Key Exchange using Ternary system to Enhance Security

Sareh Assiri
School of Informatics, Computing, and Cyber Systems
Northern Arizona University
Flagstaff, AZ 86011, USA
sa2363@nau.edu

Bertrand Cambou
School of Informatics, Computing, and Cyber Systems
Northern Arizona University
Flagstaff, AZ 86011, USA
Bertrand.Cambou@nau.edu

D. Duane Booher
School of Informatics, Computing, and Cyber Systems
Northern Arizona University
Flagstaff, AZ 86011, USA
duane.booher@nau.edu

Dina Ghanai Miandoab
School of Informatics, Computing, and Cyber Systems
Northern Arizona University
Flagstaff, AZ 86011, USA
dg856@nau.edu

Mohammad Mohammadinodoushan
School of Informatics, Computing, and Cyber Systems
Northern Arizona University
Flagstaff, AZ 86011, USA
mm3845@nau.edu

***Abstract*: This paper presents how the security of key distribution protocols can be enhanced using the ternary arithmetic conversion, along with shared keys between communicating parties. It is shown that both the transmitting and the receiving party can calculate his own private key without exchanging it during the unsecured channel. Moreover, it is explained how the binary private keys can be independently generated by both communicating parties directly from the public ternary keys. The private keys are used to encrypt messages and allow secure communication through open untrusted channels. In addition, it is shown how addressable cryptographic tables technology can be exploited to design protocols that exchange safely the shared keys.**

Keywords: key exchange, ternary security, cryptographic table, public key, private key, quantum key distribution

I. INTRODUCTION: BACKGROUND INFORMATION

Although ternary computing technology has been invented 150 years ago, it was not used in many applications because of the higher efficiency of the binary computing system. However, ternary computers are one of the hot topics in computing technology these days because of their potential applications in producing a strong cybersecurity [1]. In this regard, hybrid binary/ternary computing systems may greatly strengthen cybersecurity while keeping legacy binary codes on binary engines to maintain performance [2].

In general, public and private keys are required to enable secure communication over untrusted channels. Using the public and private keys scheme is called public key infrastructure (PKI). In a PKI system, each entity has two keys one is private, which must be kept secret, and the other is public, which is freely exchanged. As soon as both sender and receiver have the same key, they can encrypt or decrypt the message using the key. Therefore, both sender and receiver have established the secure channel between each other. After that, it is possible to use asymmetric encryption (e.g., RSA) or symmetric encryption (e.g., AES).

Northern Arizona University (NAU) previously developed a public key exchange scheme that is based on cryptographic tables containing random numbers, either bits or preferably trits, which are exchanged between the communicating parties in a secure environment (Fig. 1.) [2][5]. In the present study, we focus on "ternary addressable public key infrastructure (TA-PKI)" scheme that has been previously validated [2]. The TA-PKI employs ternary table, public key generation, hash function, and multi-factor authentication. We discuss how TA-PKI can be used to generate binary private keys. By using TA-PKI, the binary private keys can be independently generated by both communicating parties directly from the ternary keys. According to the previous work of NAU on the key exchange, public keys are binary randomly generated by the transmitting parties. Both communicating parties independently compute the keys through ternary cryptography to generate binary private keys that are then used to establish a secure communication. One possible implementation of PKI assumes a one-time use of a private-public key pair that can be combined with one-time pad cryptography. In this paper, we describe how to modify this scheme and to be able to directly use public keys consisting of streams of trits, instead of streams of bits, which have the potential to be more effective in the handling of the cryptographic tables containing trits [2][5].

