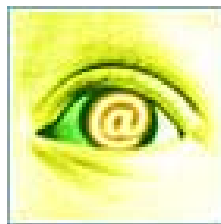




## Kaspersky anti-virus engine technology

Comprehensive protection from  
today's threats and tomorrow's



Confidential

---

# Contents

INTRODUCTION .....	3
1. 'GOOD' & 'BAD' ANTI-VIRUS ENGINES.....	4
2. KEY FEATURES OF THE KASPERSKY ANTI-VIRUS ENGINE.....	5
1. Signature analysis .....	5
2. Checksumming .....	6
3. Techniques for detecting polymorphic viruses .....	6
<i>Reduced masks</i> .....	7
<i>Known plaintext cryptanalysis</i> .....	7
<i>Statistical analysis</i> .....	7
<i>Emulation</i> .....	7
<i>Polymorphic viruses: summary</i> .....	8
4. Processing complex objects .....	8
5. Heuristic analysis .....	10
6. Updating virus signatures.....	11
3. NEW TECHNOLOGIES IN KASPERSKY ANTI-VIRUS ENGINE 5.012	
1. iChecker™ and iStreams™ .....	12
2. iCure™ .....	12
3. iArc™ .....	13
4. Multi-threaded operation.....	13
4. CONCLUSION .....	14
5. CONTACT DETAILS.....	15

---

## Introduction

The nature of the threat to PC users has changed significantly over the years. Today's threats are more complex than ever before. Much of today's 'malware' [short for **malicious software**], which includes Trojans, backdoors and spammers' proxy servers as well as viruses and worms, is purpose-built to hijack users' machines; and a single Trojan can easily be found on many thousands of infected PCs. Malicious code may be embedded in e-mail, injected into fake software packs, or placed on 'grey-zone' web pages for download by a Trojan installed on an infected machine. There has also been a growth in spyware, adware, dialers and other 'unwanted', but non-viral, programs. The scale of the problem, in terms of numbers alone, has also continued to increase.<sup>1</sup>

At the same time, the anti-virus market is saturated with products. This raises the question of how to choose the best product. Which ones will guarantee 100% detection of viruses? Which ones offer the most efficient combination of technologies capable of comprehensively protecting your computer and network from all types of malware?

The core of any anti-virus product is the so-called anti-virus 'engine', a software module purpose-built to find and remove malicious code. The engine is developed independently of any specific product implementation. So it 'plugs-in' equally well into personal products, like personal scanners or real-time monitors, or solutions for servers, mail scanners, file servers, firewalls and proxy-servers.

The reliability of malicious code detection, and, hence, the security level provided by the engine, ultimately depend on the engine's structure, its detection methods and the heuristic technologies integrated into the engine.

This document outlines the key elements of the Kaspersky® anti-virus engine. This includes scanning features that are common to many anti-virus products, but also unique technologies that make the Kaspersky Lab anti-virus engine so effective in finding and removing malicious code.

---

<sup>1</sup> The Kaspersky Lab anti-virus databases now contain more than 100,000 records.

---

# 1. 'Good' & 'bad' anti-virus engines

Anti-virus vendors tend to conceal the details of their engines from the public. And with good reason, of course, since they have no wish to publish information that the 'bad guys' might be able to use to circumvent particular techniques used in the engine. However, there are indirect ways you can use to determine whether a particular engine is 'good' or 'bad'. That is, is it more or less effective at finding and removing malicious code?

**How to determine whether a particular engine is 'good' or 'bad'?**

Below is a list of the main criteria for selecting an anti-virus engine.

**Quality of virus detection** indicates the effectiveness with which the anti-virus program detects viruses. The best way to assess an anti-virus vendor's detection capability is to check out its track record in a range of independent tests.<sup>2</sup>

**Level of heuristic detection** indicates a program's ability to find new, unknown threats. Proactive detection has become increasingly important given the speed at which today's threats spread. Unfortunately, it's very difficult to assess a product's capability in this area without access to a virus collection. However, a number of independent test organizations have begun to include this in their test methodologies.<sup>2</sup> In addition, the number of 'false alarms', which is easier to measure, is also indicative of the quality of an engine's heuristic analyzer.

**Number of false alarms** is an important measure of an engine's quality. If an anti-virus program reports an infection in a clean file, this is called a false alarm, or false-positive. Not only do frequent false alarms undermine a user's confidence in a program's heuristic analyzer. They can also prevent a user from recognizing a new virus [the program 'cries wolf' so often that the user stops trusting it].

