Providing a Distance Bounding Protocol Named Pasargad in order to Defend against Relay Attacks on RFID-Based Electronic Voting System

Mohammad Arjmand[1], Mahmoud Gardeshi[2], Reza Taheri zohur[3] and Mohammad Kazemi[4]

[1]Department of Information Technology and Communication, IHU University, Tehran, Iran
arjmand88@yahoo.com
[2]Department of Information Technology and Communication, IHU University, Tehran, Iran
mgardeshi2000@yahoo.com
[3]Department of Information Technology and Communication, Sharif University, Tehran, Iran
reza@taherizohur.com
[4]Department of Information Technology and Communication, Sharif University, Tehran, Iran
mk8008mk@gmail.com

ABSTRACT

The most important characteristic of RFID-based electronic voting system compared to traditional voting system is that votes in the electronic system are as contactless smart cards in place of paper ballots. For casting ballots, voters use a computer terminal to write their choices (their chosen candidates) into contactless smart cards and then put the smart card inside the box. The most important threat for RFIDsystems is information robbery and relay attacks. In this article, by designing a protocol called Distance Bounding Protocol it is tried to defend these systems against relay attacks.

KEYWORDS

RFID, data-reader, tag, relay attack, security, electronic voting, smart card, distance bounding, Pasargad Protocol, Pars Protocol, Huffman Algorithm

1. INTRODUCTION

RFID is an identification system using radio waves that since 1940 has existed. Most important components of a RFID system are Tag, Reader and Verifier. Tags in terms of type are divided into three main groups of Active, Inactive and Semi-active. Active tags possess internal electricity feeding source and their reading range is by far larger than inactive tags. Inactive tags do not possess internal electricity feeding source; In fact, these tags receive their energy from a current which is transmitted from reader signals as a result of which reading range of these tags is shorter than active tags. Semi-active tags use their internal battery to respond to readers and supply of the energy required by memory. Use of battery enlarges tag's reading range [1], [4]. Since invention of this technology in 1939, RFID application was extended to fields. On July 2004, use of RFID system for identification of patients in hospital or employee's access to patient files was

DOI:10.5121/iju.2011.2305

investigated. Afterwards, American hospitals began to implant RFID systems in patients for the purpose of better management [5].

2. ELECTRONIC VOTING PLAN

The most important characteristic of electronic voting program based on RFID compared to traditional voting system is that ballots in electronic voting system are cast using contactless smart cards in place of paper ballots. For casting ballots, voters use a computer terminal to write their choices (their chosen candidates) into contactless smart cards and then cast the smart card inside ballot box. Details of this voting method are as follows.

2.1. Plan's component

Components of voting station are shown in figure (1). Each voting station is comprised of the following elements:

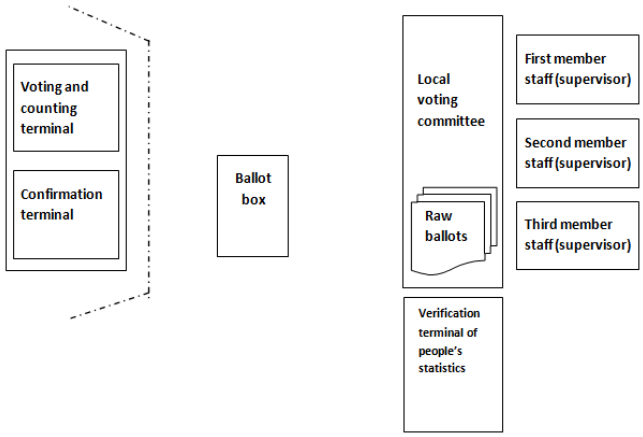

Figure 1.Components of voting station

- Voting terminal: voting terminal is contact display screen with a reader of contactless smart card. Voters use this terminal for casting their ballot. Each vote is registered two times: first time, the vote is written on blank paper (i.e. RFID contactless smart card) and for the second time, after final confirmation of voter, these votes are written in data bank[2].

