fixed and reused.

In our tests we used the universal$_2$ class of functions based on shift registers proposed in [IZ89]:

Let $r = r_1 r_2 \dots r_n$, $x = x_1 x_2 \dots x_n$, $x, r \in \{0,1\}^n$. Let $r \cdot x$ denote the scalar product of r and x, i.e., $r \cdot x = 1$ if the parity of $XOR(r, x)$ is odd and is equal to 0 otherwise. Let $r^{(l)}$ denotes the left bit rotation of r by l positions, i.e., $r^{(l)} = r_{l+1} r_{l+2} \dots r_n r_1 r_2 \dots r_l$. We define the m-convolution of r on x as the bit string $r(x)_m = r \cdot x, \dots, r^{(1)} \cdot x, \dots, r^{(m-1)} \cdot x$.

Let p be a prime such that 2 is a primitive root modulo p, let $n = p - 1$. Let $C_{n,m} = \{h_r | r \in \{0,1\}^p\}$, where $h_r : \{0,1\}^n \rightarrow \{0,1\}^m$ and $h_r(y) = r(1 \circ y)_m$ with $\circ$ denoting the concatenation of bit strings. The class of functions $C_{n,m}$ is universal$_2$ [IZ89].

Implementation of this class is efficient and straightforward – all necessary calculations are scalar product (i.e., bitwise XOR and mod 2 addition) and bit string rotation.

It is important to set suitable parameters for our application. As a final output of the extractor we would like to obtain a random sequence that is at least $\epsilon = 2^{-64}$ close to the uniform distribution in the trace distance, however, we would like to reuse the initial randomness (from the Nokia N73 digital camera) a number of times. For purposes of our testing we use less than 2^{20} repetitions (what suffices for more than 100 days of extracting), however, even doubling the exponent to 2^{40} implies only a loss of extra 40 bits per extraction.

Observing Theorem 4.7 we set parameters $e = 84$ and $p = 839$ (for efficiency reasons sufficiently large suitable prime number) what gives $m = k - 2e = \lfloor log_2 14 \cdot 838/4 - 168 \rfloor = 629$, since we expect that the min-entropy of our source is at least $\log_2 14$ and we have 838/4 four-bit samples (see Section 5). Thus, from each 838 bits of input we extract 629 bits of output (compare to the theoretic optimal value 797 bits) giving the bit rate approximatively 36 bits of key per second.

5 Analysis of Acquired Random Data

To verify the input of the extractor and its min-entropy we ran a number of statistical tests on outputs of the function f. To obtain acceptable quality of the tests on an achievable number of samples (we used 1.5 mil. sample frames) we output only 4 bits from each frame (16 different values). We always tested the output for data captured at 8, 20 and 45 degrees Celsius.

First battery of tests concentrates mainly on the statistical *stability* of the output distribution – i.e., that the output data have all time (even in different external conditions) almost the same distribution (not necessarily uniform). We applied Pearson's (χ^2) goodness of fit test on several sequences and subsequences. As the expected distribution we take the relative frequency of 4-bit values from the whole sequence generated from all 1.5 mil. sample frames – in our case given by values: 0.0630, 0.0608, 0.0674, 0.0629, 0.0607, 0.0697, 0.0639, 0.0607, 0.0642, 0.0606, 0.0573, 0.0580, 0.0642, 0.0627, 0.0612, 0.0625. As the observed distribution we use the relative frequency of the 4-bit values from various subsequences – in our case, e.g., given by values: 0.0642, 0.0642, 0.0706, 0.0645, 0.0545, 0.0691, 0.0621, 0.0594, 0.0624, 0.0624, 0.0561, 0.0542, 0.0624, 0.0585, 0.0606, 0.0645. The resulting p-value (in this particular case 0.22) at the significance level $\alpha = 0.01$ clearly confirms our hypotheses that the sequences are at 99% from the same probability distribution.

Consulting the relative frequency we obtained (at the normal temperature of 20 degrees Celsius) we see that the min-entropy of tested distribution is approx. 3.984, what is almost the maximum of 4. As a further evidence of high entropy of our source we calculated the minimum of the Pearson's χ^2 test of our estimated distribution over all distributions with min-entropy at most $\log_2 14$. Recall that we designed the extractor the way it works for any distribution with min-entropy at least $\log_2 14$. We calculated

$$\min_{H_\infty(q) \leq \log 14} \chi^2(p, q)$$

with p being our estimated distribution. The outcome is $\chi^2 = 338.9947$, p-value is 2.0167×10^{-63}, what strongly rejects such a hypothesis.

As a part of our testing we considered other functions f as well. While it is easy to see that XORing an extra random bit to a given random bit can only decrease possible bias (or preserve it when the bit is deterministic), it would be nice to use as few input bits as possible. The reasons for such an optimization is not only the computational efficiency, but allocation of resources as well. As already mentioned, in our testing we used only a 4 bit output, but it might be useful to use just a few input bits to design extractors with higher bit rate, where independence of input data is argued in the terms of hardware. Our analysis showed that green color behaves in general more deterministically than red and blue color. Also, XORing a few pixels together already helps significantly to obtain higher entropy probability distribution. When comparing function f as described in Section 3, it has only a negligible (improving) impact if we XOR all bits from the pixel color channel instead of using only LSBs. Therefore, a suitable function f should XOR together bits from several pixel distributed over the frame and use data from all color channels. On the other hand, it is sufficient to use LSBs only.

