

Breaking the Code: Hacking the Alpha Secure Communications Hardware Platform

Matthew Hicks
Department of Computer Science
University of Illinois
mdhicks2@uiuc.edu
© Copyright 2008

Abstract

This paper describes two hardware attacks implemented and tested on the Alpha secure communications platform. One of the attacks, Kill Switch, is a denial-of-service attack that prevents Alpha from functioning correctly after being triggered by a predetermined input sequence. Like a switch, an attacker can turn Kill Switch on and off at will, making detection more difficult and masking the source of the error. The second attack is an information leakage attack entitled Piggyback. As the name implies, Piggyback modulates the encryption key inside the normal RS-232 communication stream of Alpha. Unknowing receivers tuned to the slower frequency never observe the high frequency key message encoded in the signal, while a knowing receiver trivially decodes the secret message. This paper presents, in detail, the design and implementation results of the two attacks. Also presented, is an evaluation for each attack, which highlights the minimal increase in area resources required for the attacks, the lack of timing impact, and the change in power consumption differences. The paper concludes with a look at other possible attacks with reasoning as to their value given the constraints of the competition.

1 Introduction

The goal of the competition is to insert a malicious circuit in the Alpha platform that performs some sort of attack on the secure communications system. Some provided examples of attack directions are denial-of-service, information leakage, and general behavior manipulation. A denial-of-service attack prevents the functioning of some component of the system when triggered. An information leakage attack either stores sensitive information inside the device, allowing an attacker with physical access the ability of extract the information at a later date, or transmits it off the device to an attacker in real-time. The behavioral attack is more like a combination of the two other attack types.

Whatever the type of attack, it should use a minimum of additional resources, not consume much more power than the original “golden” system, and pass the verification test, as these are the three main judging criteria used in evaluating an attack. Above and beyond these predetermined metrics of attack success, a few other design goals become critical when building attacks. They are the power of the attack, the practicality of the attack, and the difficulty of detection. Many of these properties are interrelated in a complex manner. For instance, an attack with a larger logic utilization will generally be more powerful than an attack with a smaller footprint. Another example is the tight relationship between power and area. The larger

the attack footprint, the more power it tends to use. An exception to this rule is a larger attack that runs at a low frequency will use less power than a smaller attack that runs at a much higher frequency. The relationships between metrics are far too complicated to deal with all at once, so an attack designer must order the importance of each metric and progressively massage the attack design to fit each metric.

When designing attacks for real systems, the most important constraint is whether it will function in the wild. This means, if a proposed attack isn't practical, it isn't implemented. Section 4 describes attacks that are interesting, but, for one reason or another, are impractical in the context of the competition. Another deal breaking constraint is the need to pass verification testing. If an attack doesn't make it into use, it isn't practical, and thus not implemented. The four remaining metrics serve as implementation constraints, driving the direction of implementation. Of these, the power, the expressiveness, of the attack is the most important. Since an attacker can only implement a limited number of attacks, it is wise to select the attacks able to do the most damage or provide the most information. In hardware, more power attacks also are the more difficult attacks to defeat as they tend to be extremely intertwined in system functionality. This is a good property for maintaining control over a victim system, but means that attack design becomes more complicated. Once the set of most powerful attacks is chosen, the key constraint become the logic overhead of each attack. Attacks that use fewer gates are obviously better than those which use more. Empirical observation shows that these attacks tend to use less power and are harder to detect using conventional methods. Once the logic overhead of an attack is within the area budget, the power draw of the attack becomes the focus. The difference in power draw of a hacked system is generally not important because it is limited with the area and number of IO pins of the device. Excess power draw only becomes important when a defender use low-level malicious circuit detect methodologies like power draw analysis, thermal signature analysis, or EMI analysis. The final metric of detection difficulty is nebulous unless an attacker knows what detection methodology a defender will use.

Both Kill Switch and Piggyback represent an excellent balance of all design metrics. Kill Switch is an ideal example of a denial-of-service attack that prevents the transmitter from sending intelligible messages to the receiver. Kill Switch allows for remote, in-band attacks with almost no area overhead and little excess power draw. Kill Switch enables attackers to control the whether the system functions correctly or not, at will. Kill Switch doesn't affect the functionality of the system when not triggered and the triggering mechanism can be made impossible to trigger during normal operation. Piggyback, on the other hand, is an ideal example of an remote information leakage attack. Piggyback encodes a high frequency RS-232 compliant signal on the pre-existing low frequency RS-232 signal. The encoding is done in such a way that it takes little logic in the transmitter and the transmitted signal looks untainted to unaware receivers. This means verification is trivial.