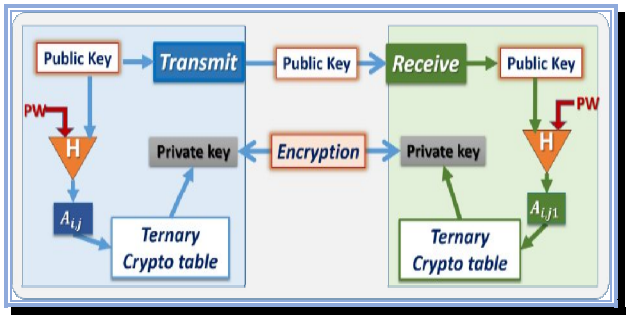

Fig. 1. Block diagram of the encryption scheme PKI with addressable elements

978-1-7281-0554-3/19/$31.00©2019 IEEE

II. CANDIDATE ALGORITHM TO IMPLEMENT A TERNARY PKI

The use of native ternary keys with native ternary algorithmic units can be benefited with the advantage of the increased entropy of ternary computing, and thus enhanced cybersecurity and information assurance between the communicating parties. Considering that the non-secure channel is used in this scheme (Fig 2), various encryption methods can be employed to protect the transfer of information such as, but not limited to, DES, Triple DES, AES, or Blowfish. A safer method in this regard is to generate very long private keys, one-time use only, and encrypt the messages with one-time pad schemes. Additional protections include multi-factor authentication of the communicating parties. Here, we will describe the implementation of a TA-PKI algorithm for the key exchange between communicating parties via unsecured channel using a ternary system.

Fig. 2. Block diagram of ternary public keys to binary private keys exchange scheme

The development of a version of TA-PKI with ternary public key needs to be developed to generate the long private key. This private key is just used one time. In addition, we use a random number generator and password as multi-factor to give more protection. Afterward, we use cryptographic tables (CT) containing large amounts of trits that are shared between the communicating parties during an initial step in a secure environment.

Now, we can describe how ternary random number is generated. A ternary random number (TRN) can be generated in several ways. It can be generated with a native ternary random number generator or by converting a random binary number [4][5].

A. Random number creation from a binary random number

The strategy followed in this paper is converting a random binary number to ternary random number. In fact, the binary states are "0" and "1" whereas the ternary states are -, 0, and +. To convert a random binary number to ternary, we need to replace the "-" in ternary form with "0" in binary, the "+" in ternary form with "1" in binary form, and present the third state "0" in ternary in a fuzzy state. Therefore, by considering Binary Boolean logic as a subset of ternary Boolean logic, we can get a stream of trits (ST) from the stream of bits [2]. The way to generate a ST from a random stream of bits is done as follows:

Every two bits from the binary random stream are combined together which we refer to as "cell". We will get four different states of the cell: 00, 01, 10, and 11. We clean up the data from "11" state. We consider that "00 is -", "01 is +", and "10 is 0". For example, if we have this stream of bits "0110010000111000001". The first step is to combine every two bits together. As a result, the stream becomes like this "01 10 01 00 00 11 10 00 01". After that, we will check every cell and see what the result is. The result of "11" will be ignored but the results of "00", "01", and "10" will be kept. Therefore, the stream that we obtain after cleaning the data is "01 10 01 00 00 10 00 01". After getting the clean stream, we assign "00" to be "-", "01" to be "+", and "10" to "0". The result is shown in Table I.

TABLE I. BINARY VERSUS TERNARY STATES

01	10	01	00	00	10	00	01
+	0	+	-	-	0	-	+

At the end, we got this ST (+0 + - -0- +). After describing how ternary random number is generated, now we explain how to use cryptographic tables (CT) in the next section.

B. Cryptographic Tables

The CT contains large amounts of trits. Primary CTs contained bits but, for the purpose of the protocol, we need trits. In order to convert the stream of the bit to trits, we follow the same procedure we did in part A. The CT must be shared between the communicating parties during an initial step. Initial step must be done in a secure environment. In this scheme, we suppose that we have a client and a server such that each of them should install the CT on their system (Fig. 3).

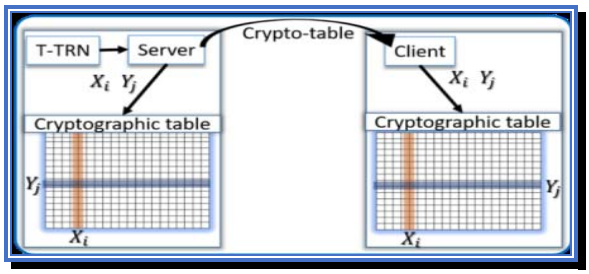

Fig. 3. Crypto Table [2].