**Detection of malicious code inside the many compressed, archived and packed formats** is critical because virus writers frequently compress their code using different compression utilities, to produce several distinct executables. In fact, all these viruses are duplicates of the same virus. And if an anti-virus engine supports all [or almost all] popular compression utilities, it will easily detect all copies of the same virus and determine its name. Other anti-virus programs, by contrast, will require a virus definition update [and may also require additional time for analysis by one of their virus researchers].

**Update size and frequency** are also indicative of the quality of the anti-virus engine [as well as the quality of the vendor's research team]. While the engine itself is designed to be updated infrequently, a frequently updated anti-virus database guarantees that a user will be constantly protected from the latest threats. The size of each database update [as well as the number of detected threats] is a good indicator of the quality of the anti-virus database and, to some degree, the engine itself.

**Engine-only updating**, without the need to update the entire anti-virus program, indicates the efficiency of the engine technology. In some cases, in order to detect a virus, a user must update not only the anti-virus database but also the engine. If it's not easy to update the engine, the user's computer or network may become infected with a new virus. In addition, this feature allows a manufacturer to quickly troubleshoot and improve the engine.

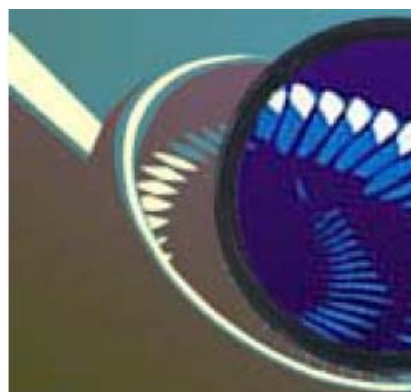
---

<sup>2</sup> See *Evaluating anti-virus tests* for further information.

---

## 2. Key features of the Kaspersky anti-virus engine

The appearance of the first computer viruses forced programmers to respond quickly to the new threat. This led to the creation of the first anti-virus programs. Since then, anti-virus software has changed dramatically, to respond to the threat posed by each successive generation of malware. Today's anti-virus programs differ as much from those old versions as an up-to-date PC differs from, say, a calculator.



The Kaspersky® anti-virus engine is integrated into all Kaspersky® anti-virus products and delivers a unique combination of technologies necessary for the successful detection of malicious code. The Kaspersky® anti-virus engine is designed on the basis of a powerful and flexible logical subsystem that employs all the latest methods needed to find and remove malware. The key features of the Kaspersky® anti-virus engine are outlined below:

- basic detection methods for viruses and other malware
- special techniques for detecting polymorphic viruses
- processing complex objects
- virus signatures and their update mechanism

### 1. Signature analysis

A signature is a unique sequence of bytes that is specific to a piece of malicious code. Signature analysis, or a modification of it, was [and remains] one of the first methods used in anti-virus engines to detect viruses and other malware. Evident advantages of this method are its high speed [especially with the use of special algorithms] and the possibility of detecting several threats using one signature. On the other hand, a serious disadvantage of this method is that for reliable detection of malicious code, the signature must be large, at least 22-40 bytes [anti-virus producers usually use longer signatures, of up to 64 bytes, to improve the detection level]. So the size of the anti-virus database also increases. Another challenge to this method is that much contemporary malware is written in high level languages such as C++, Delphi or Visual Basic. These programs contain fragments of code that do not change [the so-called Run Time library]. If a wrong signature is used, this leads to false alarms, where a clean file is reported to be infected. The false alarm problem can be solved by using extremely large signatures, or by restricting detection to certain data areas like relocation tables or text strings, which is undesirable.

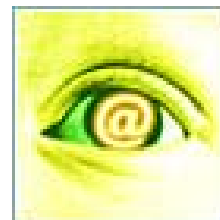
---

## 2. Checksumming

Checksumming is a method based on calculating CRC [Cyclic Redundancy Check] checksums and is a modification of signature analysis. The method was developed to overcome the main disadvantages of the signature method, large databases and frequent false alarms. Checksumming accounts for not only the search string [or, to be more precise, a checksum for the string] but the location of the string in the body of a malicious program. The location is used to calculate the checksums for the entire file. Thus, instead of a 10-12 byte search string [and this is a minimum size], the checksum takes four bytes and the location data also take four bytes. However, checksumming is more time consuming than signature analysis.

