Cryptography in Hardware

Terence O'Brien, Ian Perry

***Abstract*—Since the beginning of recorded history, the act of procuring accurate information and utilizing it has marked the difference between those organizations which fail, and those which succeed. Nowhere is this more evident than in adverserial situations. The side which is able to learn the disposition of an enemy force while concealing their own is at a great advantage [1]. As history has progressed, the simple act of hiding information has ceased to be an effective option, and methods of obfuscating, or encrypting, that information has given birth to the field of cryptography. In the modern age, cryptography is used daily by the majority of the population through the usage of ecommerce, email, or any number of secured web sites. This ubiquitous usage has created a situation where quickly processing many crypotraphic requests at once is necessary, but by its nature, these requests are computationally expensive. This is where hardware extensions which implement cryptography come into play. This paper will explore the current state of cryptographic hardware, it's role in modern usage, and the required background information to understand it.**

***Keywords*—IEEE, Cryptography, Information Theory.**

I. Introduction

THIS is the intro blahblah blahblahblahblah blahblahblah-blah blahblah blahblah blahblahblahblah blahblahblah-blah blahblahblahblah blahblahblahblah blahblah

II. History of Cryptography

III. Cryptography Basics

A review of the basics of cryptography as a whole is beyond the scope of this paper. To do that topic justice would require an entire paper in itself. Thus, the basics of modern cryptography as it applies to an understanding of its implementation in hardware will be presented.

A. Symmetric Key Encryption

Modern cryptography hinges on the use of four main functions, symetric encryption, asymetric encryption, hashing, and random number generation. Symetric encryption is method most often thought of when encryption is mentioned. It is the use of a single key to both encrypt and decrypt a message. Both user's must have the same key, and it must be shared in some fashion. This has classically been a limiting factor in the adoption and usage of cryptography.

In order to ensure a key as not been lost to the adversary, key sets must be periodically changed, which entails distributing entire new sets of keys, or having large tomes with predetermined keys available. This is an enormous burden which grows roughly on the order of n^2, where n is the number of users. Not only is this cost prohibitive at a certain n, but the logistics of securely transfering key material precludes it's use, particularly in the case of warfare or espionage.

Nevertheless, symetric key encryption has the advantage of being relatively less computationally expensive than its assymetric counterpart. This results in many uses where the initial secure session is created through the use of assymetric methods, while the bulk of the data remains secure through symmetric encryption.

B. Asymetric Key Encryption

Advancements in the field brought about the creation of asymetric key encryption. 1

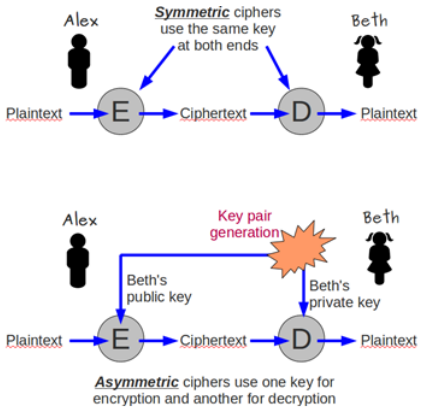

Fig. 1: Encryption Key Type Breakdown [2]

C. Hashing Functions

Hashing functions are ubiquitous in the world of computing. Whether they are used in common data structures such as a hash map, caching, computer graphics [3], or cryptography, hash functions are used in many domains for many functions.

A hash function takes a variable length input and transforms it into a fixed length output. This transformation is one way, and must not be able to be reversed. Three primary uses for hashing are found in cryptography, they are used in password checking, message authentication codes [4], and file content digesting.

When a user logs into a domain workstation, the password is not sent over the wire for authentication. Instead, it is hashed, and that hash is compared with the stored hash. This prevents a plaint text, or even an encrypted, password from being

M. Shell was with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA e-mail: (see http://www.michaelshell.org/contact.html).

J. Doe and J. Doe are with Anonymous University.

Manuscript received April 19, 2005; revised August 26, 2015.

sent, and possibly being intercepted. Furher, when storing passwords, the accepted method of doing so is by hashing them, as well as salting them.

In assymetric encryption, hashing is used to check the integrity of messages, the so called authentication code. By performing a hash of data to be encrypted, and then encrypting that hash with a private key, the recipient can decrypt the hash, verify that the data matches the hash, and then know that the user who created the hash truly sent the message. A similar idea is used in file content digesting. By storing the hash of a file in a public place, any user which downloads that file can perform a hash on their machine, compare it to the public hash, and ensure that the software they have downloaded is not a fake copy or has been altered in some way.

There are many hashing algorithms, but the most popular are MD5, SHA1, and SHA2, which is broken up into SHA-256 and SHA-512. In recent times, MD5 has become less secure [6] [5], and SHA1 is beginning to succumb to Moore's law as well. In the very near future, SHA2 varieties will make up the bulk of cryptographic hashing.