C. Key generation steps: transmitting and receiving parties

There are two parts that need to generate the keys; i.e., the transmitting and receiving the party. Each party has own steps to generate its own keys. The block diagram in Fig. 4 shows the ternary key exchange algorithm by transmitting party. The figure presents how the private and public keys are generated. The algorithm has nine steps for the transmitting. On the other hand, the receiving party has the same steps (Fig. 5.). However, the receiving party has slightly different steps compared to the transmitting party, which is described in the following sections.

D. Steps of the algorithm for TA-PKI for transmitting party

The block diagram describing a possible algorithm is shown in Fig. 4. In general, the steps for the algorithm are summarized as follows:

- Step 1: Generate a ternary random number (TRN).

- Step 2: Create a ternary password (TPW) combined with TRN using mode 3 to generate Ternary Data Stream (TDS).
- Step 3: Convert the ternary data stream (TDS) to the binary data stream (BDS).
- Step 4: Use the BDS as input to SHA 3 hash function to produce the message digest (MD).
- Step 5: Use the MD as address location in the cryptographic table (TC) to extract a stream of ternary trits (STT).
- Step 6: Use the streams of trits (STT) to create the internal mask (IM) and private key.
- Step 7: The XORing the IM and MD to obtain the binary public mask (BPM).
- Step 8: Convert the binary public Mask (BPM) to the ternary system to obtain ternary instruction number (TIN).
- Step 9: To save the TIN and TRN in one file, which is called "the public key".

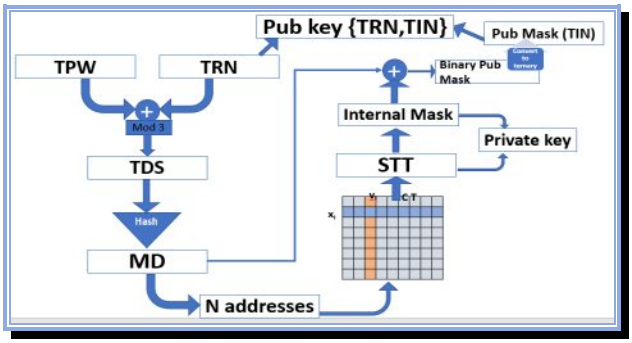

Fig. 4. The public and private key pair generation by the transmitting party.

E. Steps of the algorithm for TA-PKI for Receiving party

The receiving party has slightly different steps compared to the transmitting party (Fig. 5). These steps are as follows:

- Step 1: The public key including TRN and TIN is sent through the channel to the receiver.
- Step 2: The receiver will take TRN and TPW to do the same steps from 2 to 6 as the transmitting party did.
- Step 3: The receiver takes TIN and converts it to a binary stream.
- Step 4: The binary stream that has been extracted from TIN is XORed with MD calculated from step 2 to extract the IM.
- Step 5: The IM and STT can calculate the private key.

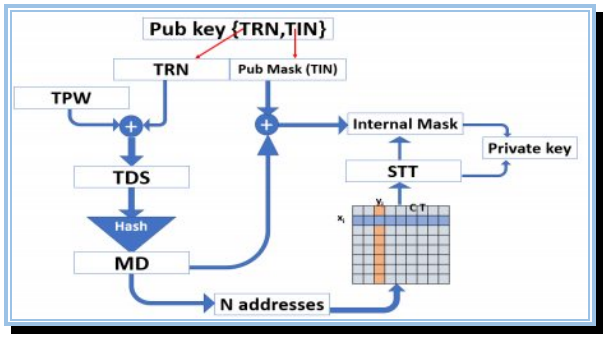

Fig. 5. The public and private key pair generation by the receiving party

III. IMPLEMENTATION

Each step has own method for implementation. In this section, we elaborate all the implementation steps for both parties.

- Steps 1 and 2: Ternary Random Number (TRN) and Ternary password (TPW)

First of all, we need a TPW and TRN. The password is created just once by the transmitting and receiving. In comparison, the ternary random number is generated every time we want to make new long private key between parties (Fig. 4). In this way, the TPW and TRN are obtained by converting the binary stream to ternary stream. First, every two bits of the binary stream are considered as one cell. Essentially, each cell should have four different status: '00', '10', '01', and '11'. We will go through the stream of bits and change each cell from {00 to 0, 01 to 1, 10 to x, and 11 to u}. After that, the cells that have the value 11 must be removed from the stream. The new stream that we will get will be stripped from the status of '11'. Now, according to 0 = 00, 1 = 01, x = 10, we will go through each cell and classify them to obtain the trit stream values of (0, 1, x) or (-,+,0). After converting the binary stream to ternary, we add mode 3 ((TPW +TRN) %3) between both TPW and TRN (Fig. 4). The addition between TPW and TRN with mode 3 generates one ST that we referred to as Ternary data stream (SDT) (Fig. 4.).