## 3. Techniques for detecting polymorphic viruses

Self-encryption and polymorphism are used in most types of virus to maximize the difficulty of their detection. Polymorphic viruses are extremely difficult to detect because they do not have signatures, that is, there's no constant fragment of virus-specific code. In most cases, two samples of the same polymorphic virus do not have a single coinciding fragment.



There are many kinds of polymorphic viruses, from boot and DOS file viruses to Windows viruses, macro and script viruses. Polymorphic 'envelopes' are also used to hide Trojan programs.

Viruses are called polymorphic if their body is self-changing during replication to avoid the presence of any constant search strings. Polymorphic viruses can not be detected [or can be detected only with great difficulty] using so-called virus signatures or masks, sequences of unchanging virus-specific code. Polymorphism is achieved by encrypting the main code of the virus with non-constant keys containing random sets of decryption commands, or by changing the executable virus code. There are also other rather exotic examples of polymorphism. For example the DOS virus Bomber is not encrypted, but the sequence of instructions, passing control to the body of the virus, is completely polymorphic.

It is problematic to use signatures [sometimes called 'search strings'], as outlined above, to detect polymorphic viruses. Since the code changes with each infection, it becomes impossible to select the correct signature. Even a very large signature can not be used to identify an encrypted virus uniquely without giving false alarms. It's not difficult to see why. The polymorphic virus encrypts its body, converting the virus code into a variable. And variable code can not be selected for a signature.

So for detection of polymorphic viruses, additional techniques must be used.

---

## ***Reduced masks***

If the encryption algorithm used by the virus is not sufficiently advanced, it's possible to use elements within the encrypted body of the virus to take the encryption key out of the equation and obtain static code. The signature, or mask, can then be taken from the resulting static code.

## ***Known plaintext cryptanalysis***

Known plaintext cryptanalysis, another method for dealing with polymorphic viruses, works like this. Using the known original virus code and the known encrypted code [or suspicious code that looks like an encrypted virus body], the engine reconstructs the keys and the algorithm of the decrypting program. The engine then decodes the encrypted virus body by applying this algorithm to the encoded fragment.

The use of a system of equations to decode an encrypted virus body is similar to the classical cryptographic problem of decoding an encoded text without keys. However, there are two key differences. First, most of the data required for the solution is known. Second, the solution must be solved using available RAM and with limited time.

In general, this method is less time consuming and uses a smaller amount of memory than emulation of virus instructions [see below]. However, this solution implies constructing a system of equations and it becomes rather complicated. The main problem is the mathematical analysis of the equation or the system of equations constructed.

## ***Statistical analysis***

Statistical analysis is another method used to detect polymorphic viruses. The engine analyzes the frequency of the processor commands used and uses this information to make a decision on whether the file is infected or not. This method is quite effective for those polymorphic viruses that use a limited set of opcodes in their decryptors, compared to clean files that use other opcodes with a different frequency. For example, many complex polymorphic viruses rarely use the DOS interrupt 21h [CDh 21h opcode] in their decryptors, while most legitimate programs use it frequently. The main disadvantage of this method is that there is a family of complex polymorphic viruses that use the opcodes of virtually all processors and the set of these commands changes dramatically from infection to infection, thus making it impossible to detect such viruses using a frequency table.

## ***Emulation***

The increase in the number of polymorphic viruses in the early 1990s, and in particular the first appearance of polymorphic viruses in the field, led to the development of a method of emulating the program code [also known as 'sandboxing']. Using this method, program execution [of both infected and clean

---

programs] is emulated in a virtual environment, called a 'sandbox' or virtual machine. After this emulation process, where the program is a polymorphic virus, the buffer contains a decoded virus body ready to be detected using standard methods [signature analysis or CRC checksumming]. Current systems emulate not only processor opcodes, but also operating system calls: in other words, the greatest part of any computer.

It is quite difficult to write a decent emulator. In addition, when an emulator is used, the actions of every command must be constantly controlled to prevent the program from occasionally executing the destructive virus instructions that are present in most known viruses. It's also important to stress that program emulates the execution of virus instructions, rather than tracing them, because tracing virus activities increases the risk of executing destructive instructions or the codes responsible for activating the virus itself.