We also verified the design and our implementation of the randomness extractor (described above) by several statistical tests of the output. In this case we applied the well-known NIST statistical test suite to 40 MB output. All sequences passed 15 out of 16 statistical tests (frequency test, runs test, etc.). We tested division to 100 subsequences at the confidence level $\alpha = 0.01$. Passing the test means that a large fraction of subsequences passed particular sub-tests and also that the resulting p-values have uniform distribution. The Chi-square (χ^2) goodness of fit test is used to test the uniformity as well, but the confidence level is more strict in this case ($\alpha = 0.001$). The only one exception was "the serial test" that failed due to the small fraction of passing subsequences – their fraction 0.9500 was lower than (for 100 subsequences) the expected 0.960150.

6 Conclusion and Future Work

In our paper we describe a true random number generator delivering a bit string distributed according to a probability distribution very close to the uniform probability distribution. The initial randomness is acquired partly as an initial true random bit string and partly as a processed output of a built-in camera. In the previous analysis we used only a negligible portion of randomness contained in each frame acquired from the camera. This was only to allow for a reasonable statistical testing of data, while in practice much larger amount of data can be obtained from each frame. This would increase the bit rate significantly.

Another limitation is the sampling rate of 12 frames per second that can be increased provided independence between data produced from consecutive frames is guaranteed, e.g., through a hardware and postprocessing analysis.

Another area to explore is to select an optimal randomness extractor from computational complexity point of view, while preserving other required properties – near-optimal length of the extracted bit string and good behavior of the extractor when true random data are replaced by almost random data (or the same true random data are used many times). Future research will also concentrate on implementation of various randomness extractors within the mobile environment and comparison of their performance.

It is of an independent interest to investigate physical abilities of the adversary to influence the output of the CCD/CMOS chip at both precise and wide-area scale, as well as hardware and camera postprocessing based dependence between individual pixels within a single frame and between consecutive frames.

Acknowledgments. We acknowledge the support of the research project of Czech Science Foundation No. 102/06/0711 and the research project MSM0021622419. Jan Bouda also acknowledges the support of the project GAČR201/06/P338 of the Czech Science Foundation. Authors thank Luděk Smolík for stimulating discussions and Filip Jurnečka for his help with the extractor implementation.

References

[BST03] Barak, B., Shaltiel, R., Tromer, E.: True random number generators secure in a changing environment. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 166–180. Springer, Heidelberg (2003)

[CW79] Carter, J.L., Wegman, M.N.: Universal hash functions. Journal of Computer and System Sciences 18, 143–144 (1979)

[GW96] Goldberg, I., Wagner, D.: Randomness and the Netscape Browser. Dr. Dobb's Journal, Special issue on Encoding: Encryption, Compression, and Error Correction (1996)

[IL89] Impagliazzo, R., Levin, L.A., Luby, M.: Pseudorandom generation from one-way functions. In: Proceeding of the 21st ACM Symposium on Theory of Computing (1989)

[IZ89] Impagliazzo, R., Zuckerman, D.: How to recycle random bits. In: Proceedings of the 30th IEEE Symposium on Foundations of Computer Science, pp. 248–253 (1989)

[KMC07] Klingsheim, A.N., Moen, V., Hole, K.J.: Challenges in Securing Networked J2ME Applications. Computer 40, 24–30 (2007)

[KSM07] Krhovjak, J., Svenda, P., Matyas, V.: The Sources of Randomness in Mobile Devices. In: Proceeding of the 12th Nordic Workshop on Secure IT Systems, Reykjavik University, pp. 73–84 (2007)

[OS07] Lowe, G.: OpenSSL Security Advisory – OpenSSL FIPS Object Module Vulnerabilities (2007), `http://www.openssl.org/news/secadv_20071129.txt`

[DS08] Bello, L.: Debian Security Advisory – OpenSSL predictable random number generator (2008), `http://www.debian.org/security/2008/dsa-1571`

[Sha02] Shaltiel, R.: Recent Developments in Explicit Constructions of Extractors. Bulletin of the EATCS 77, 67–95 (2002)

[Sho96] Shoup, V.: On fast and provably secure message authentication based on universal hashing. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 313–328. Springer, Heidelberg (1996), `http://www.shoup.net/papers/macs.pdf`

[SMH05] Simonsen, K.I.F., Moen, V., Hole, K.J.: Attack on Sun's MIDP Reference Implementation of SSL. In: Proceeding 10th Nordic Workshop on Secure IT Systems, Tartu University, pp. 96–103 (2005)

[SZ99] Srinivasan, A., Zuckerman, D.: Computing with very weak random sources. SIAM Journal on Computing 28(4), 1433–1459 (1999)