Section 2 covers the high-level idea behind the Kill Switch attack, leadin to the implementation details and results. Section 3 does the same for the Piggyback attack. Section 4 discusses other attacks that are interesting but ineffective given the framers of the competition and the requirement of practicality. Section 5 presents the key lessons from the attacks presented here, with a look toward possible defenses. An appendix is included at the end paper to provide details on how to recreate and test the attacks using the files provide on the website [1].

2 Kill Switch

Kill Switch is a lightweight, but flexible, denial-of-service attack. Kill Switch is both lightweight and easy to construct as it leverages the device's preexisting receiver circuitry. Only a small amount of additional state memorization hardware is required. Kill Switch looks for special sequences of received data for patterns that match the trigger sequence. The trigger sequence itself is flexible, an attack designer chooses a sequence that won't be triggered during testing or is easy for the attacker to send. Sending the trigger sequence a second time resets the hack, removing its impact on the system. The target of the hack is also flexible. An attacker can attack the transmitter, the receiver, the encryption unit, or any other unit that has value to them. The version of Kill Switch presented here uses a lightweight trigger to attack the transmitter, scrambling the key value reported to receivers, preventing successful decryption. This choice makes the transmission medium look like the error source, diverting attention away from the hack.

2.1 Implementation

The Alpha secure communications platform doesn't use the receiver in its current incarnation, so any transmission is seen as a trigger of the hack. This greatly reduces the device utilization of the attack circuitry, as all that is required is a few one-bit registers, a slow clock, and a debouncing circuit to avoid spurious triggering. The slower the clock that drives the trigger, the smaller the logic requirement of the debouncing circuit. To minimize the size of the debouncer, without using extra clock resources, the slowest clock in the system is used to drive the attack sequence recognizer. This clock comes from the seven-segment display driver. Using it reduces the debouncing register to only five bits from 200 if the top-level clock were used.

2.2 Results

Figure 1 is a chart of the utilization of the golden Alpha secure communications platform. It lists all of the major blocks inside of the FPGA, a Xilinx Spartan-3E, the amount of each resource available and the amount used by the design. Figure 2 details the estimated power usage of the design given the placed-and-routed design and some probabilities on input-dependent characteristics like flop toggle rates. The key here is the total power and the current draw for each voltage level. These two figures serve as a comparison point for evaluating the implementations of Kill Switch and Piggyback. Notice that 98% of the device's logic area is used by the unmodified circuit, so an attacker must be judicious when selecting and implementing attacks. On this note, an attacker may also optimize the victim circuit to make room for or hide the overhead of their attack. This strategy is used in Piggyback to great effect.

Device Utilization Summary				[1]
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	1,469	4,896	30%	
Number of 4 input LUTs	4,044	4,896	82%	
Logic Distribution				
Number of occupied Slices	2,401	2,448	98%	
Number of Slices containing only related logic	2,401	2,401	100%	
Number of Slices containing unrelated logic	0	2,401	0%	
Total Number of 4 input LUTs	4,197	4,896	85%	
Number used as logic	4,044			
Number used as a route-thru	153			
Number of bonded IOBs				
Number of bonded	46	108	42%	
IOB Flip Flops	5			
Number of RAMB16s	8	12	66%	
Number of BUFGMUXs	4	24	16%	
Number of DCMs	3	4	75%	
Number of MULT18X18SIOs	2	12	16%	

Figure 1: Device utilization for the unmodified Alpha platform

Power summary		I (mA)	P (mW)
Total estimated power consumption			120

Total Vccint	1.20V	66	79
Total Vccaux	2.50V	12	30
Total Vcco25	2.50V	4	11

Clocks		40	48
IOs		0	0
Inputs		1	1
Logic		3	3
Outputs			
Vcco25		3	7
Signals		7	8