- Confirmation terminal: confirmation terminal is a monitor with a reader of contactless smart card. This terminal is only able to read votes. Voter can optionally put one's written vote in this terminal in order to ensure that one's vote has been properly given.

- Contactless smart cards: contactless smart cards are the very ballot papers and voter is allowed to vote by this card and voter's information until end of voting is stored in this card.

- Voting booth: voting booth is small cabin which hides voting terminal and confirmation from sight of electoral committee and allows voters to choose their vote confidentially.

- Ballots box: ballots box is a box in which votes (contactless smart cards) are cast and are collected physically.

- Local Voting Committee: this committee includes three neutral persons who send voting report to the central electoral committee that supervises over voting operation.

- Verification terminal of people's statistics: local voting committee using this terminal examines eligibility of voter and voting qualification of voter (checking if the person has not already voted).

2.2. Electronic voting process

Voting process is shown in figure (2) and executed as follows.

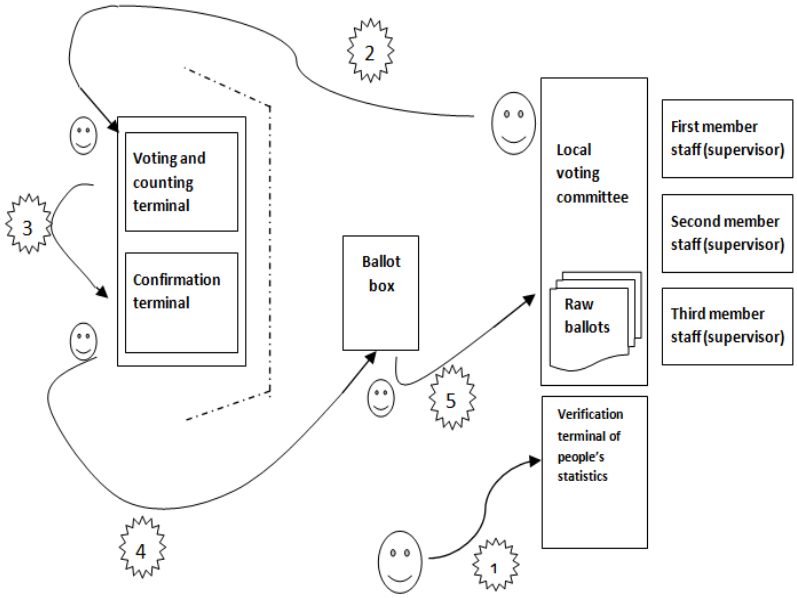

Figure 2.Voting process

- Voter comes to voting committee and this committee determines voter's voting eligibility, takes his/her ID card and gives her/him a raw ballot (contactless smart card).

- Voter enters voting booth (in this booth, where voter does is out of sight of voting committee). Next, voter puts raw ballot in the reader connected to voting terminal and chooses one's vote through a contact monitored medium[3].

- If voter wants to ensure that his/her intended vote is correctly registered on ballot, he/she may put the ballot in the reader connected to terminal in order voter's selected vote to be shown by the terminal. Confirmation terminal is only to read the votes that are cast in this particular station[3]. Ballot should be quite secure in terms of coding and its content should not be changeable or readable by any means. Security of this stage is provided using Pasargad Protocol.

- Now, voter is ensured of one's vote and then his/her vote is registered in the system's database and casts one's contactless smart card under eyes of voting committee.

- After confirmation of voting stages correctness, voting committee return voter's ID cards to them.

- In the end of the day, voting committees recollect contact smart cards from voting terminal and deliver them to the central voting committee in order the cards collected from all the regions to be counted and preliminary results of voting to be calculated[3]. At this stage, using the reader device, votes counting capacity in batches of ten thousands votes is possible.

- Final counting which takes place by reader should match the final results of data base. In case the counted votes by reader has a difference in excess of a certain percentage from the existing votes in data base, all the existing votes in such a voting station are declared invalid.