- Step 3: Ternary Data Stream (TDS)

In this step, TDS is converted into a binary data stream and then is converted into a binary MD using one of the hash functions. In this project, we use in SHA3, which will be MD 512 bits long (Fig 6).

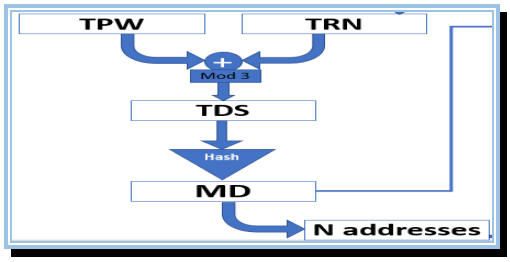

Fig. 6. TDS to MD 512 bits

- Steps 4 and 5: The MD to extract a ST

By the MD, we tend to extract the stream of ternary trits (STT) located in the cryptographic table (TC). The TC includes random trits that have been shared by both communicating parties. The MD will extract 32 different addresses; for example, if the table contains 256x265 trits, then the 512-bit long MD can be applied to find 32 different addresses, each with 16-bit vector (8 bits for the x coordinate of the table and 8 bits for the y coordinate)(Fig. 7).

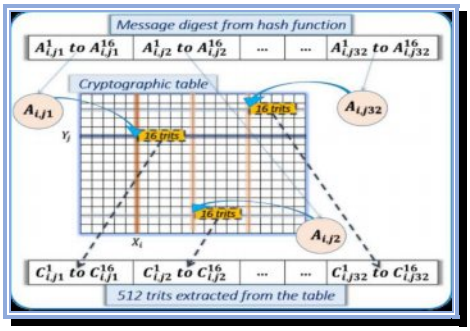

Fig. 7. different addresses from the hash digest to extract (STT) 512 trits [2]

- Step 6: STT, an IM, and a private key

This step describes how to create an IM and a private key. The following steps illustrate how the IM is created. Form 512 trits (STT) obtained from the CT, we perform the following stages to create the IM:

1. Generate random array for the mask. Initialize mask array, where the positions of the trits 'x' value have a corresponding '0' value in mask array.
2. Create an internal array of the mask '0' bits that have no corresponding data for 'x' values.
3. Create an internal array called zeros, where each element contains an index into the trit row of all of the '0' bits (Fig 8). It shows that we would have values of 0, 3, 4, 7, and 10 in the zeros array.
4. Count the mask '1' bites that have no corresponding data 'x' values.
5. Repeat these steps until mask '1' bits are 256.
6. Randomly select bits from step #3 (Fig. 9).

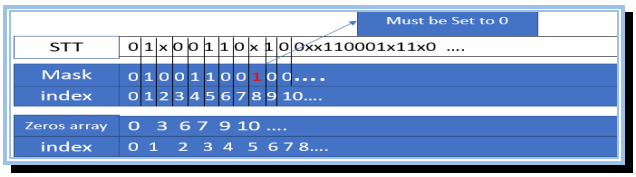

Fig. 8. How to create an array that saves the index of '0' STT

After the IM is created, then we can create the private key by comparing the bits values of TSS and mask. If the bit value of the mask is equal to '1' and the bit value of STT is not equal to x, we accept the bit value of STT and put it in private key array (Fig. 9.).

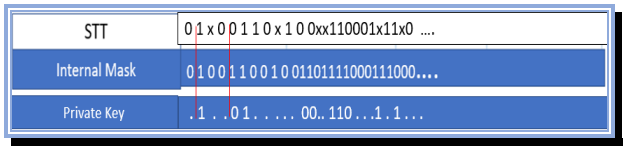

Fig. 9. Mask and private key

- Step 7: XORing the IM and MD to get the public Mask (PM)

To create the public mask (PM), we have to XOR the IM and MD. The result is a binary stream that we call it as binary public Mask (BPM) (Fig. 4.).

