BLINK: Securing Information to the Last Connection

Scott Craver, Yu Chen*, Hao Chen, Jun Yu, Idris M. Atakli
Dept. of Electrical & Computer Engineering, SUNY - Binghamton, Binghamton, NY 13902

ABSTRACT: Robust cryptography provides confidentiality and integrity for information transferred between peers. However, the decrypted plaintext in the memory of a receivers' computer is vulnerable—both to surveillance at the endpoints, and users who choose to forward confidential information. In this paper, we proposed a novel scheme called BLINK, which uses a reconfigurable hardware based decoder which operates on the link between a computer and its display. It moves the decryption outside the computer, preventing plaintext stealing, forwarding, screen capture and printing. Currently we are implementing the BLINK scheme on top of Altera FPGA board with Digital Visual Interface (DVI) ports, the correctness, effectiveness, and the performance will be evaluated through experiment.

Keywords**:** Data Security, Reconfigurable Hardware, FPGAs.

1. INTRODUCTION

Information security is an essential requirement in communications. Robust cryptography has been applied widely to provide confidentiality and integrity for information transferred through the communication channels. However, messages stored in the sender's or receiver's computer are still susceptible to interception. Many tools are available to read memory data and violate the confidentiality requirements. Not only is this susceptible to interception on the recipients' machine, but once decrypted, a document may be copied, printed, or otherwise leaked by a user who is insufficiently vigilant. To preserve secrecy, a message should remain encrypted until it is necessary for its intended recipient to view it.

In order to transfer confidential information securely online, the capability of exerting access control over decrypted documents is mandatory, such as prevention of copying, forwarding, or printing confidential materials. Under certain scenario, it is desired that a sent message bear some self-destruction feature: rendering a message unreadable beyond a set lifetime.

One solution to this problem is through customized software that exerts strict control over documents sent to users. However, people are used to using existing applications, and multiple operating systems (OS) or versions thereof. Requiring customized software limits users, and carries with it a restriction on the machines and OS versions that an organization can use. Furthermore, the security of software depends on the security of stored keys, which are vulnerable to reverse-engineering.

In this paper, we propose BLINK, or Brief-Lifetime Ink, a novel approach that encrypts sensitive data as bitmap images, encrypted in the pixel domain, and deciphers these images on the video link, beyond the reach of software. Confidential documents never exist as plaintext in either the computer system memory or video memory. This approach prevents the capturing of message from within the computer system, as well as preventing copying, pasting, screen capture or printing of a confidential document. It also allows the use of any Email, browser or other software, and does not require any specific operating system to work.

As shown in Figure 1, a hardware deciphering device is placed inline on the *Digital Visual Interface* (DVI) cable between the computer graphics card and the display. We select DVI because it was designed to supersede the old VGA interface. The device contains a set of decryption keys. When the user presses a button, the device identifies a scrambled image by a synchronization pattern in the DVI signal, and then decrypts the appropriate screen area in order to display the content of the secret message.

We adopt the *Transition Minimized Differential Signaling* (TMDS) link transmitter as the platform of our BLINK scheme [1]. TMDS is a technology for transmitting high-speed serial data and is used by the DVI and HDMI video interfaces, as well as other digital communication interfaces.

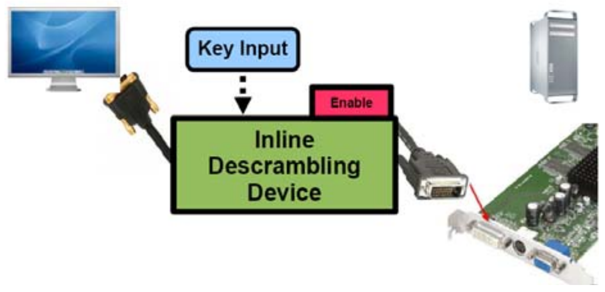
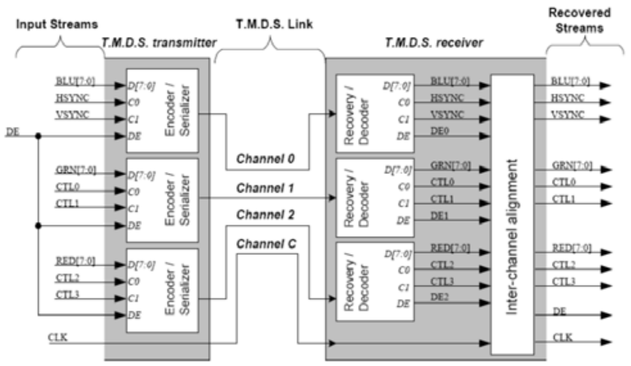