### ***Polymorphic viruses: summary***

In practice, deciding on the use of the above methods for detecting polymorphic viruses [use of reduced masks, cryptanalysis, statistical analysis and emulation], comes down to a selection of an optimal balance that provides the maximum speed and minimum memory usage. The code of most self-encrypting viruses can easily be decoded using emulation. If emulation is not an optimal solution, the virus code can be decoded using a subprogram that applies cryptanalysis to this code. To detect viruses that are non-decodable, or that can not be emulated, the engine uses a method of reduced masks.

In complicated cases, the Kaspersky® engine uses a combination of the above methods. A fragment of the decryptor code is emulated to distinguish commands that are responsible for the decrypting algorithm. Then, based on the information obtained, the engine constructs and solves a system of equations to decrypt and detect the virus code.

The above-described methods are combined in the case of multiple encoding, where a virus encrypts its body several times using various encryption algorithms.

A combination of methods for decoding information or, in other words, 'pure' emulation of the decoder code, is often used in the engine because every new virus must be analyzed and integrated into the anti-virus database in the shortest time, which is sometimes not possible with mathematical analysis. As a result, more laborious detection methods are used, leaving behind the mathematical methods that can be applied to analyze the decryption algorithms.

## **4. Processing complex objects**

In recent years, as we have already observed, anti-virus engines have changed dramatically. For the first anti-virus programs, it was enough to check system memory, executable files, and boot sectors. After several years, due to the increased popularity of special compressing utilities, anti-virus developers encountered the problem of how to extract a compressed file before scanning it.

---

Then, a new problem appeared when viruses started infecting archives [and users often sent each other infected archives]. Anti-virus programs had to learn how to process archived files. There were other related problems too. The first macro virus to infect Microsoft Word documents appeared in 1995. Word documents are stored in a closed, complex format and some anti-virus producers are still unable to process such files well. Due to the worldwide use of e-mail, contemporary anti-virus engines must also be able to scan e-mail databases and e-mail messages.

It's critical for anti-virus programs to be able to scan within such complex objects because there could be a hidden threat lurking within any one of them.

The Kaspersky® anti-virus engine currently supports 2,000 run-time packers and 300 archiving utilities.

**The Kaspersky® anti-virus engine supports over 2,000 run-time packers and 300 archiving utilities**

The engine supports a wide range of utilities for compressing executable files, as well as encryption systems. This includes the following:

Diet, AVPACK, COMPACK, Epack, ExeLock, ExePack, Expert, HackStop, Jam, LzExe, LzCom, PaquetBuilder, PGMPAK, PKLite, PackWin, Pksmart, Protect, ProtEXE, RelPack, Rerp, Rjcrush, Rucc, Scramb, SCRNCNCH, Shrink, Six-2-Four, Syspack, Trap, UCXEXE, Univac, UPD, UPX [several versions], WWPACK, ASPack [several versions], ASProtect [several versions], Astrum, BitArts, BJFnt, Cexe, Cheaters, Dialect, DXPack, Gleam, CodeSafe, ELFCrypt, JDPack, JDProtect, INFTool, Krypton, Neolite, ExeLock, NFO, NoodleCrypt, OptLink, PCPEC, PEBundle, PECompact [several versions], PCShrink, PECrypt, PE-Diminisher, PELock, PEncrypt, PE-Pack [several versions], PEProtect, PE-Shield, Petite, Pex, PKLite32, SuperCede, TeLock, VBox, WWPack32, XLoc and Yoda.

The engine also supports a wide range of archivers and installers. This reduces the time required for analysis of new viruses, thus accelerating the response to new threats and providing the highest level of detection of known viruses. Archivers and installers supported include the following:

CAB, ARJ, ZIP, GZIP, Tar, AIN, HA, LHA, RAR, ACE, BZIP2, WiseSFX [several versions], CreateInstall, Inno Installer, StarDust Installer, MS Expand, GKWare Setup, SetupFactory, SetupSpecialist, NSIS, Astrum, PCInstall, and Effect Office.

Support for all these archivers and their modifications is particularly important when scanning e-mail traffic, because a great number of viruses are sent via e-mail as archives.

Objects are extracted regardless of the archive nesting depth. For example, if an infected file is compressed with the UPX utility and then archived in a ZIP file, which in turn is archived in a CAB file, the Kaspersky engine will still be able to extract the original file and detect the virus.

The engine uses a smart algorithm that avoids extracting so-called 'archive bombs', seemingly small archives [highly compressed] that expand into huge files or several identical files. Such archives usually take quite a long time to scan, but the Kaspersky anti-virus engine can instantly recognize such bombs among normal archives.

