

CipherCAD Testbed

Vaclav PLATENKA, Antonin MAZALEK

Dept. of Communication Technologies, Electronic Warfare and Radiolocation, University of Defense,
Kounicova 65, 662 10 Brno, Czech Republic

{vaclav.platenka, antonin.mazalek}@unob.cz

Submitted April 30, 2021 / Accepted May 7, 2021

Abstract. *The CipherCAD testbed is a unique workplace for the development, design, testing, verification and teaching of the communications systems. CipherCAD is at the core of the workplace, which is an application primarily designed for solving cryptographic tasks. The application can also be used for communicating with hardware communications devices. The workplace is used in the Department of Communications Technologies at the University of Defense. The article will present selected examples used in this workplace. The introduction introduces CipherCAD and the possibilities for creating simple models. The first model to be selected shows how to control SDR IZ225 and process the signals received. The next model shows how selected modulated signals are generated in real time, and their transmission throughout the whole chain of communications. The models that follow show how they can be used in the field of communications protocols, VoIP transfer and changing any of the parameters of the transmitted information.*

Keywords

CipherCAD, communication systems, model, digital receiver, VoIP

1. Introduction

The development of communication and information systems is leading to ever more complicated and sophisticated solutions. It is impossible to focus only on parts of selected parameters and properties for transferring information, such as how efficient the use of a radio spectrum is, reaching the maximum achievable transfer rates and robustness of transfer. The issue of data transfer security is an integral part of all information systems today [1], [2]. Modern communication systems can serve to achieve data privacy, data authenticity and data integrity, while also guaranteeing the delivery of data and many other features and services [3].

The well-known principles for transferring information divided into various layers of TCP/IP, or the RM OSI model have become ever more complex. Security is

generally implemented at all levels (security on the application layer, security on the transport layer, security at the link layer and physical layer) and the systems are becoming ever more comprehensive. It is common for experts only to focus on some layers, such as securing data by encryption at the application layer. The issue of securing data against errors in the transfer channel using FEC (Forward Error Correction) codes is a field on its own covered by other specialists. The comprehensiveness of such systems is even greater for military radio communication systems, which also solve problems raised by an EW (Electronic Warfare) opponent (signal intercept, localization, analyzing, tampering, jamming,...) [4], [5]. These factors make it difficult for these systems to be analyzed, tested, verified and taught. There are a number of specific SW and HW professional and freeware tools available. However, most of them only focus on part of the field and do not allow for a comprehensive solving of the problem.

Over the last few years, it has been possible to create a relatively unique workplace, the CipherCAD Testbed at the Department of Communication Systems. This workplace facilitates the comprehensive analysis, modelling, testing, verification and demonstration of the principles of action for various communications systems. The software application CipherCAD is at the core of the whole workplace [6]. This application makes it possible to create models of communication systems in real time. The model created can communicate with real HW devices. Moreover, the application has a number of suitable forms for displaying information, as well as providing easy access to each partial element of the model.

The workplace is modular and makes it possible to connect CipherCAD with HW and SW devices communicating via an IP protocol or an input/output computer sound card. In this article, we will focus on working closely with the IZ225 receiver, the communications system Emona TIMS and the Cisco IP phone. We will also use the software AKRS-RT for displaying and analyzing the signals received.

The aim of the article is to present the CipherCAD Testbed and to demonstrate its possibilities in specific examples. The basis of the workplace is the application CipherCAD, which will be introduced in detail in Sec. 2.

The following section will be devoted to showing how to make use of CipherCAD for controlling a software-defined radio and processing received samples of signals. The next aspect to be presented will be the possibilities of generating one's own signals and sending them using a sound card. Section 4 describes several models focused on the issue of voice transfer in IP networks. Created models are supporting communication with HW phones. In the conclusion, other possible uses of the workplace are discussed, along with suggestions for extending the functionality of CipherCAD.

2. Description of the CipherCAD Tool

The software application CipherCAD has been developed for the needs of NCISA (National Cyber and Information Security Agency). The original idea was to develop a CAD-type application which allows modelling, research, testing and verifying cryptographic primitives, algorithms and protocols. After several years of development, the application has been extended to include a number of new functions and possibilities. It currently provides comprehensive access to solving various tasks not only in the field of cryptography. An important feature of CipherCAD that we use in the field of communication systems is the ability to communicate with external HW and SW using an IP interface and sound card. Any data can be read and sent using these interfaces.

Basic diagrams can be composed from the prepared graphics component. The basic graphics components are contained in the tools bar. This includes the sources of data in various formats, the basic algebraic components for addition and multiplication, delay, the collector, the static and dynamic switch, the component for viewing data in various formats and the component for the graphic display of data in various modes, as can be seen in Fig. 1.

Figure 1 shows a simple diagram illustrating counting using the exclusive or (XOR). The input values are entered using a hexadecimal data editor, which shows the green components. The data from the output of components is brought to the addition operator set to XOR mode via connections. The output of the addition is sent to two components: the yellow component - data view; and the blue component - the static switcher. The bottom part of the picture shows the permutations features of setting the static switcher. The output of the static switcher is again sent to the viewer in hexadecimal form. Much more complex diagrams can be formed using this process.

The programmable component shown in Fig. 2 is important. This component allows the users to write their own codes of component behavior using their own programming language (similar to the programming language C#).

Figure 2 shows the same calculation as in Fig. 1, but written in the programming component. Firstly, two variables A and B are defined, to which the same values

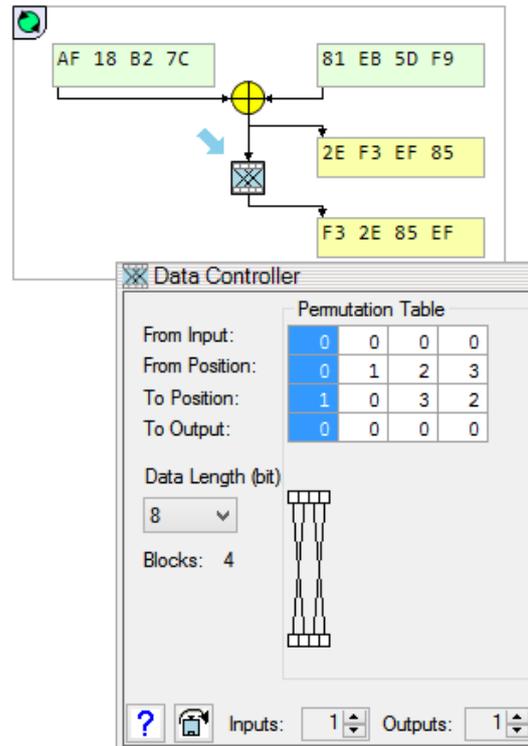


Fig. 1. Example of the basic components of CipherCAD.

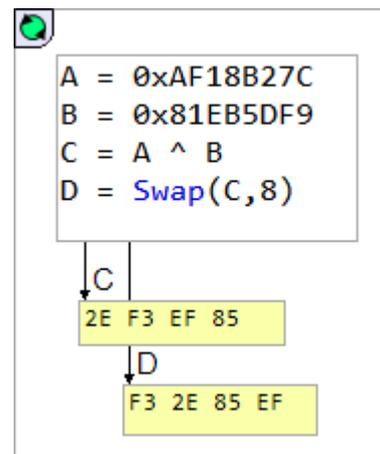


Fig. 2. Calculating in the programming component.

are associated as the input components had in the previous diagram. The result of the XOR operation for the variables A and B is stored as variable C. Next, the Swap function is used to carry out the same permutation, which is performed by a static switch. The programmable component is set up to transfer the values of two variables C and D to the output. The user can change the number of inputs and outputs of the programmable component. Naturally, it is possible to transfer these values from any other components, such as from the hexadecimal data editor as in the last example, to the place of definition of variables A and B inside the programmable component. The programming component can be packed thus creating small component with a defined number of inputs and outputs and adjustable graphics (color, shape, description, etc.) Components created in this way can easily be copied and reused as often as required

when creating complex diagrams. The packed components will be used frequently in the examples described below.