Quiescent Vccint	1.20V	16	19
Quiescent Vccaux	2.50V	12	30
Quiescent Vcco25	2.50V	2	4
Thermal summary			
Estimated junction temperature			29C
Ambient temp			25C
Case temp			29C
Theta J-A			38C/W

Figure 2: Estimated power requirements for the unmodified Alpha platform

Device Utilization Summary				[1]
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	1,460	4,896	29%	
Number of 4 input LUTs	4,048	4,896	82%	
Logic Distribution				
Number of occupied Slices	2,374	2,448	96%	
Number of Slices containing only related logic	2,374	2,374	100%	
Number of Slices containing unrelated logic	0	2,374	0%	
Total Number of 4 input LUTs	4,190	4,896	85%	
Number used as logic	4,048			
Number used as a route-thru	142			
Number of bonded IOBs				
Number of bonded	47	108	43%	
IOB Flip Flops	20			
Number of RAMB16s	8	12	66%	
Number of BUFGMUXs	4	24	16%	
Number of DCMs	3	4	75%	
Number of MULT18X18SIOs	2	12	16%	

Figure 3: Device utilization for the Alpha platform with the Kill Switch hack

Power summary		I (mA)	P (mW)
Total estimated power consumption			121

Total Vccint	1.20V	67	81
Total Vccaux	2.50V	12	30
Total Vcco25	2.50V	4	10

Clocks		40	48
IOs		0	0
Inputs		1	1
Logic		2	3
Outputs			
Vcco25		3	6
Signals		9	10

Quiescent Vccint	1.20V	16	19
Quiescent Vccaux	2.50V	12	30
Quiescent Vcco25	2.50V	2	4
Thermal summary			
Estimated junction temperature		30C	
Ambient temp		25C	
Case temp		29C	
Theta J-A		38C/W	

Figure 4: Estimated power requirements for the Alpha platform with the Kill Switch hack

Figure 3 shows the area utilization for the Kill Switch enabled version of Alpha. Comparing the values in Figure 3 to those of the untainted platform listed in Figure 1, it becomes clear that there is little difference. On first glance, it seems odd that the malicious circuit uses

fewer registers than the original, but there is a reason for this behavior. Being a research/competition grade platform, Alpha comprises a bunch of open source hardware blocks held together by glue logic. Alpha, as the source code states, isn't production quality, thoroughly vetted code. While it kind of functions correctly, slight changes to the code break it, in hard to define places. This was the case for Kill Switch, which shouldn't break anything as it is fairly self-contained. The initial implementation caused the VGA port not to function. To fix this error, both the top-level state machine and the clock dividers were reconstructed, using professional coding practices, including some minor optimization. The end result is a correctly functioning hacked system with about the same area, power, and operating frequency.

Figure 4 shows the power report for the Kill Switch enabled system. Comparing this figure to Figure 2, Kill Switch requires a paltry 1% more power, within the margin of variation. Since the power is so close, the thermal numbers are practically identical. These results mean that detection mechanism that look at side channel information like device frequency, area, power, or thermal footprint are going to be ineffective in detecting Kill Switch. Given the amount of recoding that took place and the limited impact on power and thermal radiation, a higher-level takeaway message is that using either as a defense foundation is ill-advised.

3 Piggyback

Piggyback is an example of an information leakage attack. The goal of the attack is to allow valid, normal, transmission between sender and receiver while encoding the encryption key used to encode the message onto this transmission. An aware receiver knows where to locate the hidden message and extracts the encryption key from the message, while an unaware receiver only observes the encrypted message, not the key. How to accomplish this secret encoding depends on the communication method

3.1 Implementation

The minimum message length is 1 byte for the key identifier, 16 bytes for the smallest message, and 15 bytes in end of message padding. The AES key is 16 bytes long. Since each message may use a different key, ideally each message carries the key needed to decode it. The RS-232 protocol transmits data 8-bits at a time, prefixed by a start bit and postfixed by one stop bit. The start and stop bits allow the receiver to tune itself to the transmitters clock which is required due to the asynchronous nature of the protocol. A start bit is a logical 0, while a stop bit is a logical 1. Since these are the only bits with a constant value, they are the optimal place to encode the key. The key is encoded 8-bits at a time, one key byte per message byte. The initial implementation of Piggyback encode the key byte on the start bit, having the two transmissions share synchronized start bits. This choice didn't work due to the way modern RS-232 receivers dynamically capture the input using automatic tuning. The stop bit, which isn't visible to receiver, only exists to separate consecutive transmissions to make-up for clock difference between send and receiver. The start of the high frequency transmission is offset from the start of the stop bit to prevent corrupting the low frequency receiver. With this framework, a complete encryption key can be transmitted twice in a minimum sized message.