---

## 5. Heuristic analysis

In the early 1990s, as the number of viruses grew to a level exceeding several hundreds, anti-virus experts investigated the possibility of detecting viruses that were currently unknown and for which there was no signature. As a result, the so-called heuristic analyzers were created.



A heuristic analyzer is a set of subprograms used to analyze the code of executable files, macros and scripts, in memory, files or boot sectors, to detect various types of malware. The two main principles used in heuristic analyzers, static and dynamic analyses, are outlined below.

Static heuristic analysis involves a search for general short signatures specific to most viruses [so-called 'suspicious commands']. For example, many viruses search for files using the \*.EXE mask, open the file found and write their code into this file. The task of the heuristic analyzer is to find signatures that are indicative of these activities. Then the program analyzes the signatures and, if a number of 'suspicious commands' are found, it reaches the decision that the file is infected. This method is easy to implement and delivers high-speed scanner performance. However, the level of detection of new malicious programs is rather low.

Dynamic heuristic analysis was developed simultaneously with the introduction of code emulators into anti-virus programs [see above]. The dynamic method emulates program performance and logs all 'suspicious actions' of the program. This log is then used to make a decision about whether the program is infected or not. Unlike the static method, the dynamic heuristic analysis method requires more resources but provides a higher level of detection.

The heuristic analyzer integrated into the Kaspersky® anti-virus uses both cryptanalysis and statistical analysis. It was designed from the outset as an extensible module, unlike many other first-generation heuristic analyzers that were designed to detect malicious code only in executable files. At present, the Kaspersky® heuristic analyzer successfully detects malicious code in executable files, disk sectors and computer memory. It also effectively reveals new script viruses and malware for Microsoft Office [and other programs that use Visual Basic for Applications], as well as code written in high level languages like Microsoft Visual Basic.

Due to its flexible architecture and combination of various methods, the Kaspersky® heuristic analyzer is able to detect new malware with high efficiency. At the same time, the number of false alarms has been minimized. At present, the Kaspersky Anti-Virus engine is clearly superior to other well-known products. The number of false alarms has been reduced virtually to zero, confirmed by numerous independent tests and feedback from Kaspersky Lab customers.

---

## 6. Updating virus signatures

The anti-virus database is an inseparable part of an anti-virus engine. As already observed, a well-designed engine is not often updated, whereas the database must be constantly updated because it stores signatures, checksums and special modules for detecting malware as it appears. It's well-known that new threats appear every day.<sup>4</sup> So it's important to update the anti-virus database as frequently as possible. In the early days of PC viruses, quarterly updates were enough for most customers. Later, monthly updates became standard. Even five years ago, it was normal to update the anti-virus database only every week. Now it's better to update more frequently. Home users should update their databases every day. Enterprises, with thousands of PCs to protect, have a higher risk of infection because of the number of possible victims, so protection is more critical. It's advisable for enterprises to update several times a day [around once every three to six hours at least]. ISPs [and this also applies to corporate e-mail servers and other 'perimeter' anti-virus defenses] should check for new updates even more frequently.<sup>5</sup>

The elements included in the anti-virus database are also significant, since there may be not only virus signatures in the database but also other program procedures. Such procedures offer a way of updating the engine through a database update.

The Kaspersky® anti-virus database is updated hourly. Owing to the smart architecture of the Kaspersky anti-virus engine, these updates are incremental, adding detection just for new threats rather than replacing the entire database each time the user does an update. In addition, the size of the updates doesn't normally exceed 40KB, although sometimes Kaspersky releases updates containing specific fixes [for a new unpacker, for example], so it's possible that an update may be as big as 1MB. Approximately 70% of the anti-virus engine functionality is integrated into the database. In this way, for example, support for a new archiver or compression utility can be added to the anti-virus database at any time. Thus, regular daily updates provide not only enhanced detection for malware, but also updated engine functionality. This feature ensures a very quick response to any given situation and maximum protection against viruses.

**Regular daily updates provide not only enhanced detection for malware, but also updated engine functionality**

---

<sup>4</sup> Currently [November 2004] more than 100 new records are added to the Kaspersky® anti-virus database every day.

<sup>5</sup> One ISP that partners with Kaspersky Lab checks for new updates every 10 minutes.

---

### 3. New technologies in Kaspersky anti-virus engine 5.0

The latest technologies implemented in the Kaspersky anti-virus engine accelerate the scanning of objects at a high detection level and enhance the detection and disinfection of archived malicious programs. These include the following.