The above components create the basic building blocks which can be used to create more complex diagrams and models. The blocks can exchange data using the displayed lines, or in order to keep the diagrams clear by using transmitting and receiving points or by using global or local variables. The displayed lines also define the order of calculations for the individual blocks. If the order is not defined by lines, the calculations are made from left to right and from top to bottom. A particular space can be marked on the desktop as a rectangle, the so-called iterator. The iterator defines a certain area where the calculations will be made once initialized. Moreover, other features can be set on the iterator, such as the number of repeated calculations and how to initialize it. The iterator can be initialized in a number of ways, whether by manually clicking on the mouse, sending instructions using the line or by calling functions from another already-working iterator.

The working diagram created inside the iterator can be packed into a component as in the case for the programmable component. Such a component will have the necessary number of inputs and outputs, its own form, color and description and can be used for creating more complex algorithms. This can be repeated many times, leading to absorption in the line with the logical structure of the algorithm's design, as can be seen in Fig. 3. Even if absorbed many times over, the user continues to have access to any information appearing in the diagram and can record, display and modify it in real time.

Figure 3 shows how to use the absorption function for the model of the hash function Keccak. The left part of the figure shows the packed component Keccak-512. By clicking the mouse on the packed component, the component is marked top left with a blue arrow as well as displaying an overview of the inner structure of the component. Once in this view of the inner structure, it is possible to click the mouse again to unpack the selected component.

This way it is possible to get to the lowest level of the diagram's structure. The right part of the picture shows the activity of the chi function, which is created from the basic components of CipherCAD. The original iterator in the left part of the picture shows the packed component of the algorithm Keccak-512. This component can be used to create complex models of communication protocols. At the same time, it will be possible to work against HW devices in real time.

Another strong point of CipherCAD is that it is easy, quick and illustrative to display data. Figure 4 shows how to use this feature.

The top component in Fig. 4, the data editor in decimal form, is used to enter the input values. It may be noticed that the component allows text notes in inverted commas to be inserted. With such a component, the user can also set a fixed bit-length for the block where the entered values will be represented in the output of component. The length of the block in the example is set at 64 bits. In this example, five values have been entered into the component separated by spaces. The components' output is transferred to three viewing components. Each of these components has been set for a different format of displaying data, in this example decimal, hexadecimal and binary. The user can select the length of the blocks displayed for the viewed component either automatically according to features of the entered data, or manually as required. There are several options for using the viewed component, whether with the display, data properties, number of values or data checksum. In addition, data can also be stored in a file, again either manually or automatically upon completing a calculation.

The component Graphic Viewer is used to display data in CipherCAD too. This component can be used for a number of tasks from a simple view of the binary order, viewing periods in the displayed data to the dynamic viewing of the frequency spectrum using waterfall. Figures 5 and 6 show the basics of this component.

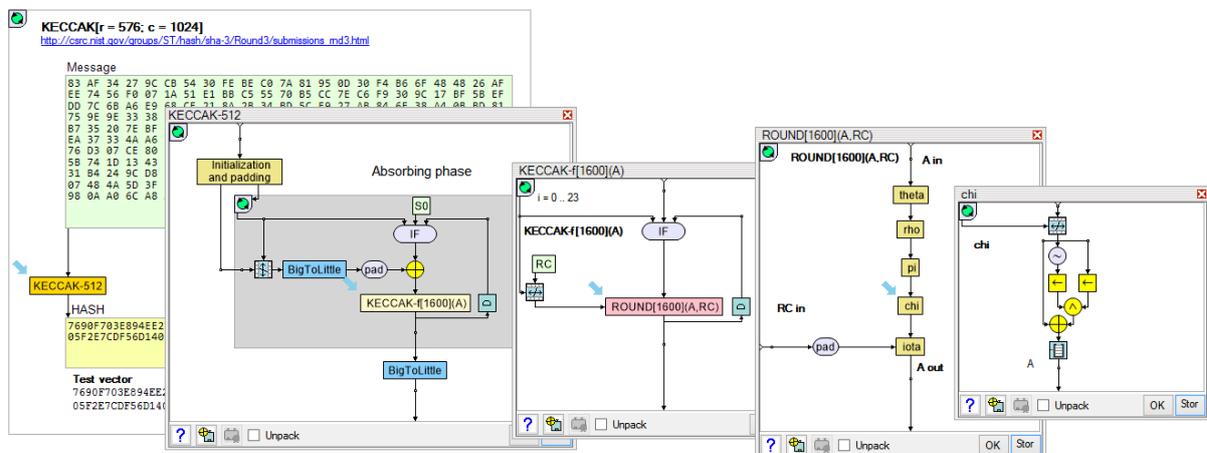


Fig. 3. Absorbing the hash function Keccak in an algorithm.

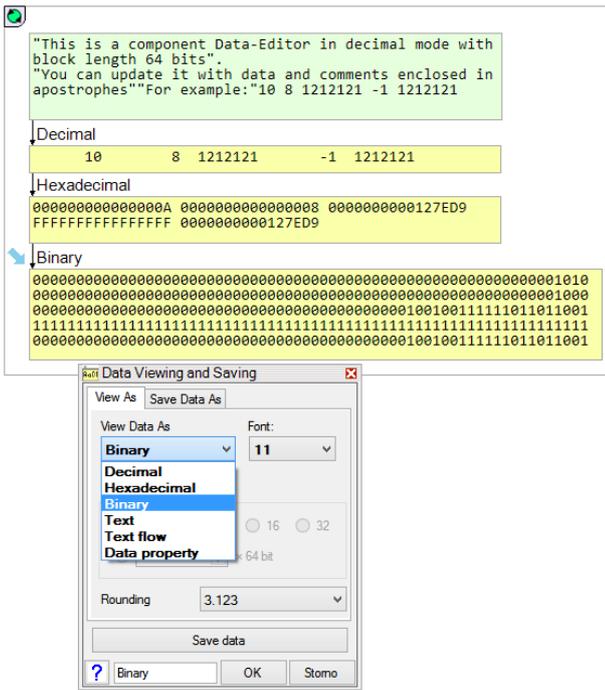


Fig. 4. Data viewing.

Figure 5 shows a lower-frequency digital filter diagram. The diagram shows how static data can be displayed: in this case displaying the input and output filter signal over time. In order to be able to display the chronological order of the sample, the input and output data of the filter need to be saved in the collector. The data from the collector is transferred to the Graphic Viewer. The user can set a wide range of parameters for the component, not just the color, background, grid, line thickness, but especially the pre-processing of input data and selecting how it will be displayed. The basic options include display of waveform signals, maximum, minimum and average values, including bit and byte maps. In addition to static display, CipherCAD enables to display data dynamically.

Figure 6 shows a diagram simulating the activities of an eight-track oscilloscope in X-Y mode. A total of 8 harmonic waveforms have been generated using the programmable component. Eight new values are calculated and immediately displayed in each cycle. However, the graphic component remembers the history of the previous values. The user can set the length of memory. The track's brightness can be set from the checklist. CipherCAD supports up to 6 various tracks on one graph. The graphics component has a number of other properties. One of them, events, will be presented in the following section for controlling receivers.

For the needs of asymmetric cryptography, the application was equipped to deal with large numbers, i.e. numbers that cannot process individually a common arithmetical unit (64-bit computer). The size and accuracy of processing numbers is only limited by the memory capacity of the computer. In addition to the basic arithmetical operations, selected algorithms for number theory are also available. The options can be seen in Fig. 7.