3. USE OF DISTANCE BOUNDING PROTOCOL IN ORDER TO PREVENT RELAY ATTACKS OF ELECTRONIC VOTING DESIGN

Relay attack occurs when an valid tag is deceived by an attacker, i.e. a situation in which attacker makes contact with tag or reader, while the tag or reader thinks they are directly in contact with each other [6]. Relay attacks are divided into mafia counterfeit attacks and terroristic counterfeit attacks. Several protocols have been suggested which can prevent relay attacks ([7-11]). However, these protocols are not absolutely resistant against relay attacks.

Mafia counterfeit attack occurs when an attacker for signal relay puts an invalid tag and reader between valid reader and tag, respectively, for signal relay [12]. Brands and Chaum introduced a distance bounding protocol which fundamentally by examination of physical vicinity of a tag through a series of quick challenge-response circuits of bits exchange prevents mafia counterfeit attack [13]. Then the time of round trip between tag and reader is calculated. If distance between tag and reader is an equal to time of the round trip calculated in an acceptable ranges, the tag is supposed to be valid.

When the main (valid) tag and reader are not aware of this process, mafia counterfeit attack takes place. On the other side, terroristic counterfeit attack occurs when an invalid tag in order to deceive a reader which is positioned in its vicinity colludes with the attacker. The colluding tag can redistribute all the required information for distance bounding phase to the attacker and the attacker can successfully counterfeit the colluding tag's identity. Most of the distance bounding protocols resist against relay attacks and measure the round trip distance between tag and reader.

3.1. Use of Pasargad distance bounding protocol for preventing relay attacks

Thus, in order to prevent voter's identity falsification in the time of voting or change of voter's vote by terroristic of mafia counterfeit attack, distance bounding protocol can be used.
We have designed a protocol which minimizes attacker's success probability. In the presented protocol by Perniel and Single, attacker's success probability was ½. It means that attacker only had to guess whether the forwarded bit is zero or one [14]. But in the protocol designed by us, the forwarded bit is transformed into 16 bits. It means that attacker's success chance is equal to 2^{-16}.
In Pasargad distance bounding protocol which is shown in the figure below, two algorithms of Pars and Huffman is used. This algorithm will be fully explained in the next sections. This protocol is transformed into two phases of identity verification and distance bounding identification. First, we explain identity verification phase.

3.2. Identity verification phase

Alice and Bob agree with each other on a shared key which its name is K. Steps of identity verification as follows.