- ✓ **Accelerated scanning**
- ✓ **Enhanced detection & disinfection of archived malware**

#### 1. iChecker™ and iStreams™

*iChecker™* and *iStreams™* are new scanning technologies created by Kaspersky® anti-virus experts. They are designed to provide an optimal balance between the level of anti-virus protection for workstations, and especially servers, and the system resources of protected computers. These technologies reduce the system startup time by between 30% and 40% and also reduce application launch time when real-time protection is active. In this case, it is guaranteed that all files stored on hard drives have been scanned and identified as clean.

*iChecker™* is based on the fact that most routine scans are redundant. More often than not, no virus is found when a scan is done. Using *iChecker™*, the Kaspersky® engine does not need to scan files that have not been changed since the last scan. Here's how it works. The engine creates a database of checksums for all scanned [and therefore virus-free] files. Before any file is scanned, the engine calculates a checksum for the file again and compares the result with the original stored in the database. If the checksums are identical, the file has not changed since it was last scanned and does not need to be scanned again. The process of checksumming the file and comparing it with the checksum in the database is less time consuming than scanning the file. *iChecker™* works for the following types of files: EXE, COM, LNK, TTF, ELF, INF, SYS, CHM, ZIP, RAR, DOC, XLS and PPT.

*iStreams™* works in the same way. However, it is specially tailored for NTFS file systems [the native file system for Windows NT and Windows XP]. Rather than storing checksums for each clean file in a database, the engine stores information on previous scans in an NTFS alternate data stream. This makes it even more efficient than *iChecker™*, but it works only for NTFS systems.

#### 2. iCure™

*iCure™* is a technology for disinfecting archived files. Using this technology, infected objects inside archives are successfully disinfecting or deleted, depending on user-defined settings, without using other archiving utilities. Now, the Kaspersky Anti-Virus engine removes viruses from the following types of archives: ARJ, CAB, RAR, and ZIP.

---

### 3. iArc™

*iArc™* is another technology for processing archives. This technology is designed to improve the processing of multi-volume archives implemented in previous versions of the Kaspersky® anti-virus engine. The *iArc™* technology allows scanning of multi-volume archives. Now the Kaspersky® engine can detect even a virus added to a multi-volume archive, which is in turn added to another multi-volume archive.

### 4. Multi-threaded operation

The Kaspersky® anti-virus engine is a multi-threaded module that can process several objects at the same time [files, sectors, scripts, etc.].

---

## 4. Conclusion

In the 'good old days', the ability of a virus to spread was limited by the user. Viruses could only travel as fast as users' activity allowed them to. Boot sector viruses, for example, which accounted for around 75% of all infections until the mid-1990s, relied on the exchange of floppy disk in order to spread. This meant that they moved only slowly in global terms and infections tended to be localized. The macro viruses that dominated the field until 1999 still relied on unsuspecting users to exchange infected documents and spreadsheets. So they too were limited in the speed at which they could spread. In these circumstances, it was generally possible to deliver a cure before a virus had the chance to spread very far.

The spread of today's threats, by contrast, leaves very little time in which to respond. With malware that can reach epidemic proportions in hours or [in worst cases] minutes, anti-virus vendors must provide a fix in the same timeframe. An anti-virus engine that is 'built for speed' is essential. On the one hand, it must be capable of scanning large volumes of data in a timely fashion. On the other hand, it must be architected like a high performance car, so that it can be easily modified to deal with the complexity of successive generations of malware. The Kaspersky® anti-virus engine, now in its fifth generation, is purpose-built to protect the enterprise and home user alike from the threats of today and tomorrow.

## 5. Contact Details

We are open to discuss all cooperation offers and will take into account the specifics of any business model. Our comprehensive approach to business allows us to be flexible and consider all possible interests.

Please feel free to contact us at any time regarding any of our products or services.

### Kaspersky Lab

**Postal Address:**

10 Geroyev Panfilovtsev St.,  
125363 - Moscow,  
Russian Federation

**WWW Address:**

Corporate Site: <http://www.kaspersky.com>

OEM and Technology Solutions Division:

<http://oem.kaspersky.com/>

**Phone and Fax Numbers:**

Phone: +7 (095) 797 8700

Fax: +7 (095) 780 33 68

**E-mail:**

OEM and Technology Solutions Department [oem@kaspersky.com](mailto:oem@kaspersky.com)