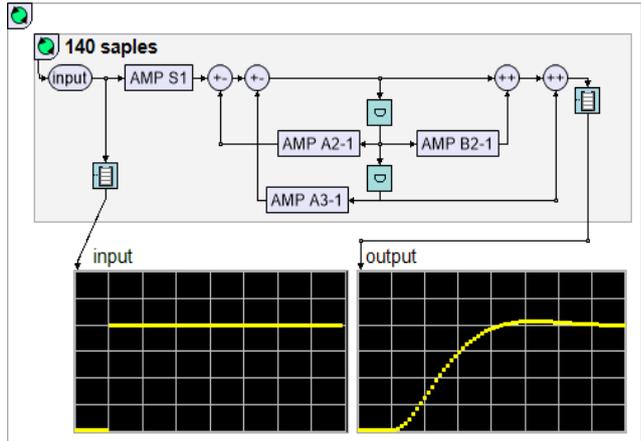


Fig. 5. Graphic view of filter response.

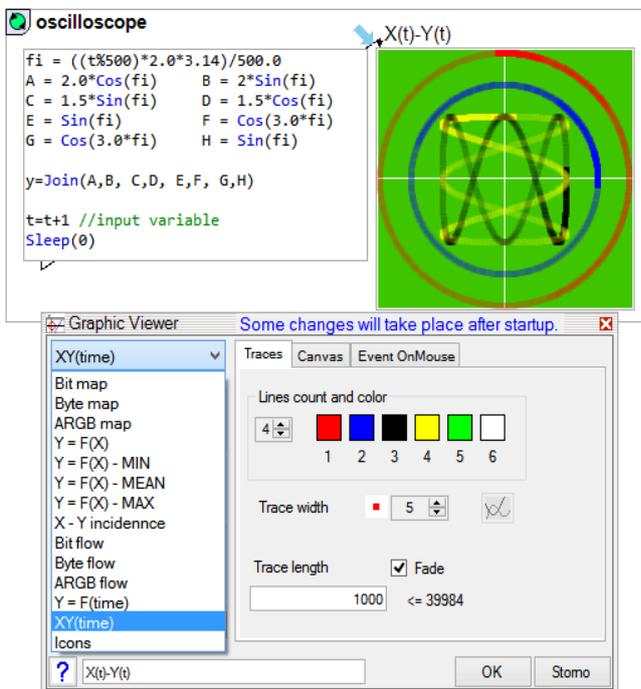


Fig. 6. Dynamic graphic view - oscilloscope.

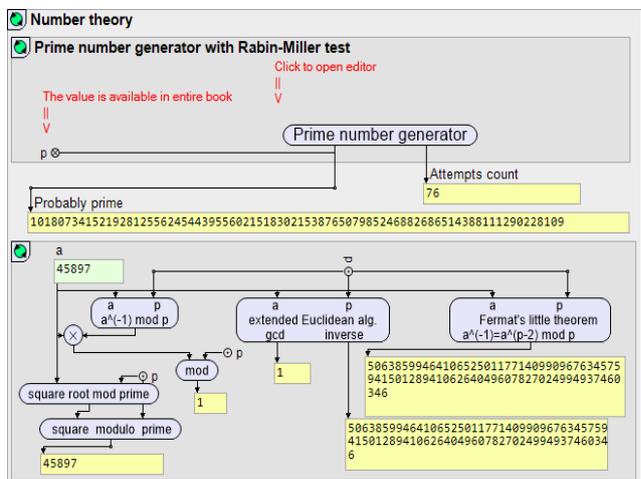


Fig. 7. Generating prime numbers with the Rabin-Miller test.

```

Computing Square Root (Newton)
x=2 // the number we want to root
len = 1000*64 // total bit len of root
rationalPartLen = len-64 // in bits
rp = NewData(1, rationalPartLen)
X = ModifyData(Join(x,rp),len,rationalPartLen)
ui = 1<<(DegPol((1)x)/2)+1 //initial estimation
u = ui
countIter = DegPol((1)len)
loop(countIter)
{ u=(u+X/u)/2}
check = u*u //must be close to x
    
```

1.4142135623730950488016887242096980785696718753769
 480731766797379907324784621070388503875343276415727
 350138462309122970249248360558507372126441214970999
 358314132226659275055927557999505011527820605714701
 095599716059702745345968620147285174186408891986095
 523292304843087143214508397626036279952514079896872
 533965463318088296406206152583523950547457502877599
 617298355752203375318570113543746034084988471603868
 999706990048150305440277903164542478230684929369186
 215805784631115966687130130156185689872372352885092
 648612494977154218334204285686060146824720771435854
 874155657069677653720226485447015858801620758474922
 657226002085584466521458398893944370926591800311388
 246468157082630100594858704003186480342194897278290
 641045072636881313739855256117322040245091227700226
 941127573627280495738108967504018369868368450725799
 364729060762996941380475654823728997180326802474420
 629269124859052181004459842150591120249441341728531

Fig. 8. Calculating the square root of 2 with the required accuracy.

Rational numbers are also used in cryptography with an extensive rational part of the number. Figure 8 shows a way of obtaining an extremely accurate breakdown of the decimal places for the square root of 2. The number of decimal places is 19,246, with less than 1/20 shown in Fig. 8. On a normal computer, it takes less than a second to calculate and display the calculation.

CipherCAD can also operate with polynomials and matrices with any size and accuracy of coefficients.

The ability of CipherCAD to communicate with real HW and SW devices is made possible through a number of network functions for the application. In order to communicate through the IP interface, the user can use simple functions for sending and receiving data using the UDP and TCP protocols. To send the data, it is enough to enter the IP address and the number of the port which the application should use. Due to their overload, the basic functions allow for more sophisticated parameters to be specified for making the connection. The user can thus set the specific IP interface from which the data will be sent/received and the source port number. With the TCP protocol they can change the size of buffer for sending and receiving data and also influence the value of the timer for receiving data in order to transfer a large volume of data over low-speed lines. The application can also send any IP packet on the network layer. In this case, the users must create their own header of the IP protocol, including the calculations of the header checksum field. However, the SendPacket() function allows other protocols to be used than just UDP and TCP, e.g. ICMP. Alternatively, the whole network operation can be picked up on a chosen network interface by switching the network interface to

promiscuous mode and transferring the data from the level of the interface controller. Therefore, the flow of incoming packets is not affected by the settings and filtering of the Windows operating system. Consequently, the network communications are extremely wide and varied, with CipherCAD being able to send and receive any packets at the network level.

Another PC hardware interface used by CipherCAD is the sound card. The user can communicate with any sound interface through the two basic functions PlaySound() and RecordSound(). Overloading both of these functions makes it possible to select a number of parameters such as the number of sound channels, sampling frequency and the number of bits per sample. It can be also determined which of the sound devices available will be used, how many queues will be there and how long queues will be used for communicating with the sound interface. This concept of how the application communicates with the sound interface offers the user a wide range of uses, such as sending/reading the data even in individual bits.

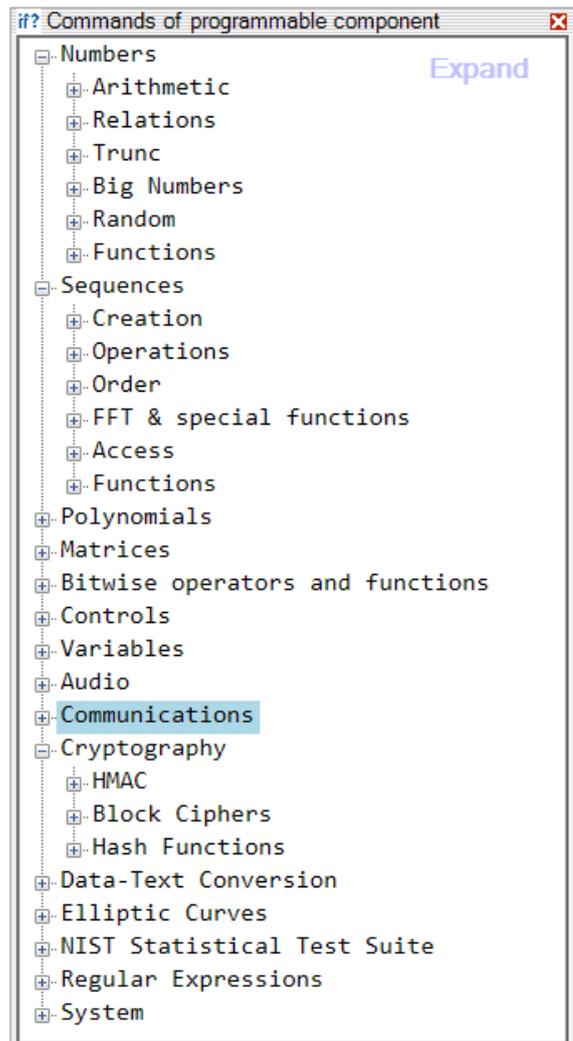


Fig. 9. Overview of instructions for the programmable component.