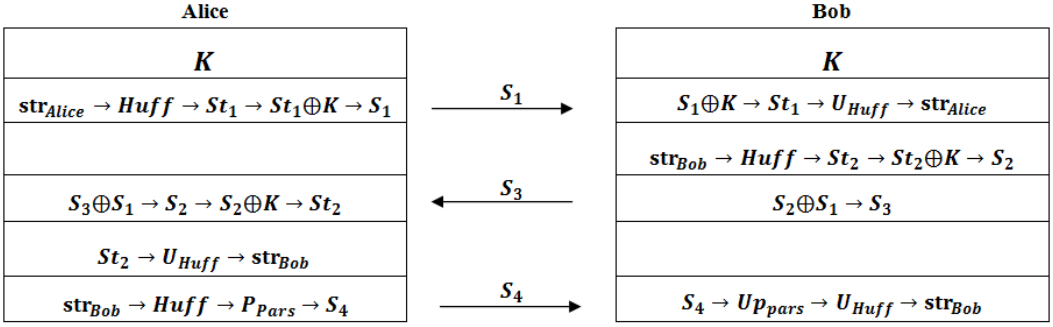

Figure 3.Identity verification phase

- Alice using Huffman algorithm to encodes her determinant code (str_{Alice}) which is a string using for createsSt_1.
- Alice XOR $St_1$with the shared key (K) and creates $S_1$and sends it to Bob.
- Bob receives $S_1$and XOR it with the shared key and obtains the St_1.
- Bob executes Huffman decoding operation on the $St_1$and from there he obtains str_{Alice}.
- Bob using Huffman algorithm to encodes her determinant code (str_{Bob}) which is a string using for creates St_2.
- Bob XOR St_2 with the shared key (K) and from there he obtains S_2.
- Bob calculates $S_2 \oplus S_1$ and from there he obtains S_3 and sends it to Alice.
- Alice calculates $S_3 \oplus S_1$ and obtains S_2.
- Alice calculates $S_2 \oplus K$ and obtains St_2.
- Alice using Huffman algorithm to decode St_2 and from there she obtains (str_{Bob}).
- Alice respectively executes Huffman and Pars operation on (str_{Bob}) and sends the S_4 as a result for Bob.
- Bob respectively executes Pars and Huffman decoding operation on S_4 and obtains (str_{Bob}).
- If identity verification phase is done successfully, distance bounding phase starts.

3.3. Distance Bounding Phase

In this phase, characters are forwarded one by one and distance bounding between Alice and Bob is specified (figure 4). Stages of this phase are as follows.

Alice		Bob
K Shared Key		K Shared Key
$b_i \rightarrow p_{pars} \rightarrow B_i$	$B_i \rightarrow$	$B_i \rightarrow up_{pars} \rightarrow b_i$
$Time\ on$		
$A_i \rightarrow up_{pars} \rightarrow a_i$	$\leftarrow A_i$	$a_i \rightarrow p_{pars} \rightarrow A_i$
$Time\ off$		
$If\ \Delta t_i \leq t_{max}$		
$protocol\ success$		

Figure 4.Distance bounding Phase

- Alice executes Pars algorithm on the i'th character of the string (str_{Bob}) which means b_i and obtains B_i.
- Alice switches on her chronometer and sends B_i which is a 16-bit string to Bob.
- Bob receives B_i and executes Pars decoding operation on B_i and obtains b_i.
- Bob executes Pars algorithm on the i'th character of the string (str_{Alice}) which means a_i and sends A_i which is a 16-bit string for Alice.
- Alice calculates the time between sending and receiving a_i and b_i, if this time is shorter or equal to protocol's allowed time, the protocol has been executed successfully and no identity falsification has taken place, otherwise the protocol has failed.

4. PARS PROTOCOL

As we know, each character in ASCII Code is equal to 8 bits. Therefore, in ASCII Code, for each bite there are $2^8 = 256$ states. Suppose, the three bites below which correspond to three words are going to be encoded using Pars Protocol. As we see in table (1), the three bites are equal to 24 bits and each bite is denoted by a symbol.

Table 1.Example for Pars Protocol

Byte 1	Byte 2	Byte 3
#######	********	$$$$$$$$

We can divide these 8-bit bites into four six-bit groups. Therefore for each group, there are $2^6 = 64$ states. Thus, table (1) can be represented as table (2).

Table 2.Process of classification in Pars Protocol

Group 1	Group 2	Group 3	Group 4
######	## ****	**** $$	$$$$$$

As was observed, there are 64 states for 6 bits. Hence, we have designated 64 characters for Pars protocol and we show it in the continuation. These 64 characters are 26 English capital letters (A-Z), 26 English lower case letters (a-z), numbers 0-9, and characters "/" and "+" which in total become 64 characters which we need. Thus, all the three bites are equal to the four characters in Pars protocol.

4.1. Encoding process in Pars Protocol

Before explaining encoding method by Pars protocol, we have to equalize the numbers 0-63 with characters of Pars protocol. This equalization is shown in table (3).

Table 3.Equivalent to making table of Base64 in Pars Protocol

Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PPars	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Value	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PPars	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f
Value	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
PPars	g	h	i	j	k	I	m	n	o	p	q	r	s	t	u	v
Value	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
PPars	w	x	y	z	0	1	2	3	4	5	6	7	8	9	+	/

For example, the value 25 (the binary equivalent of which is 011001) corresponds to the character "Z" in Pars protocol. Or the binary value 101010 (the decimal equivalent to which is 42) corresponds to the character "q". To show encoding process by Pars protocol, we begin with a simple example. Suppose we want to encode the phrase "Hello World!" using Pars protocol. First of all, we obtain the ASCII code and binary value of each character (see table 4).

