Public Key Exchange scheme that is Addressable (PKA)

Bilal Habib, Bertrand Cambou, Duane Booher, Christopher Philabaum
School of Informatics, Computing and Cyber Systems
Northern Arizona University
Flagstaff, AZ
{bilal.habib, bertrand.cambou, duane.booher, cp723}@nau.edu

***Abstract*—The objective of the PKA encryption scheme is to complement, or replace, existing Public Key Infrastructures (PKI) [1]. In this scheme, the initialization step is based on the secure exchange of addressable cryptographic tables between the communicating parties. These tables are generated either with random numbers, or with arrays of addressable Physical Unclonable Function (PUFs). The subsequent communications between the parties can therefore occurs over untrusted channels, by exchanging dynamically generated public keys. Private keys are, generated independently with all communicating parties using their cryptographic tables, and the shared public keys. The private keys are combined with methods such as the Advanced Encryption Standard (AES) to encrypt and decrypt the communication between users. The generation of private keys is done without mathematical computations that are potentially vulnerable to quantum computers using algorithms such as the one developed by Shor [2]. PKA is fast and requires approximately 800 CPU clock cycles. We implemented, and tested the PKA dynamic key exchange scheme in legacy systems to secure PC-to-PC communication, and PC to smart card communication with AES.**

I. INTRODUCTION AND PRIOR WORK

Typically the exchange of cryptographic keys between two parties to enable secure communication over untrusted channels is not an easy scheme. In public key infrastructure (PKI), each entity is equipped with two keys; the public key that is freely exchanged and the private key that needs to be kept secret. For example, Alice can generate a random session key K and then encrypts it with the Public key of Bob. Only Bob can decrypt it using his Private key to recover K. Once both entities have the same session key K, then they can transmit messages that are encrypted/decrypted using K. After the key exchange, asymmetric encryption (e.g. RSA) or symmetric encryption (e.g. AES) can be used. In commercially available PKI schemes, there are mathematical relationships between public and private keys that usually require enormous processing power to independently uncover the private keys, thereby preventing third party cryptanalyst to break the encryption methods.

Diffie-Hellman proposed the first key-exchange protocol [3]. The earliest practical implementation of it was reported in [4].

It needs to be mentioned that the discrete logarithm problem, and the integer factorization problems, are vulnerable to quantum computer attacks. In the case of RSA, they can factorize the prime numbers and determine the Private keys. If Private keys are discovered by the attacker, then it can recover the session keys previously used. To overcome this problem, one time pad schemes with random numbers have been proposed, however the problem of secret key distribution remains.

We are proposing a PKI scheme with addressable elements (PKA), in which the public keys are similar to one-time pads, and the private keys are generated without any mathematical relationship between Public and Private keys. The main motivation of our approach is to remove the mathematical relationship between public and private keys, thus making our system impregnable to Quantum computing attack.

II. METHODOLOGY

A. PERSONALIZATION STEP

The initial step of the PKA, also called personalization, is based on the generation of tables with random keys by the secure Server. It generates one cryptographic table per client device using true random numbers generator. As an example, each table can have the format of 256 rows, and 256 columns, for a total bit density of 64Kbits. During personalization, the cryptographic tables are securely downloaded to the connected devices, which could be terminal devices or secure microcontrollers. These devices have secure non-volatile memories embedded in them to store the cryptographic tables. This personalization step has to be done once for each connected device. The address to the crypto table is based on a random number ($\mathbf{T_{i,j1}}$). This number is the public key. Similarly the key read from the crypto table after a hash function becomes the private key as shown in figure 1.