It is impossible to present all the functions and options of CipherCAD in brief. However, this would include working with global and local variables, matrices, regular expressions, working with data blocks, polynomials, elliptic curves, etc. Figure 9 gives an overview of the instruction areas used in the programming component.

CipherCAD is unique in its combination of these functions and features. This can be seen when designing, testing and verifying cryptographic systems, as well as when modelling comprehensive communications systems. Another area of use of no less importance is that it can be taught to students in the field of communications technology in a comprehensive and practical way. Therefore, it has a wide range of uses. In the next sections, we will show how it can be used with the software receiver IZ225 and in the field of VoIP.

3. Signals in CipherCAD

As has already been mentioned in the introduction, the workplace CipherCAD Testbed contains a wide-band digital receiver, the IZ225 made by Intriple. The workplace is equipped with two of these receivers (see Fig. 10). The IZ225 receiver has software defined radio architecture which provides two modes – receiver and scanner. In the receiver mode it is used for signal analysis – swept spectrum measurements, demodulation, signal capturing and streaming. In the scanner mode it continuously scans frequency spectrum in spectrum monitoring applications. The

scanning speed is extremely high – it can be more than 100 GHz/s, which means that a scan in 1 GHz range is updated every 10 ms [7].

The receiver works in the frequency band from 1 kHz to 3 GHz with a maximum band width of 40 MHz. The digital processing of the signal in the receiver is carried out in FPGA. The receiver is controlled using software via a 10/100/1000 Mbit Ethernet interface. The SCPI control instructions in ASCII format are sent to the receiver via TCP/IP. A summary of all usable SCPI instructions is supplied with the receiver in the Programmer Manual [8]. Data from the receiver is sent via the same interface, but via the UDP protocol. At the same time, more possible data can be sent in the various ports. This means especially real-time IQ data (up to 2.75 MS/s), off-line IQ data (up to 40 MS/s), FFT and demodulated audio signal in digital format.

A number of models have been created in CipherCAD to control the IZ225 receiver and to process the data



Fig. 10. Digital receiver IZ225, version for installation into advanced system – front panel [7].

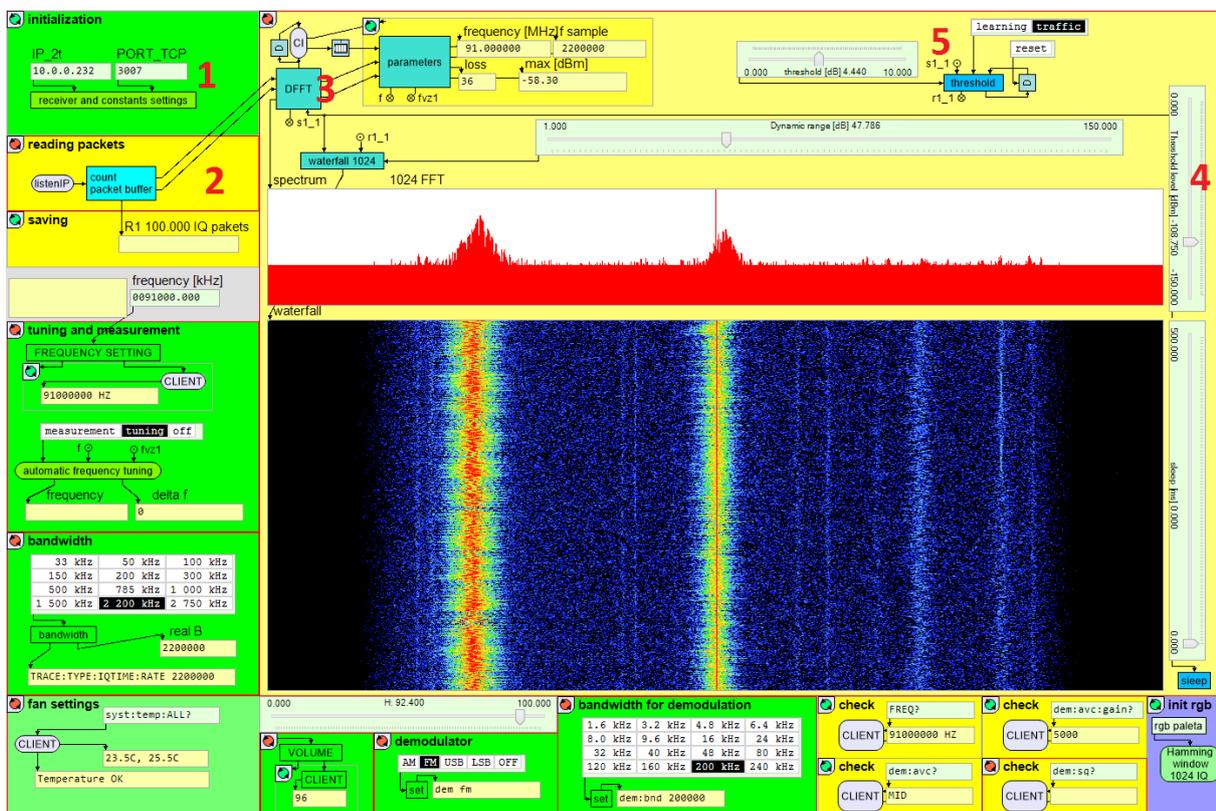


Fig. 11. The basic control environment for the IZ225 receiver in CipherCAD.

received from the receiver. These models are distinguished by a number of suitable elements, which have been taken on from previous models, especially control blocks and the unique blocks designed to process and display the specific types of data received. Figure 11 shows the most universal diagram of receiver control. The blocks designed for control are in green. The blocks designed for processing, storing and displaying the data received are in yellow. Some of these blocks will be introduced in more detail in the following part.

The first of the blocks to be described is initialization. It is marked as 1 in Fig. 11. This block is used to set up the initial settings of the receiver and to send the sequence for running the selected blocks of the diagram. The initialization block is represented in detail in Fig. 12. The user can set the IP address of the selected receiver in the two editors which are visible in the basic menu, as well as the port for sending the SCPI instructions.

After unpacking the receiver's settings block, the user can adjust the sequence of initialization SCPI instructions in the text editor. These are represented in Fig. 12 by the green text fields. Each instruction is written on a separate line, which is separated by enter. In CipherCAD, the line break is represented in the text editor by 0x0D0A in hexadecimal form. However, the IZ225 receiver requires a line break in the abbreviated 0x0A format. Therefore, the ASCII text is converted with SCPI instructions using Replace in the Client programming component. In the specific example in Fig. 12, six instructions are being sent at the same time.

The first instruction SYSTEM:COMMAND RESP sets the short answer format for sending instructions. The short answer format is more suitable for simple parsing. The instruction SYSTEM:ETH:FARG 1 switches off the block fragmentation of the data sent to several packets. As has been described in Sec. 2, the data in CipherCAD is also processed directly in the network interface, making it quicker to process the incoming UDP data packets in the same format. The next instruction TRACE:TYPE:IQTIME:PORT 3010 specifies the port where real-time IQ data is sent. The next instruction is TRACE:TYPE:IQTIME:RATE 2 750 000, which is used for setting the sampling frequency of real-time data in IQ format sent from the receiver. The instruction TRACE:TYPE:IQTIME: ON starts sending IQ data from the receiver. The last instruction used to initialize the receiver is TRACE:TYPE:SPECTRUM OFF, which switches off the calculation of the FFT frequency spectrum in the receiver and sending from it. The displayed frequency spectrum in Fig. 8 is calculated from the incoming IQ data in CipherCAD, which is quicker and better for other uses. The principle will be described in more detail in other part of the article.