Table 4. Equivalent of ASCII code Hello World

Character	H	e	l	l	o	[space]
ASCII Value	72	101	108	108	111	32
Binary Value	01001000	01100101	01101100	01101100	01101111	00100000
Character	W	o	r	l	d	!
ASCII Value	87	111	114	108	100	33
Binary Value	01010111	01101111	01110010	01101100	01100100	00100001

As was said, for transformation into Pars protocol, each time we use three bites. Each ASCII character is equal to only one bite. Hence, we divide the phrase "Hello World!" in to four 3-character groups: ('Hel", "lo[space]", "Wor", and "ld"). Encoding stages of this phrase using Pars protocol are as below:

We write binary equivalent of characters Hel: 01001000 01100101 01101100
We divide these three 8-bit bites into four 6-bit groups: 010010 000110 010101 101100
We write decimal equivalent of 6-bit groups: 010010 = 18 000110 =6 010101 = 21 101100 = 44

We obtain solution of these numbers using Pars protocol table:
44 = “s”, 21 = “v”, 6 = “G”, 18 = “S”

Thus, the three first ASCII characters (i.e. “Hel”) using Pars protocol was encoded into “SGVs”. To obtain the remaining characters, the above stages should be repeated. We have done this already and the following results have been obtained: "Hel=SGVs; "lo [space]”=bG8g; "Wor"=V29y; "ld!"= bGQh;

Thus, the phrase “Hello World” was transformed into: "SGVsbG8gV29ybGQh"
As you observed, the main phrase includes 12 ASCII characters which by Pars protocol was transformed into 16 characters. Now the question may arise that if a phrase cannot be divided into 3-characters groups, then what should be done?

For example, if a phrase which is supposed to be encoded is consisted of 5 ASCII characters (e.g. “Hello” or “blue”), how encoding should be executed? In this case, “Hello” is divided into two groups of “Hel” and “lo” and the work “blue” is divided into two groups of “Blu” and “e”. As we know, one and two-character phrases cannot be divided into 6-bit groups. Thus, the character “=” is used to complete phrases. For example, for the phrase “Hello” we act as follows (table 5). This phrase is divided into two groups of “Hel” and “lo”. The phrase “Hel” as we saw earlier using Pars protocol is transformed into “SGVs” and for the two remaining characters (i.e. “lo”) we act as below:

Table 5. Equivalent of ASCII and Binary code Hello phrase

Character	*H*	*e*	*l*	*l*	*o*
ASCII Value	72	101	108	108	111
Binary Value	01001000	01100101	01101100	01101100	01101111

- First, we write “lo” binary equivalent: “lo” = 01101100 01101111
- We start from left side and classify into 6-bit groups: 011011, 000110, 1111

As you see, the third group for being completed need two bits. In addition to these two bits, there is need for another six bite to form the fourth group in order by Pars protocol to encode it (because we have to have four 6-bit groups). Thus, we need a phrase similar to the phrase below:
011011 000110 1111** ******

In this phrase, only the first two groups can be transformed into decimal. To solve this problem, we complete the third group by adding two zeros to its end and in place of the fourth group, we use the character “=”. 011011 = 27, 000110 = 6, 111100 = 60, ****** = “=”

Now, we refer to Pars table and replace their equivalents: 27 = “b”, 6 = “g”, 60 = “8” and in place of the last phrase we put the symbol “=”. Thus, the phrase “Hello” is transformed into “SGVsbg8 =”.