- Steps 8, 9, and 10: Convert the binary public Mask (BPM) to the ternary system to get Ternary Instruction Number (TIN) and create the public key.

To create a TIN, we convert the binary stream that has been gotten from (BPM) to ternary stream. The procedure applied for this purpose is slightly different from the procedure used in Steps 1 and 2. In this step, we convert each '0' bit to become '00' cell and each '1' bit to be '01' cell. For example, if the (BPM) = 110110011, after converting to ternary it becomes like 0101000101000001 01 (Fig. 4.). The obtained ternary stream becomes easy to read by the man in the middle. The man in the middle can recognize easily the '0' that is added to create the ternary stream. However, the edit distance will be used to make the ternary stream ambiguous for the man in the middle. We insert some random numbers after each byte. Later on, the receiving party takes out all the appendages of the random numbers from the stream. After accomplishing the edit distance, the new TIN is produced. TIN and Ternary Random Number (TRN) from Step 1 is stored in a file called "the public key" and then sent through a channel to the receiver (Fig. 4.).

- Steps 11 to 14: The receiver takes TRN and Ternary password (TPW) to do the same Steps 2 to 6.

By assuming the receiver already has the same TPW. Then, from the public key file, the receiver can obtain also the TRN. Now the receiver is able to repeat the same Steps 2 to 6 to gain STT. Also, the receiver takes TIN and removes the inserted edit distance bit. Now, the receiver is able to convert it to a binary stream. The extracted binary stream is XORed with MD, which is calculated from Step 5, to extract the IM. Finally, the receiver becomes able to obtain the same private key (Fig. 5.).

IV. RESULT

In this section, we show the output of all steps of a private key generation that we have coded in C++. First, we present the TRN generated from random number function. TRN is a binary random number that does not include the '11' status and only includes '00', '01', and '10'.

Public key(TRN)
:40510201A6A2506A544421546A29581A6698695A5618A25492101901294488555A88951546A410242258489866A8168501965096186042190 4AA55A09515A5662899642814 82A5AA92102062212805568482885 0A51860A9A598A00562A8A9A89480AA104820A920A0625695A5004429111481450404251111 8AA4A1A2A92962.

Fig. 10. Ternary Random Number

In the following, we present that TPW that should be entered from the client. It is a binary random number that does not include the '11' status. Rather, it only includes '00', '01', and '10'.

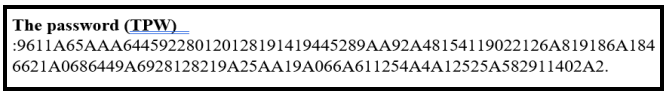

Fig. 11. Ternary Password

As can be seen in Fig. 12 upper party, the ternary state is printed using samples - for 00, + for 01, and 0 for 10. In Fig. 12 lower part, the ternary state is printed using samples 0 for 00, 1 for 01, and 2 for 10.

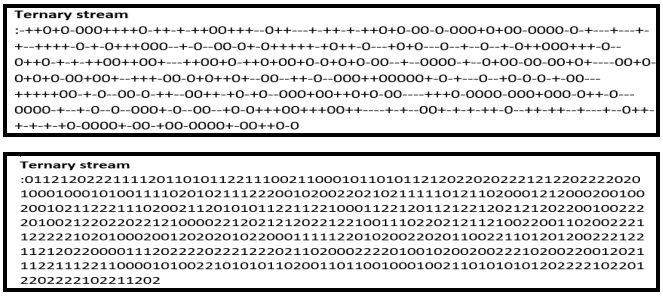

Fig. 12. Ternary State

The MD, STT, Internal mask of TT, The MD_XOR_IM and Private key of TT is shown in Fig. 10.