Another block in the diagram is the data reading and storage block. This is represented by 2 in Fig. 11. The instruction ListenIP(device_Index) accepts all IP communication in the selected network interface. The accepted packets are then filtered to see whether this is UDP data

from a selected IP address and port. The data packets are then stored in the collector, i.e. the buffer with the pointer assigned to the last position in the memory. In our case, the size of the memory has been chosen as 100,000 data packets. This size is not due to a requirement for the real-time processing of data, but for storage and detailed off-line analysis. The size of the packets is 11,808 bits, of which 608 bits are headers including information about the receiver's settings and 11,200 bits are IQ data. The samples are in 16-bit format, meaning that there are 350 samples in one packet. It can be calculated from the above that, when sampling is set to a maximum of 2.75 MS/s, the last 12.72 seconds of continual IQ samples are recorded.

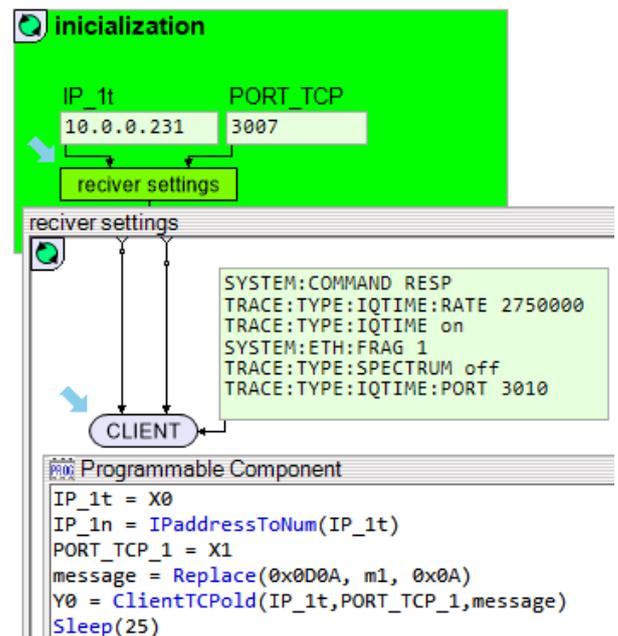


Fig. 12. Sending the SCPI instructions from CipherCAD.

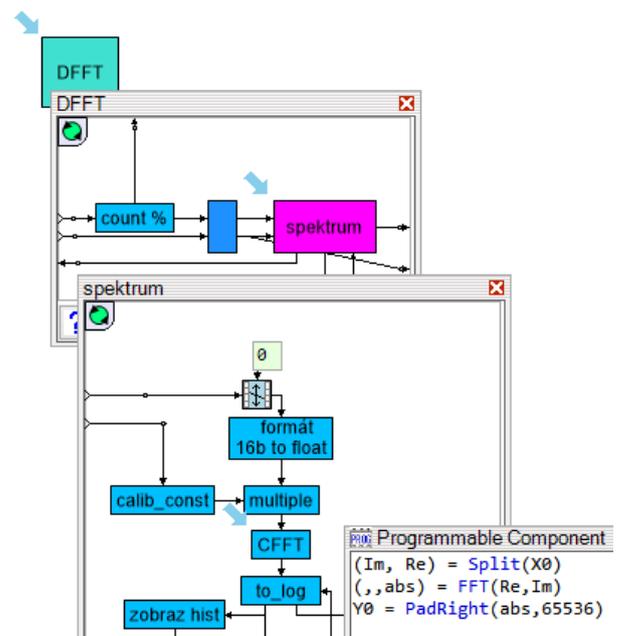


Fig. 13. Block for calculating the frequency spectrum.

The next block calculates the frequency spectrum using DFFT and the resulting processing and display of data. It is marked 3 in Fig. 11. The first block to process the received IQ data is the block DFFT. This block and its internal structure are shown in Fig. 13.

The received IQ data enters the DFFT block from the buffer, along with the pointer for the latest IQ data and the required threshold level for displaying the frequency spectrum. The first two parameters come from the data reading and storage block, the third parameter comes from the control bar, which is to the right of the displayed frequency spectrum. 1,024 IQ samples are always selected from the input data, i.e. the above has just under 3 packets. The required calibration constant for setting the spectrum level correctly is obtained from the header of one of these packets. This constant changes depending on the receiver settings, for example the selected size of the pre-amplifier. The received IQ data is reformatted from the 16-bit signed integer to 64 float format. Next, the data is multiplied with the already-mentioned calibration constant and the selected Hamming window. The frequency spectrum is then calculated using FFT() function, where only absolute values are of interest. After that, the calculated spectrum is transferred to units in dBm and the threshold level and display are set in the graphics component, marked as 4 in Fig. 11. In addition, other parameters of the recently-received signal are obtained and displayed while calculating the spectrum. These include the carrier frequency and sampling frequency. Moreover, information is displayed showing the maximum level of signal in the spectrum and the so-called loss. The word loss refers to the number of spectrums which cannot be calculated and displayed. Either this is due to the speed of sampling, which makes it impossible to calculate and especially display the data. For example, at the fastest sampling at 2.75 MS/s, every 15th block is calculated and displayed of the 1,024 point complex FFT. Alternatively, it can be caused by the deliberate slowing in displaying the frequency spectrum in waterfall. This can be useful for searching for slow changes in the frequency spectrum [9].

The graphics window of waterfall displays the frequency spectrum over time. As mentioned above, the delay can be changed between displaying the spectrums. Moreover, the common threshold level is set, producing the 256 shades of color. Furthermore, the dynamic extent of this pallet can be changed at any time as required, even when active. The graphics components showing the frequency spectrum and the waterfall are set to process events at the click of a mouse. This is done based on the alternative selected either for measuring frequency or for retuning the receiver to the place clicked. In order to search for new signals in the frequency spectrum in waterfall, an uneven threshold level can be set, which creates the so-called learning, or setting the maximum value for the spectrum line that shows during active teaching. The waterfall desktop is black after switching to operations and the only signals to show are those which are higher than the uneven threshold level. The block set in the uneven threshold level

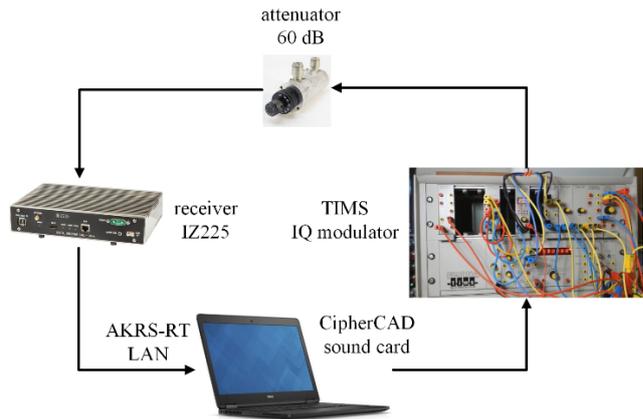


Fig. 14. Diagram for generating, modulating and processing APSK signals.