4.2. Decoding process in Pars Protocol

Now we want return an encoded phrase by Pars protocol to its initial state (decoding). For this purpose, we begin with a simple example. Suppose we want to decode the string “YmFzZTY0IGLzIGZ1biEh” by means of Pars protocol. For decoding, we have to repeat encoding operation from the last stage to the first stage. For this purpose, first, we obtain Pars protocol’s equivalent of each character from Pars protocol table (see table 6)

Table 6. Decoding of phrase YmFzZTY0IGLzIGZ1biEh by Pars Protocol

Character	Y	m	F	z
PPars Value	24	38	5	51
Binary Value	011000	100110	000101	110011
Character	*Z*	*T*	*Y*	0
PPars Value	25	19	24	52
Binary Value	011001	010011	011000	110100
Character	*I*	*G*	*L*	*z*
PPars Value	8	6	37	51
Binary Value	001000	000110	100101	110011
Character	*I*	*G*	*Z*	1
PPars Value	8	6	25	53
Binary Value	001000	000110	011000	110101
Character	*b*	*i*	*E*	*h*
PPars Value	27	34	4	33
Binary Value	011011	100010	000100	100001

As was saw earlier, in the time of encoding, we used 8 bits to show two characters. But in the time of decoding, we have to divide them into 3-character groups. Thus, in this example, the string is broken as follows: “YmFz, “ZTY0”, “IGLz”, “biEh”
We explain decoding process for the first group, i.e. “YmFz” and we repeat this process for other groups.

- We write binary equivalent of Pars characters in the group “YmFz”: 011000 100110 000101 110011
- We divide this 24 bits which are comprised of four 6-bit groups into three 8-bit groups: 01100010 01100001 01110011
- We write decimal equivalent of each 8-bit group: 01100010 = 98 01100001 = 97 01110011 = 115
- We obtain ASCII code equivalent of each one of these numbers from ASCII table and write: 98 = “b” 87 = “a” 115 = “s”

Now, the four Pars characters which are equal to “YmFz” is decoded into three ASCII code of “bas”. If the same stages are repeated for the four remaining groups, the following results are obtained: “ZTY0” = “e64”, “IG1z” = “is”, “IGZ1” = “fu” and “biEh” = “n!”.

Thus, the coded phrase “YmFzZTY0IGLzIGZ1biEh” using Pars protocol is decoded into the phrase “base64 is fun!!”.

As was earlier seen, in the time of encoding, 24 characters were not completed and we had to add a number of bits in order the number of these characters to reach 24. Then we divided them into 6-bit groups. Now, if we want to bring a character which by Pars protocol is transformed into “=”, we act as follows.

We describe decoding process by an example. Suppose we have the following encoded phrase:

"Li4ub3IgbwF5YmUgb90Lg = =".

Decoding process of five groups is done like the previous method. Thus, we decode only the sixth group, i.e. "Lg = =". As you know the symbol "=" may have two meanings; first, as an equivalent to a 6-bit group one phrase of which is used for completion four groups and second, as the third group's complementary bits in divided phrase into 6-bit groups. Now, we act as follows:

- We obtain Pars equivalent of each character from Pars table: "L" = 11, "g" = 32, "=" = nothing, "=" = nothing
- We write binary equivalent of these values: "11" = 001011, "32" = 100000, "nothing" = ******, and "nothing" = ******
- We put these four 6-bit groups next to each other and divide them into three 8-bit groups: 001101110 000**** ********

Since there are two symbols in the end of the phrase, we have had two bits short in the main frame (before encoding). You remember that when we wanted to encode by Pars, we had to add a number of zeros. These zeros are the same zeros existing in the second 8-bit group (0000****), because each 8-bit group only indicates one bite from the main phrase.

Thus, the two last bites have not been in the main phrase. Hence, we discard the two 8-bit groups of "0000****" and "********". As a result, the only remaining bite is "00101110" the decimal equivalent of which is "46". Now we obtain ASCII code equivalent of "46" from ASCII table which is equal to the character ".". The remaining data are decoded as follows:

Therefore the encoding string is decoded into the phrase "… or, maybe not".