B. SIMPLER SCHEME FOR KEY GENERATION

In its simpler form, we are referring to the ***public key*** of the PKA as the information needed to generate a particular ***private key*** from the cryptographic table, such as an address within the table. With the publicly communicated key, only the server and the client device can independently generate

978-1-5386-0683-4/17/$31.00 ©2017 IEEE

the data stream that will become the private key. Any symmetrical encryption schemes such as AES, is used with the private key to encrypt and decrypt messages, and perform authentication cycles, see Fig. 1. For example, a private key can be the 256 bits that are located in the cryptographic table following the particular address pointed by the public key. The table can be treated circularly, so that any remaining bits after the end of the table can be extracted from the beginning. Only the server, and the client with the appropriate cryptographic table can generate the same private key for the PKA protocol.

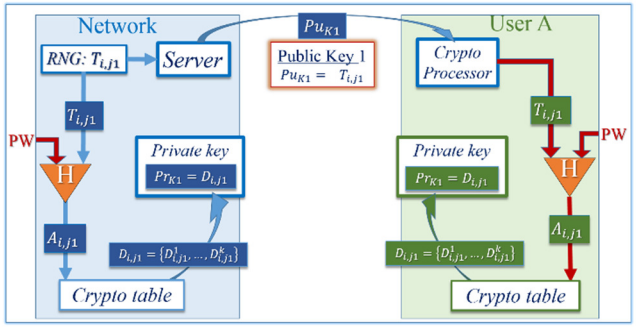

Fig.1: The **PKA** scheme is based on cryptographic tables. The hash function convert the random number $\mathbf{T_{i,j1}}$ (the public key) and **PW** into the address $\mathbf{A_{i,j1}}$ allowing the generation of the data stream $\mathbf{D_{i,j1}}$ (the private key). **PW** is the password entered by the user using a keypad.

RSA		ECC finite field (p)		ECC binary field (2n)		PKA	
Key Size (bits)	**CPU Clock Ticks**	*ECC Curve (bits)*	**CPU Clock Ticks**	*ECC Curve (bits)*	**CPU Clock Ticks**	*Key Size (bits)*	**CPU Clock Ticks**
1024	**28,304**	*secp160r1*	**470**	*sect163r1*	**2,811**	*160*	
2048	**92,625**	*secp224r1*	**688**	*sect233r1*	**3,797**	*224*	
3072	**494,283**	*secp256r1*	**803**	*sect283r1*	**6,066**	*256*	**809**
7680	**8,331,548**	*secp384r1*	**1,649**	*sect409r1*	**12,724**	*384*	
15360	**89,128,598**	*secp521r1*	**4,908**	*sect571r1*	**608,943**	*521*	

Table.1: Comparison of PKA key generation to the RSA and ECC key generation

The benchmarking of CPU ticks to generate keys is shown in Table 1. All results shown in the table were run on a MacBook Pro 3.1 GHz Intel® Core TM i7. As shown above in the Table 1, our method of PKA takes only 800 CPU clocks for 256 key size. It is significantly less than the equivalent figures for RSA and comparable to ECC schemes for similar key strength.

III. SUMMARY AND FUTURE WORK

In this paper, we presented the PKA, a public key scheme based on addressable cryptographic tables. We also extended the concept by using native ternary memories to increase entropy, as well as the use of addressable PUF arrays that can prevent the loss of the cryptographic table through side channel analysis. This method can overcome several problems associated with PKI like, RSA and ECC. Unlike legacy PKI, we believe that PKA is not as vulnerable to quantum computer attacks, as there is no mathematical relationship between public and private keys. We implemented successfully PKA to secure the server to java card, and server to terminal communication. The PKA key exchange is very efficient compared to RSA and ECC.

Going forward, we want to integrate PKA with several encryption methods, add multi-factor authentication protocol, and quantify the respective value of the elements presented in this paper: hash function, masking to increase entropy, use of ternary cryptographic tables and use of PUFs.

REFERRENCES

1. C. Paar, J. Pezl; Understanding Cryptography- A text book for students and practitioners; Spinger editions, 2011;
2. P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In 35th FOCS, pages 124–134. IEEE Computer Society Press, Nov. 1994.
3. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory IT-22, 644–654 (1976).
4. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)