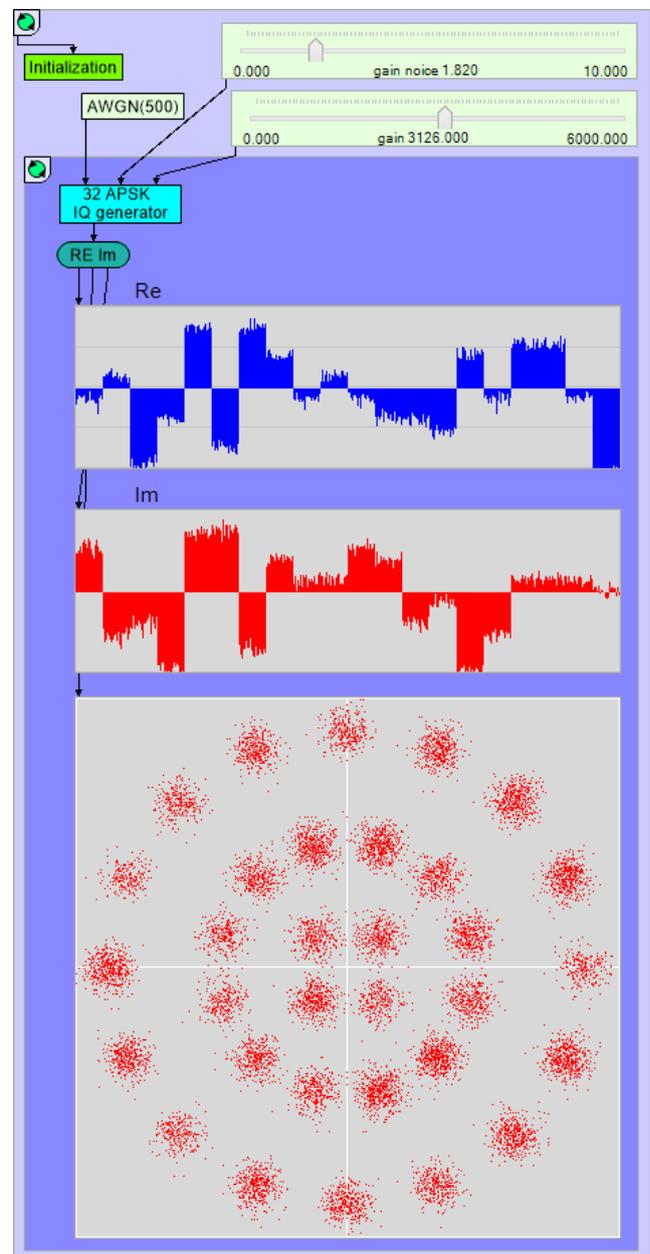


Fig. 15. Generating 32APSK IQ data in CipherCAD.

mode is marked as 5 in Fig. 11. The video [9] shows how to control the receiver as described.

This part will show another way of using CipherCAD in Testbed. The model in CipherCAD will not be used to process the signals received, but the opposite, to create signals in IQ format in real time and to send these signals to an external modulator. The connections diagram is in Fig. 14. IQ samples are generated in the model 32APSK created in CipherCAD. The model is shown in Fig. 15.

The parameters generating the 32APSK signal are set by choosing the output device, in this case a sound card. A sampling frequency of $f_s = 40$ kHz was used, along with a symbols rate of 2,000 symbols per second. The data is generated randomly in blocks of 100 bits, i.e. in lengths of 10 ms. The IQ samples of the 32APSK signal are generated based on the standards for DVB-S2 [10]. AWGN noise with optional SNR is then added to the samples. These IQ samples are sent to the sound card in 10 ms blocks in 16 signed integer format using the PlaySound (Data, bitsInSample, samplingRate, channelCount, deviceID) command. Figure 15 shows the IQ samples against time and in a constellation diagram. The I samples are sent to the left channel of the sound card, while the Q samples are sent to the right channel of the sound card. The analogue signal from the sound card is transferred to the IQ modulator, which is set up in the hardware system TIMS (Telecommunication Instructional Modelling System). TIMS is a flexible and versatile communications trainer that incorporates all of the instruments needed to quickly and easily carry out every communications laboratory experiment required in a range of courses, from technical college to university level [11]. The IQ 32APSK samples are modulated at a carrier frequency of $f_c = 100$ kHz. The modulated

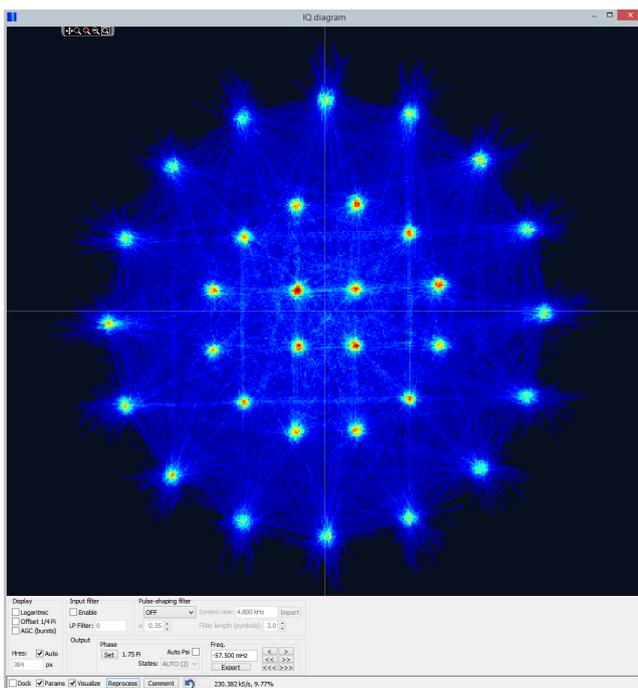


Fig. 16. The 32APSK signal received in AKRS-RT.

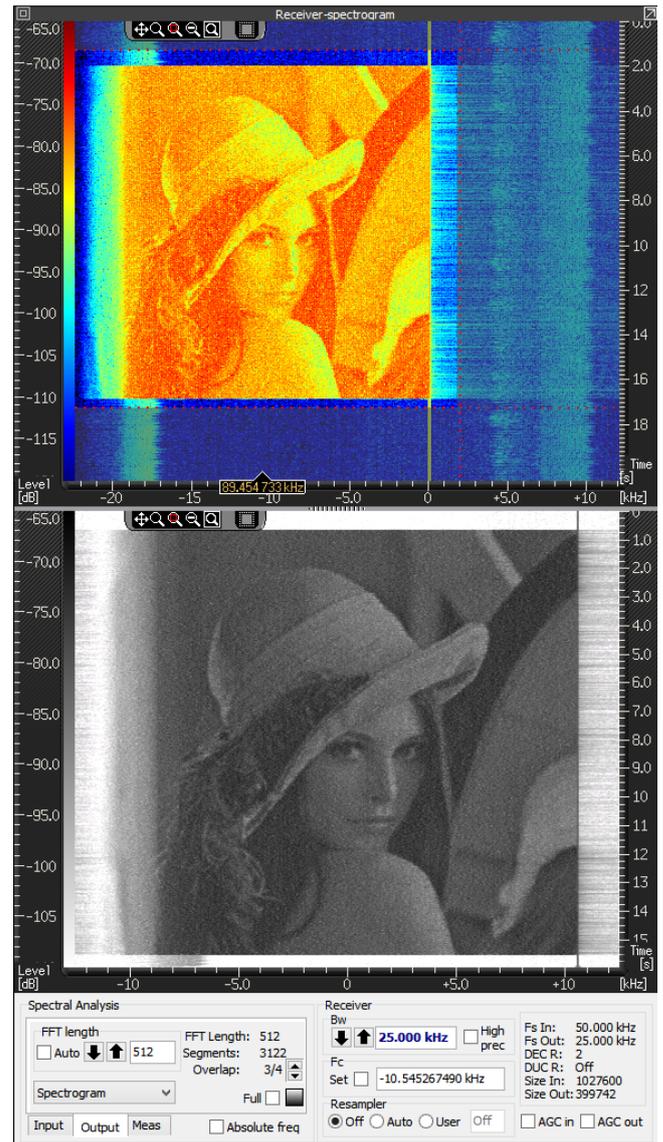


Fig. 17. The OFDM signal received in AKRS-RT.

signal from the TIMS output is transferred via a 60 dB attenuator to the input of the IZ225 receiver. The signal received is then processed on the computer in the program AKRS-RT (Radiosignal analysis and classification application – Real Time). The modular SW application AKRS-RT is intended for online/offline radio-signal analysis, classification and decoding. AKRS-RT is applicable as a standalone installation or as a module of the complex software solution SYMON SW., deployed in the Czech Armed Forces since 2010 [12]. The constellation diagram of the signal received in AKRS-RT is shown in Fig. 16. It is of interest that AKRS-RT works on the same computer as the 32APSK generator in CipherCAD.