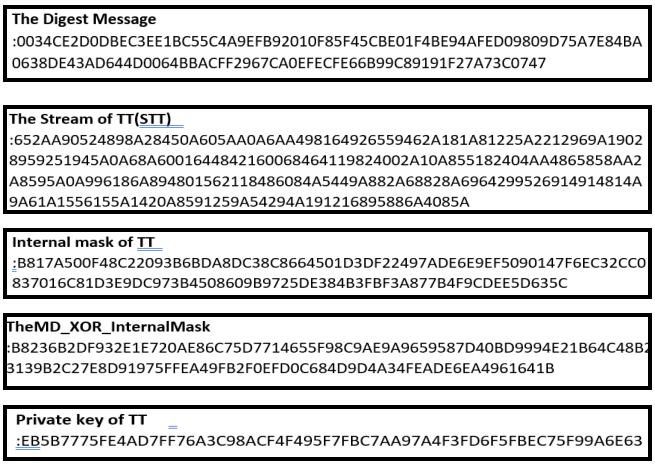

Fig. 13. The outputs of The MD, STT, Internal mask of TT, The MD_XOR_IM, and Private key

The result that we got after we have implemented the TA-PKI algorithm for the key exchange is that we got 32 bytes long private key in both communication parties that have the same CT. The private key that we have produced can use for encryption and decryption the messages between communication parties. The private key that we have obtained can use just for one time. Without having the same CT no one can calculate the same private key. As we have mentioned in section III that the CT must install in a secure environment which makes having the same CT difficult to the one who wants to have it.

V. CONCLUSION

In this paper, we have shown how the private keys can be independently generated by both communicating parties directly from the ternary keys. Using TA-PKI both parties can get access to the same private key for encryption and decryption. The private key that has been generated from the protocol is used only one time. Not exchanging private key through an unsecured channel, improves the security of the key exchange between both sides of the communication. In addition, we used multi-factor authentication to give a high level of security.

VI. FUTURE WORK

For the future work, using Ternary Addressable Public Key Infrastructure protocol, we can make Quantum Key Distribution more secure. Using this method, we can eliminate the demand for privacy amplification and enable the receiver to detect eavesdropping only by looking at its measured data. We eventually plan to replace the cryptographic table with Physically Unclonable Functions to design a protocol that exchanges shared keys safely.

REFERENCES

[1] B Cambou, P Flikkema, J Palmer, D Telesca and C Philabaum " Can Ternary Computing Improve Information Assurance? " MDPI cryptography 18 January 2018; Accepted: 27 February 2018; Published: 2 March 2018.

[2] B. Cambou, D.Telesca, "Ternary Computing to Strengthen CybersecurityDevelopment of Ternary State-based Public Key Exchange". Computing Conference | London, UK July 2018.

[3] D. Booher, B. Cambou; Generation of Composite Private Keys; NAU disclosure D2018-038; April 2018

[4] B. Cambou, "Quantum Ternary Key Distribution Schemes". December 2017.

[5] B. Habib, B. Cambou, D. Booher, C. Philabaum, "Public key exchange scheme that is addressable", CNS 2017;

[6] B. Cambou, A XOR data compiler combined with physical unclonable function for true random number generation; SAI/IEEE computing conference, July 2017;

[7] B. Cambou; "Multi-factor authentication using a combined secure pattern"; US patent 9,514,293, 2015.

[8] Peter W. Shor, and John Preskill; "Simple Proof of Security of the BB84 Quantum Key Distribution Protocol"; volume 85, No2, Physical Review Letters, July 2000;

[9] Abdulbast Abushgra, and Khaled Elleithy; "Security of Quantum Key Distribution"; 2015 ASEE Northeast Section Conference;

[10] Charles H. Bennett, and Gilles Brassard; "Quantum cryptography: Public key distribution and coin tossing"; Theoretical Computer Science 560 (2014) 7–11;

[11] Glenn Wellbrock, Tiejun Xia, David Chen; "quantum Key Distribution System", US patent application US 2007/0076884 A1;

[12] Sheila Cobourne; "Quantum Key Distribution –Protocols and Applications"; Technical Report RHUL-MA-2011-05 March 2011, University of London.

[13] M. Khalid, and J. Singh; Memristor based unbalanced ternary logic gates; Analog Integrated Circuits and Signal Processing, April 2016.

[14] B. Cambou; Data compiler for True Random Number Generation and Related Methods; NAU disclosure D2017-03, Aug 2016.

[15] B. Cambou; Encryption Schemes with Addressable Elements; NAU disclosure D2017-21, Jan 2.

[16] D. Booher, "NAU PKA ternary Table Host-Side implementation". April 26, 2017.