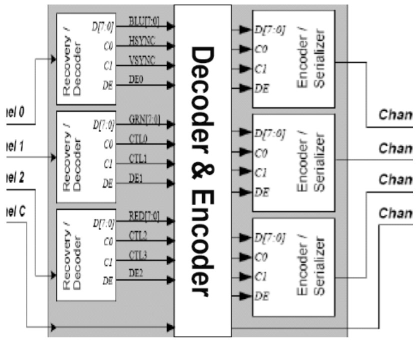
Figure 1. BLINK Architecture

2. TMDS ENCODER & DECODER

According to the specification of TMDS link, all data transmitted through the TMDS link are encoded, while our scheme does perform the decryption operation at the middle of the link. In order to achieve this goal, it is necessary to decode the received data through the TMDS link before the process is started. After the process is done, it is also necessary to re-encode the decrypted data in order for continuing transmitting to terminal monitor via TMDS link. Fortunately, since the decryption module features the same interface as the TMDS links, which means the standard encoders and decoders used for TMDS link are applicable for our application. Figure 2(a) shows the TMDS channel map [1], and Figure 2(b) demonstrates where decoder and encoder are inserted into the data path of the channel.



(a) The TMDS Channel Map



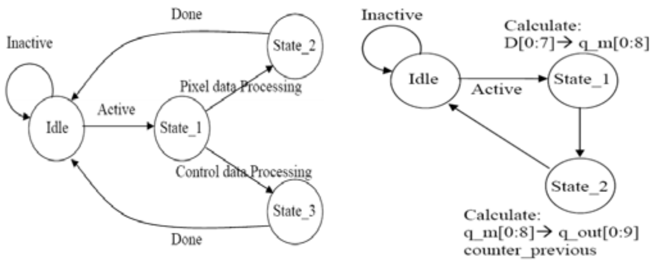
(b) Locations of Decoder & Encoder

Figure 2. Illustration of Decoder & Encoder Locations

As shown in the TMDS channel map of Fig.2(a), there are four channels. Except for the **channel_C** pin which does nothing but pass the reference clock at character rate, the three channels possess the same structure. From the perspective of hardware design, as long as one channel has been designed, the other two channels can instantiate the structure of this channel. Design tools will automatically handle the other issues.

We've successfully built a decoder and encoder with Verilog-HDL coding on the Xilinx ISE 9.1 design platform. The decoding algorithm can be represented by a Finite State Machine (FSM) as shown by Fig. 3(a). The FSM stays in idle if there is no activity. When certain event triggered the FSM, it starts to move forward based on its determination. It takes one clock cycle for a process in current state moving to the next state. It takes only two clock cycles for a process moving form idle back to idle. Comparing with software implementation, parallel execution is more efficient than serial execution.

The FSM of encoder is presented in Figure 3(b). Eight bits data along with other control signals will be fetched at the beginning. After one clock cycle, this eight-bit data will be expanded to nine-bit data. With certain determination, the FSM choose either flip the data bits to keep the DC-balance or do nothing. According to the different DE value, FSM will take the encoder to generate corresponding control data or pixel data. As a result, ten bits data will be finally output. Though the encode algorithm is more complicated than decode algorithm, the FSM of encoder takes the same clock cycles as the FMS of decoder to complete the processing.



(a). FSM of Decoder **(b). FSM of Encoder**

Figure 3. Illustration of FSM

We tested the decoder and encoder via simulation. Both decoder and encoder were hooked up directly. The rationale is that the raw data and the processed data should be the "same", if there is nothing being connected between the decoder and the encoder. The result of simulation verified that only four clock cycles were taken to execute the whole decode and encode operation from the beginning to the end, which matches our expectation. Comparing with software implementation, concurrent execution is more efficient than serial execution.

3. CONCLUSION

In this paper, we proposed BLINK, a novel scheme that secures the information to the last connection. Blink documents are encrypted bitmap images, which can be identified and scrambled by a dedicated device on the display link. We implemented a prototype, an embedded descrambling device working inline on the DVI cable which is able to carry out stream cipher decryption and then send the decrypted pixel values back to the display. By using this system, decrypted message would ever exist neither in the system memory nor the video memory of a computer, hence it would nullify all possible software attacks. Moreover, since any computer systems using DVI cable to connect graphics card and display will follow the DVI specification, our system is independent of any software applications and operation systems.

In our ongoing efforts, we are implementing the Trivium decryption algorithm [2] on FPGA device. And system level experiment will be conducted once the function blocks are tested.

REFERENCES

[1] Digital Display Working G Digital Display Working Group. Digital Visual Interface DVI - Revision 1.0, 1999.

[2] C. De Canniere and B. Preneel, "Trivium Specifications", eSTREAM, ECRYPT Stream Cipher Project, http://cr.yp.to/streamciphers/trivium/desc.pdf, 2006