This example shows the possibilities of the workplace to create any signal with the necessary parameters, modulate this signal and transfer it to the input of the SDR receiver and process it. An example is the creation of an OFDM signal in CipherCAD. The information will not be contained in the individual subcarriers. However, an image will be transmitted in the envelope of the frequency spectrum,

which can be displayed using waterfall. Although this variant has no practical use, it can successfully engage students in the field of digital modulations. The displayed capture of the OFDM signal in AKRS-RT can be seen in Fig. 17. The video captured in CipherCAD can be seen in [9]. This model demonstrates the comprehensive capabilities offered by the CipherCAD Testbed workplace.

4. VoIP Telephony

The most common way of communicating between people is by voice. Over the last decades, there have been huge developments in telephone systems, especially in mobile cell systems. There is currently a change from a purely telephone network to networks based on IP protocols. Nowadays it is extremely common to transfer voice using Voice over IP technologies between anywhere in the world. CipherCAD can communicate with any device via IP protocols and communicate simultaneously with a PC's sound card. These possibilities have given rise to a number of models from the simple model of the IP telephone, the model of the SIP switchboard, the model for testing SIP devices, the model for changing voice packets in real time, to the model for a secure call with the possibility of incorporating steganographic information [13], [14]. Selected models will be introduced in more detail in this section.

The first model is a simple IP phone using SIP signaling protocol, which can be seen in Fig. 18. This model facilitates establishing, modifying, and terminating calls with any SIP phone based on direct phoning using an IP

address. The A-law version of the simplest codec G.711 is used for coding the voice. The model overview represents a call carried out in a laboratory between CipherCAD and the IP phone Linksys SPA922.

The Initialization block is in the top left corner of the model. Into the text field, the users enter the phone number and IP address of the phone they want to connect with. After running the Initialization block, this will also start the blocks Receiving SIP Packets, SIP Signalization and Analysis Tool. The aim of the block Receiving SIP Packets is to monitor the selected port, in our case port 5060, and filter all packets transferring the SIP signaling protocol. This block is responsible for processing and generating all signaling messages. The model makes it possible both to start the call and to accept any call. The flow of basic signaling methods and answers can be monitored in the viewing components inside the block, including a complete list of signaling messages.

The green Start Call block is used to start or receive calls. If the user starts the call, the message INVITE is sent after clicking on the block. If, however, the call is initiated from other side, the model sends a response Trying and Ringing after receiving the message INVITE and starts ringing. The Start Call block has to be selected to receive the call. While making the connection, the model with the phone exchanges information on the port numbers that will be used for accepting the voice RTP stream. Once the connection has been made successfully, the Receiving Voice and Sending Voice blocks are initialized. The Receiving Voice block receives packets of RTP stream from the phone,

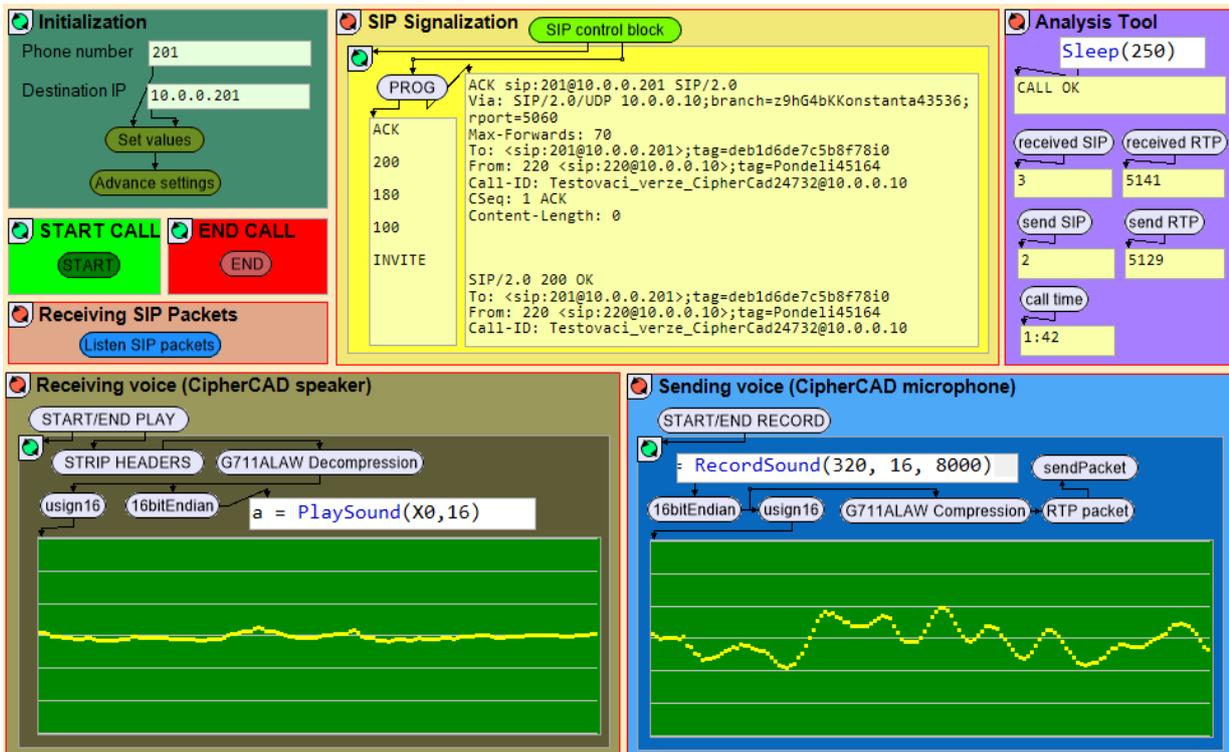


Fig. 18. Simple model of an SIP phone.

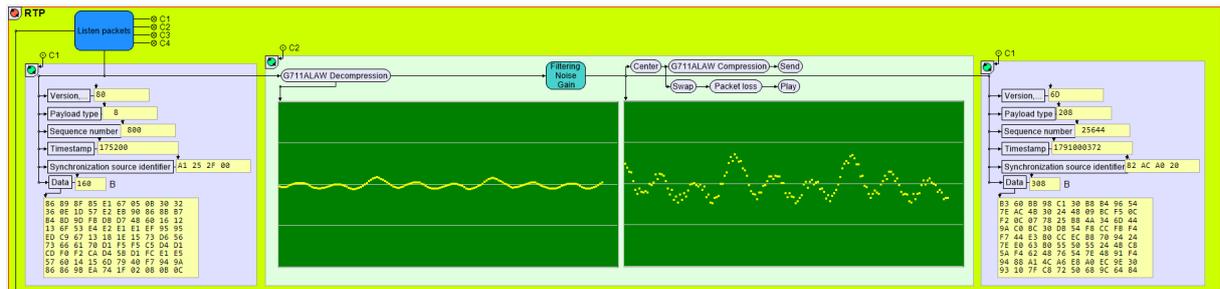


Fig. 19. Changes to voice packets by increasing the volume and adding noise.

removes the protocol headers, decompresses G.711 and sends the voice samples received to the sound card. The viewing component inside the block displays the current waveform of the voice received. In contrast, the Sending Voice block captures voice samples from the microphone, compresses them in codec G.711, adds the headers of the RTP protocols and sends them to the phone using the SendUDP() function.