Replicate transmitter state machine and combine the resultant RS-232-like signals to form a single output signal. Figure 5 shows how the malicious transmitter combines the signals

from the slow and fast state machines in such a way that a receiver won't detect any difference from the original RS-232 signal. An aware receiver with software filtering based on the output from the lower speed RS-232 receiver can accurately decode information from the fast signal. Another method of obtaining just the data encoded in the higher baud signal is using traditional signal filtering, either analog or digital. Using a high-pass filter tuned to accept 115200 will eliminate the 9600-baud signal's interference. Since 9600 baud is twelve times slower than 115200 baud, eleven harmonics of the 9600-baud signal are attenuated, practically eliminating it. The easiest method for pulling-out the high speed message is using a specially tuned hardware receiver.

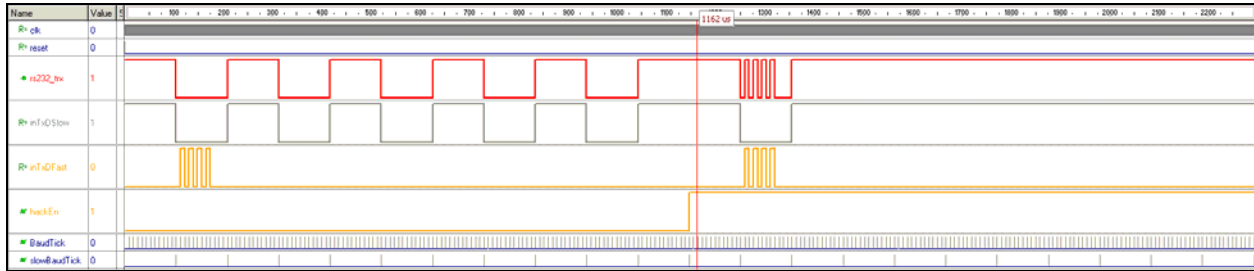


Figure 5: Waveform showing the encoding of the high frequency malicious RS-232-like signal on the low frequency RS-232 signal

3.2 Results

Device Utilization Summary				[-]
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	1,440	4,896	29%	
Number of 4 input LUTs	4,180	4,896	85%	
Logic Distribution				
Number of occupied Slices	2,373	2,448	96%	
Number of Slices containing only related logic	2,373	2,373	100%	
Number of Slices containing unrelated logic	0	2,373	0%	
Total Number of 4 input LUTs				
Number used as logic	4,180			
Number used as a route-thru	90			
Number of bonded IOBs				
Number of bonded	46	108	42%	
IOB Flip Flops	19			
Number of RAMB16s	8	12	66%	
Number of BUFGMUXs	3	24	12%	
Number of DCMs	3	4	75%	
Number of MULT18X18SIOs	2	12	16%	

Figure 6: Device utilization for the Alpha platform with the Piggyback hack

Power summary		I (mA)	P (mW)
Total estimated power consumption			126

Total Vccint	1.20V	71	85
Total Vccaux	2.50V	12	30
Total Vcco25	2.50V	4	11

	Clocks	40	48
	IOs	0	0
	Inputs	1	1
	Logic	4	5
	Outputs		
	Vcco25	3	7
	Signals	11	13

Quiescent Vccint	1.20V	16	19
Quiescent Vccaux	2.50V	12	30
Quiescent Vcco25	2.50V	2	4
Thermal summary			
Estimated junction temperature		30C	
	Ambient temp	25C	
	Case temp	29C	
	Theta J-A	38C/W	

Figure 7: Estimated power requirements for the Alpha platform with the Piggyback hack

To get a correct implementation of Piggyback, much recoding of the original system was required. The utilization number shown in Figure 6 support this, showing a marked decrease in registers and occupied slices with a 3% increase in computational logic. The logic increase is due to the replicated transmitter state machine. This 3% overhead could be hidden given more time to recode the rest of the system. The areas where Piggyback has negative overhead are trivial to obfuscate by adding unused registers.