5. HUFFMAN ALGORITHM

Encoding by Huffman's method was published by David Huffman, PhD student of MIT in 1962 in his famous article named "A method for Code Production with the Least Redundancy". In this method, codes are produced with varying length. Huffman's codes have characteristics of unique prefixes as well, i.e. they can be decoded correctly. This is usually done by following a binary tree. In this method a binary tree is constructed bottom-up. The tree construction process is as follows. Symbols will be placed at the bottom level as nodes which are supposed to be made by the binary tree. Each node has its own weight which is the frequency of its repetition or possibility of its being seen.

Step-by-step process of Huffman three are as follows:

- Two nodes which have the least weight and have not yet been used.
- A father node is made for these two nodes and its weight is equal to total weight of these two offspring.
- The father node is added to the list of nodes and offspring nodes are eliminated from this list.

- One of the children optionally gets the code zero and another offspring the code 1.
- These steps will continue until when there remains only one node. This node is the tree's root and at this point the algorithm ends [15].
- Suppose we want to code the phrase "The Pasargad is good" using Huffman's algorithm. First, we obtain repetition (iteration) frequencies in this phrase (table 7) and then we draw its tree.

Table 7. Repeated frequencies of a phrase The pasargad is good

Symbol	*Repetition Frequency*	*Symbol*	*Repetition Frequency*
T	1	*r*	1
h	1	*g*	2
e	1	*d*	2
p	1	*i*	1
a	3	*o*	2
s	2	*space*	3

The binary tree of the above phrase is as figure (5).

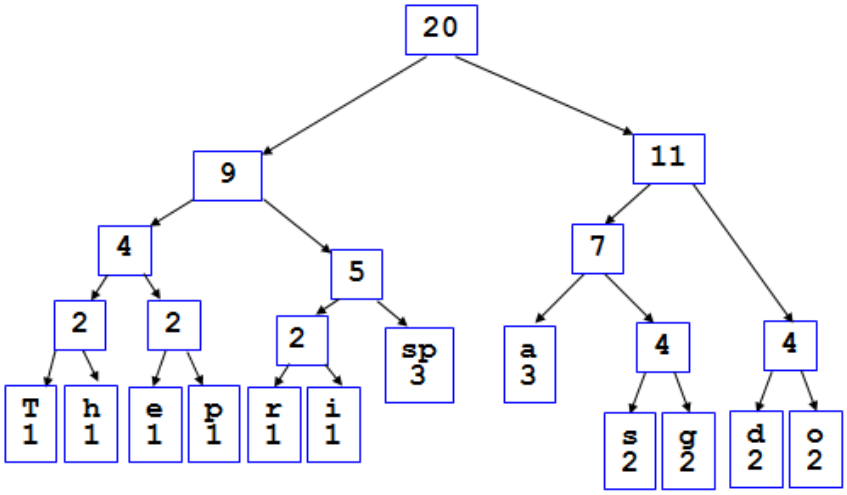

Figure 5.Binary tree of phrase The pasargad is good

Now, we dedicate digit zero to left-hand side branches and digit one to right-hand side branches. Thus, in this tree, each character is assigned a new code. The existing frequency in the root node should be equal to number of characters of the above phrase. As you observe, number of characters of the phrase "The pasargad is good" is 20 which is equal to repetition frequency of the root's node. Now, to obtain new code of each character, we move from the tree's root towards the intended character. Therefore, the above phrase is encoded as follows (table 8):

Table 8.Huffman codes characters of phrase The pasargad is good

Symbol	*T*	*h*	*e*	*p*	*a*	*s*
Code	0000	0001	0010	0011	100	1010
Symbol	*r*	*g*	*d*	*i*	*o*	*space*
Code	0100	1011	110	0101	111	011

Thus the phrase “The pasargad is good” is encoded as follows (table 9):

Table 9.Huffman code of phrase The pasargad is good.