The purple block Analysis Tool in the top right corner displays information on the status of the phone and information on the progress of the call. The number of received and sent signaling and RTP packets are displayed along with the length of call. This information is updated every 250 ms. The red End Call block allows the user of the model to end the call.

The second model presented facilitates changes to the voice packets in real time, i.e. changes to packets transferring voice between any two mobile phones. This model uses two SIP accounts managed by a public provider of phone services www.802.cz on the phone numbers 910 800 818 and 511 115 347. In the Initialization block, the user sets the Interface ID of the PC network interface and its IP address. The whole model will communicate with the provider of phone services via the selected interface. Moreover, it will select and set any phone number which the call is transferred to. After activating the Initialization block, a number of blocks are initialized responsible for receiving, filtering, processing and sending SIP signaling messages, but also displaying important events and parameters. The whole model can be found in the link [9].

The Initialization block first runs the registration processes of the above-mentioned SIP accounts at the providers of phone services. In both cases, CipherCAD acts as the SIP UA (User Agent) end device. The registration process is provided by the Register 802 phone blocks. These are responsible for sending Register signaling messages, processing the answers, the calculations of authenticated hash chains and regular periodic pre-registration. The user has information on the progress and results of the registration processes available in the viewing components. Next, the CALL Signaling block is initialized. This block waits for the INVITE incoming signaling message. This arrives as soon as the number 910 800 818 is dialed by any mobile phone. The provider of phone services forwards the request to the model created, since the end phone is expected here. The model responds with the message Ringing and simul-

taneously sends the INVITE request from the second phone number 511 115 347 to the phone number entered by the user. If the call is picked up, the call will be connected between the mobile phones, during which all voice and signaling messages will pass through the model. Information needed for forwarding RTP streams is displayed in the viewing component. If any of the mobile phones closes the call, the connection will be ended.

If the mobile phones are successfully connected via the model, other block are initialized in the list Voice packets. The most important block is the RTP block. This block receives all RTP packets from one SIP account and sends them to another SIP account. For each received packet, information is displayed from the header of the RTP protocol, the codec G.711 is decompressed and the voice waveform is displayed. The users can change the voice samples. They have simple operations available which should be easy to distinguish upon hearing. They can turn up or down the volume, add noise, set the loss of the packets or filter the voice using some of the low-pass filters offered. The voice can also be forwarded without changes. They can change or combine the individual operations during the call through the simple click of the mouse. As soon as the samples of voice have been processed, the resulting waveform of the changed voice is displayed, the resulting voice is again compressed by codec G.711, new information is displayed from the RTP header and the data is sent to the second phone. Figure 19 shows the changes to the waveform of a call by increasing the volume and adding noise.

5. Conclusion

The aim of the article was to present the unique workplace CipherCAD Testbed and use its features for analyzing, verifying, testing and teaching complex problems from the field of communications systems. Section 2 introduced the core of the workplace CipherCAD. This application is constantly developing and it can be expected that new features can be expected and the user interface will be perfected. The ability of the application to communicate via a USB interface would be a welcome development.

The next section was devoted to an example of controlling the IZ225 receiver and processing the signals re-

ceived in different variants. The models presented facilitate processing the IQ samples received, displaying them in the spectrum or waterfall and storing data for off-line analysis. One of the biggest advantages of the models presented was the real-time reaction at the click of a mouse, making it easy to retune the frequency and to define the parameters of the spectrum. Users also appreciate the ability to set their own dynamic signal level, set uneven threshold level and change the speed of transfer to waterfall. These features are especially useful when investigating signals with slowly changing frequencies, signals appearing in random for a short time and atypical signals. The end of the section showed the interconnection of workplace where CipherCAD generated IQ samples of the signal. The application facilitates the selection of the modulation required, symbol rate and the level of SNR. IQ samples generated in this way are then sent to an external hardware modulator. The modulated signal is transferred to a receiver with the required attenuation. The received signal is then processed in AKRS-RT.

The real-time features of CipherCAD have also been demonstrated in the last section devoted to VoIP. This section introduces two models. This first model is for connecting and carrying out a call with any SIP phone. The second model makes it possible in CipherCAD to make a call between two mobile phones and to change and display the voice packets exchanged by the phones in real time.

The CipherCAD Testbed workplace is suitable for creating, processing and analyzing radio signals and other tasks in the field of communications, including ensuring secure communications in the various layers of the RM OSI model. In the near future, we expect the workplace to have another SDR transmitter.

Acknowledgments

This work has been supported by the project for development of K-207 department, University of Defense, Ministry of Defense of the Czech Republic – development program “AIROPS”.

References

- [1] BURG, A., CHATTOPADHYAY, A., LAM, K.-Y. Wireless communication and security issues for cyber-physical systems and

- the internet-of-things. *Proceedings of the IEEE*, 2018, vol. 106, no. 1, p. 38–60. DOI: 10.1109/JPROC.2017.2780172
- [2] AHMAD, I., SHAHABUDDIN, S., KUMAR, T., et al. Security for 5G and beyond. *IEEE Communications Surveys & Tutorials*, 2019, vol. 21, no. 4, p. 3682–3722. DOI: 10.1109/COMST.2019.2916180
- [3] SARMILA, K. B., MANISEKARAN, S. V. A study on security considerations in IoT environment and data protection methodologies for communication in cloud computing. In *International Carnahan Conference on Security Technology (ICCST)*. Chennai (India), 2019, p. 1–6. DOI: 10.1109/CCST.2019.8888414
- [4] DULIK, M., DULIK, M. jr. Cyber security challenges in future military battlefield information network. *AiMT Advances in Military Technology*, 2019, vol. 14, no. 2, p. 263–277. DOI: 10.3849/aimt.01248
- [5] RIIHONEN, T., KORPI, D., TURUNEN, M., et al. Tactical communication link under joint jamming and interception by same-frequency simultaneous transmit and receive radio. In *IEEE Military Communications Conference (MILCOM)*. Los Angeles (CA, USA), 2018, p. 1–5. DOI: 10.1109/MILCOM.2018.8599793
- [6] KLIMA, V., PLATENKA, V. The cryptographic software tool CipherCAD and cryptanalysis. In *Proceedings of Security and Protection of Information 2011*. Prague (Czech Republic), 2011, p. 54–65. ISBN: 978-80-7231-777-6
- [7] INTRIPLE. IZ225. 1 page. [Online] Cited 2021-04-26. Available at: <https://intriple.eu/product/show?productId=38>
- [8] INTRIPLE. IZ225 Programmer Manual – IZ225 SCPI Commands, Prague (Czech Republic), 2015.
- [9] CipherCAD. [Online] Cited 2021-04-26. Available at: www.platenka.cz
- [10] EUROPEAN TELECOMMUNICATIONS STANDARDS INSTITUTE (ETSI). *Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications. Part 1: DVB-S2*. European Telecommunications Standards Institute (ETSI), 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France, 2014 [Online] Cited 2021-04-26. Available at: https://www.etsi.org/deliver/etsi_en/302300_302399/30230701/01_04.01_60/en_30230701v010401p.pdf
- [11] EMONA TIMS. [Online] Cited 2021-04-26. Available at: www.emona-tims.com
- [12] URC SYSTEMS. *AKRS RT - Radiosignal Analysis and Classification Application*. [Online] Cited 2021-04-26. Available at: www.urc-systems.cz/en/product/akrs-rt
- [13] MAZALEK, A., VRANOVA, Z., PLATENKA, V., et al. Testing of incorrect SIP messages processing. In *International Conference on Military Technologies (ICMT)*. Brno (Czech Republic), 2017, p. 419–423. DOI: 10.1109/MILTECHS.2017.7988796
- [14] PLATENKA, V., MAZALEK, A., VRANOVA, Z. The transfer of hidden information in data in the AMR-WB codec. In *Communication and Information Technologies (KIT)*. Vysoké Tatry (Slovakia), 2019, p. 1–5. DOI: 10.23919/KIT.2019.8883461