While the thermal numbers, shown in Figure 7, are identical to the golden system, total power draw increases by 5%. Further optimizing the other parts of the system would likely reduce this overhead to a number within a normal variation.

4 Other interesting hacks

During the process of selecting which methods to implement for the competition, many methods came-up, but were rejected because they were impractical given the constraints of the competition. A review of the rejected ideas may be helpful to others in evaluating their hacking options. A survey of other ideas is also valuable as a source for hack ideas given a design used in slightly different circumstances.

First, a review of some of the more limiting constraints will serve as an eyepiece that will aid in evaluating proposed attacks. For one, the attacker can only affect a limited subset of all possible communications devices. As a check, the customer may compare results produced by different vendors. This requires that all devices have the same observable behaviors at the

outputs for a given input. Side channel sources of information like power draw and thermal signature are not as critical as there is a range of acceptable values for these items due to variations in things like process and manufacturing. A second limit on attack options is how the device will actually be used in the real-world. While this is a competition with little testing and actual use, a valuable attack should port seamlessly to the real-world. Communications devices, in general, are, one activated, always on and polling their environment for opportunities to establish communication with another device. Real-devices must abide strictly to the communication protocol as host will reject malformed transmissions. This also means that use of the communication channel is potentially visible to the hackees. Meaning, most in-band attacks will be visible. The last constraint is the one of limited access. While access to the communication channel is wide open, access to the devices will most likely be limited to none. This is especially true for fully-functional, powered-on devices. Allowing access to the location of transmission nullifies the need for special secure communications devices, as the attacker can just read the paper or screen that the sender is using to create the message. A captured building or vehicle may provide access to an encryption system, but these systems will likely be non-functioning and most definitely not on. This means the memory will be clear of all private data.

4.1 Proximity Attacks

Local attacks are those, which require close proximity to the device in order for that attack to take place. Examples of this style of attack are data-based modulation of electromagnetic waves, including the power signature, thermal footprint, or using an internal feedback loop to create an oscillator and broadcasting using an internal antenna. While the latter is extremely difficult to do in FPGAs, these options never affect outputs and don't require any additional pins on the FPGA. These circuits are great for things like debugging [2], but fail as a good attack venue due to previously mentioned constraints. It is unrealistic and burdensome that an attacker be required to have close proximity to the victim. A look at software attacks confirms this point.

4.2 Surplus Information

Allowing for the transmission of data opens the door to remote attacks. While these attacks are clearly more powerful than proximity-based attacks, they are also easier to detect. Information must be visible at one of the outputs, namely the communication subsystem output; otherwise, an attacker wouldn't obtain the data. This means that these attacks are inherently more visible. This makes the choice of how to communicate the data to the outside world critical. The easy decision is to utilize the communications subsystem to send additional information. This is light on resources, but trivial to detect. Since the receiver is monitoring the communications channel, he will likely see the extra communications and cease further communications in fear of a corrupt channel. This clearly violates the given constraints. Use of other channels for backdoor communication is even easier to detect, and unlikely to work, as most systems are *not* constructed with superfluous modules that would allow such long-distance communication.

4.3 Algorithm Attacks

A very interesting attack is an attack on the algorithm the device uses to encrypt and decrypt the data. Such a hack would augment the algorithm in such a way as to introduce a fundamental weakness that would allow remote attackers to break quickly the encryption on saved transmissions. The transmissions would remain technically encrypted, meaning both sides would see no difference in communication, but the attacker would know how to quickly find the shared key given enough data. This is ideologically similar to the flaws found in the Wireless Encryption Protocol (WEP) [xxx] and the method used for WEP cracking [xxx]. While this idea is great in that there is actually no real change to the system for a defender to observe, it violates one of the main constraints of the competition; you only control a subset of the communication devices. For an attack of this ilk to work, all devices must use identical encryption and decryption protocols.