D. Random Number Generation

One integral challenge a computer system has to achieve when doing anything with cryptography is creating truly random data. This is extremely important, because as discussed above, cryptographic systems have been defeated simply because the random generator used to create a key wasnt random enough. This is a difficult challenge due to the nature of digital machines always being in well-defined, very predictable states, only changing when programs tell it to. The best that machines like this can do is simulate randomness through algorithms that create pseudorandom numbers following mathematical procedures. Such a set of data would look very random, but another computer following the same procedure could create the exact same sequence. These pseudorandom numbers usually start off with a special seed value otherwise theyd always generate the same numbers.

Luckily, digital machines can look outward to the vast, chaotic universe around them for pure randomness. One such tool that took advantage of such chaos was called Lavarand and consisted of a lava lamp with a camera. The camera would take pictures of the lava lamps seemingly random blob-like movements and through computer visions algorithms output random numbers. Such a device could then just be used to create a random seed for a pseudorandom number generator that could generate random numbers at a much higher frequency. This example shows that randomness exists in nature and can be used for digital purposes, but is obviously limited.

E. Cryptographic Protocols

Cryptographic protocols are security minded protocols which wrap the various cryptographic algorithms into cohesive blocks which perform all of the steps of initializing a session, encrypting, and decrypting. Usability is of paramount importance and has been the driving factor for adoption of cryptography by common users. Although there are many such protocols, for the purposes of this paper, onlyy SSL/TLS and HTTPS will be discussed, as they encompass the bulk of the need for hardware acceleration.

SSL/TLS, or Secure Sockets Layer/Transport Layer Security, often just called SSL, is a protocol which provides secure communications for VoIP, IM and web browsers. When using a web browser to connect to a server over HTTPS, it is nothing more than an HTTP session over an SSL channel [7]. One of the key elements of this protocol is the SSL certificate. Certificate Authorities (CA) sign certificates for domains and store their information. When a server provides a certificate, it can be vetted against the information in the CA to ensure that the server is who it claims to be. A great deal of trust and security is implicit in the operation of a CA, and thus has been theorized as a potential weak point by public key critics. To date no serious breaches of a CA's operation have occured.

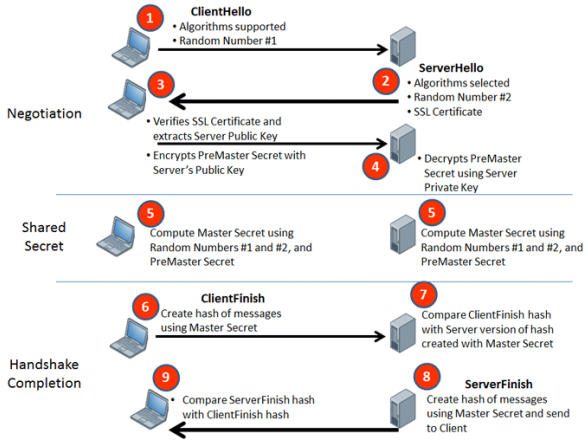

Fig. 2: SSL Process Diagram [9]

This handshake, as the initial authentication is called, also contains information regarding the remainder of the session. Included are details such as the TLS version, and which cipher suite will be used. Available suites include AES, Camellia, and SEED [8]. Available protocols for handshakes include RSA, PSK, various forms of Diffie-Helman, and others. Public-key infrastructure, which implies assymetric keys, is used for this handshake. Further, this key protects the transmission of the Master Secret, typically an AES key, which is then in turn used to protect the actual data. This process of assymetric keys allowing the transmission of symetric keys is the realization of overcoming the obstacle of symetric keys. With a method of authenticating user identity, and then passing the encryption key, all of the logistical cost discussed earlier is negated. Further, the computational cost of encrypting via AES and symetric keys is must lower than that of performing assymetric operations. The Figure 2 demonstrates the operation of SSL in a more detailed manner.

IV. Standalone Cryptographic Hardware

In the course of using an ecommerce web page, a user may only require a single SSL session every few minutes.

This level of computation is quite low, and does not require much processing power. Contrast this with the number of SSL connections per second that even a mid or low activity server require to service all incoming connections, and it is clear that hardware acceleration is required to alleviate the burden. To this effect the past two decades have seen a niche market arise to provide cryptographic accelerators, as well as entire cryptographic hardware suites which take care of key management as well as algorithmic heavy lifting.

A. Secure Cryptoprocessors

Secure Cryptoprocessors form the heart of many security based devices. The Primary focus of these processors is not acceleration or performance, but of performing cryptographic operations in a single tamper proof location [10]. When utilizing encryption in any form, it is common for one end of the session, if not both, to be in a non-secure location. An example of this could be as mundane as a desktop sitting in an internet cafe. Without explicit guardianship of the physical device, it is conceivable that an adversary would be able to gain access to the hardware in order to either extract key information, or modify it in such a way as to produce the information at a later time. If this key material were to be lost to the adversary, impersonation of the user, and the decrypting of communications for both sender and receiver could occur. There is no software or algorithm that could prevent this from occuring, as is encapsulated in the The purpose of a secure cryptoprocessor is to alleviate this oncern.

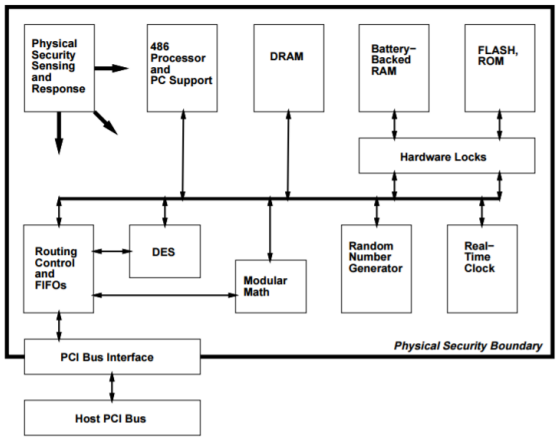

Fig. 3: Secure Cryptoprocessor Diagram [10]

In Figure 3 a secure cryptoprocessor block diagram can be seen. Although this particular processor is from a theoretical white paper, it will serve to demonstrate operations. As can be expected, scant details are the norm regarding anything to do with security, especially so with cryptography. At the peripherary of the block diagram is the physical security boundary. This construct is common to all cryptoprocessors and is the demarcation point where no unencrypted information is allowd to pass. All keys, all processing, must occur and be stored behind that boundary, unless it has been encrypted. This boundary is not only figurative, it is a literal physical boundary. The processor is enmeshed in a conductive grid [10] which allows any physical tampering to be detected. In the event that tampering is detected, the key material, at the very least, is deleted by zeroizing the memory. Ideally the operating system is also zeroized along with any other information. In this way, it can be assured that key material on the chip stays on the chip, and can not be removed, at least in a physical manner.

B. SSL Acceleration

C. Hardware Security Modules

V. Instruction Set Extensions

VI. Conclusion

The conclusion goes here.

References

[1] S. Tzu, "A quote from the art of war," in Good Reads, Goodreads, 2015. [Online]. Available: http://www.goodreads.com/quotes/744436-conceal-your-dispositions-and-your-condition-will-remain-secret-which. Accessed: Dec. 13, 2016.

[2] ict@innovation, "File: Ict-innovation-LPI-Fig-110-3 1.png - Wikimedia commons," in Wikipedia, 2012. [Online]. Available: https://commons.wikimedia.org/wiki/File:Ict-innovation-LPI-Fig-110-31.png. Accessed: Dec. 13, 2016.

[3] G. J. van den Braak, J. Gmez-Luna, J. M. Gonzlez-Linares, H. Corporaal and N. Guil, "Configurable XOR Hash Functions for Banked Scratchpad Memories in GPUs," in IEEE Transactions on Computers, vol. 65, no. 7, pp. 2045-2058, July 1 2016. doi: 10.1109/TC.2015.2479595

[4] P. Gutmann, D. Naccache and C. C. Palmer, "When hashes collide [applied cryptography]," in IEEE Security & Privacy, vol. 3, no. 3, pp. 68-71, May-June 2005. doi: 10.1109/MSP.2005.84

[5] J. Anish Dev, "Usage of botnets for high speed MD5 hash cracking," Third International Conference on Innovative Computing Technology (INTECH 2013), London, 2013, pp. 314-320. doi: 10.1109/INTECH.2013.6653658

[6] H. Kumar et al., "Rainbow table to crack password using MD5 hashing algorithm," 2013 IEEE Conference on Information & Communication Technologies, JeJu Island, 2013, pp. 433-439. doi: 10.1109/CICT.2013.6558135

[7] 2016 S. Corporation, "SSL by Symantec - learn how SSL works," in Symantec, 1995. [Online]. Available: https://www.symantec.com/content/en/us/enterprise/white_papers/b-beginners-guide-to-ssl-certificates_WP.pdf. Accessed: Dec. 14, 2016.

[8] J. Salowey and A. Choudhury, AES Galois Counter Mode (GCM) Cipher Suites for TLS, RFC 5288, Aug. 2008.

[9] "What is SSL?," in IdenTrustSSL. [Online]. Available: https://www.identrustssl.com/images/learn_ssl_diagram.gif. Accessed: Dec. 14, 2016.

[10] Building a high-performance, programmable secure coprocessor, Comput. Netw., vol. 31, no. 9, pp. 831860, Apr. 1999. [Online]. Available: http://dl.acm.org/citation.cfm?id=324119.324128