5 Conclusions

Both Kill Switch and Piggyback prove that it is possible for an attacker to implement an array of practical, hard to detect, high-value attacks in a given hardware platform. While it is difficult for an attacker to limit the footprint of the attack in terms of resource utilization and power at the same time, it is still possible. Given enough time, an attacker can easily modify the hardware such that most side channel symptoms of a malicious circuit are removed. This was clearly demonstrated in both attacks presented here, which had a relatively small power signature above the golden design and little logic overhead. These resulted in a heavily constrained scenario with less than a month of part-time, single person, development. Although, working on a buggy research platform does make implementing an attack much more difficult as things that barely work tend to break with only minor changes to the code. This may lead to very interesting defense that leverage this observation.

A key take-away from the presented attacks is for those who wish to defend their hardware from such malicious circuits. What can be learned is that many low-level, side-channel information defenses only catch poorly crafted attacks and thus aren't suitable as a long-term or effective defense. A better defense option is a high-level hardware based defense that leverages behavioral information and other semantic information to verify dynamically the functionality of the modules in a system. Although, as Piggyback demonstrates, there will always be some attacks that no general methodology will detect or prevent.

References

[1] <http://www.firefalcon.com/CSAW08/>

[2] Paul Dillien, "How to defend against the cloning of your FPGA designs", Programmable Logic DesignLine, September 2008.

[3] IEEE, "IEEE 802.11-1999: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", <http://standards.ieee.org/getieee802/download/802.11-1999.pdf>, 1999.

[4] Michael Ossmann, "WEP: Dead Again", <http://www.SecurityFocus.com/infocus1814>, December 2004.

Appendix

A.1 Implementing Kill Switch or Piggyback

Each hack has its own implementation directory and its own set of implementation scripts. Navigate to the appropriate directory and if running Windows, execute `xilinx_tools.bat`, if running in Linux, execute the `xilinx_tools` script. These scripts contain shell commands that run the synthesis, mapping, and place and route programs using Xilinx ISE 10.1.02 tools in a batch (non-GUI) mode. After implementation completes, the scripts cleanup the directory, removing all extraneous files. The `.map` and `.par` files contain resource utilization information and the timing score for the design. To program the FPGA, run Xilinx Impact, via the command line or by navigating to it via the start menu. If using the onboard USB programming option on Windows, run the Digilent Adept ExPort program. In either program, Impact or Export, select and load the `.bit` file located in the implementation directory. The remaining implementation files in the directory are useful for working with other Xilinx tools like static timing analysis or advanced power analysis.

A.2 Running the Kill Switch Demo

Before running any programs on the host PC, ensure that the BASYS board has the RS-232 Pmod unit attached to port A and the RS-232 cable is connected between the BASYS board and a serial port in the host PC. The programs work with both traditional serial ports and RS-232 to USB converters, which emulate a serial port, so using a serial adapter is acceptable. Just make sure that `enc_verifier.c` and `sendSerial.c` reflect the appropriate serial device. Also, make sure that you have read and write access to the serial port.

To run the demonstration program, first navigate to the demo directory, which is a subdirectory of the Kill Switch implementation directory. Type “make linux” to make the Linux version of the trigger program or “make windows” to make the Windows version. Then, execute the appropriate trigger program for your operating system. The trigger program triggers the attack, turning the hack either on or off. The system starts in the golden (unhacked) state. Running the trigger program corrupts the output of the transmitter, which corrupts the output of the terminal on the host PC, as seen by running the `enc_verifier` program in conjunction with a triggered system.

A.3 Running the Piggyback Demo

As of right now, the demo software needed to exercise the Piggyback hack only runs correctly in a Windows environment. To compile the program, navigate to the demo directory in the Piggyback implementation directory, and type “make” (requires a cygwin-like environment). Before compiling the program, ensure that the reference serial device accurately reflects the port connected to the Alpha platform. The resulting program, `keyGrabber.exe`, reads data from the RS-232 port, filtering the encryption key value from the data values. A 16-byte reference key is the final output of the program. Note that the key may require barrel shifting several bytes to the right when the trailing bytes of the key are zeros. Also, note that the key is in reverse or with respect to byte ordering.

During testing, where only one serial port is available, two passes are required, one pass at 9600 baud, using `enc_verifier`, to capture the message data and a second pass at 115200 baud, using `keyGrabber`, to capture the encryption key. For production use, an attacker may split the RS-232 signal delivering it to two different serial ports on the host PC. A single program that monitors each serial port with a thread combining the results from each thread to form a decrypted message is feasible.