T	*h*	*e*		*p*	*a*	*s*	*a*	*r*	*g*
0000	0001	0010	1010	0011	100	1010	100	0100	1011
a	*d*		*i*	*s*		*g*	*o*	*o*	*d*
100	110	1010	0101	1010	1010	1011	111	111	110

“The pasargad is good”:
00000001001010100011100101010010010
11100111010001011010101011111111110
Huffman’s decoding operation is very simple in which no two codes with identical prefix is found. Therefore, codes related to each character are easily discerned from each other.

5. SECURITY MEASURES

To enhance security of voting stages paying attention to the following point can be useful:

- Voting and confirmation terminals should be in harmony with raw ballot used in each station in terms of encoding in the sense that the ballot cannot be read or written outside its specific voting terminal. Paying attention to this point prevents attacker from stealing away a voting terminal from a voting station and using it in its advantage in another station.

- Voting terminals should not have any connection to Internet network and identification of voter is done only through people statistics registration terminal which is used by voting committee. Disconnection of this terminal from Internet prevents voting system from being attacked by hackers and attackers.

- Counting of votes should be done both by data reader and electronically. This prevents likely elimination of votes by colluding people, attackers and active and inactive parasites.

- Voting station and its environment should be protected against deliberate and unintentional electromagnetic waves such as Zapper attack, active and inactive parasites. Parasite production attack is an optional non-acceptance service attack, because allows attacker to easily choose a certain set of voters for attack. This attack due to its very extensive function scope is incomparable. Preventing this attack is not possible, unless in walls, doors, and windows of every voting station, an electronic buffer is used.

- When displacing boxes containing votes, these boxes should be put in faradic cage made of very thick sheets of conducting materials like aluminium. Faradic cage act like an insulator against waves and does not allow any wave to reach inside the boxes (contactless smart cards). In addition, outside the votes box, individual ballots (whether used or not used) should be put inside envelopes made of conductors.

- Voting and confirmation terminals should be so set that in the time of reading or writing a tag, a beep is heard from them. This by producing sound prevents an attacker to attack a voting system and to start changing the votes and arise voting committee's suspicion [2].

- Voting terminals after each time using by voters, should pass a waiting time (e.g. 30 seconds) and prevents attacker from changing votes or giving invalid votes by taking the attacker a long time to do so and this may draw voting committee's attention [3].

6. CONCLUSION

Despite presence of relay attacks which is one of the strongest attacks in the field of RFID technology, we showed that by using a secure distance bounding protocol, these threats can be properly deterred. To enhance security of electronic voting plan, we designed a distance bounding protocol named Pasargad which minimizes attacker's success chance for implementation of relay attack. In Pasargad Protocol, Huffman protocol and another algorithm named Pars which is designed by us have been used. In this protocol, by measuring distance between tag and data reader, relay attack by attackers is prevented. Pasargad distance bounding protocol is easily implementable on RFID systems. First, we have simulated this protocol and then implemented it and have obtained good results.

ACKNOWLEDGEMENTS

In the end, we are very grateful to International Saba pooyan Asia Company and Tehran branch of ICT Telecommunication Office for their support of this article.

REFERENCES

[1] Roussos G, Kostakos V. RFID in pervasive computing: state-of-the-art and outlook. Pervasive and Mobile Computing 2009; 5:110–31.

[2] D. Gritzalis, editor. Secure Electronic Voting. Springer-Verlag, Berlin Germany, 2003.

[3] D. W. Jones. Problems with Voting Systems and the Applicable Standards May 2001.

[4] K. Finkenzeller. RFID Hand book: Fundamentals and Applications in Contactless Smart Cards and Identification. John Wiley & Sons, 2003.

[5] N. Eberhardt, towards, "RFID performance benchmark test", Technological report, Auto – ID, Massachusetts Institute of Technology, 2008.

[6] Y. Desmedt, C. Goutier, and S. Bengio. Special uses and abuses of the Fiat-Shamir passport protocol. In CRYPTO, pages 21–39, 1987.

[7] S. Brands and D. Chaum. Distance-Bounding Protocols. Advances in Cryptology - EUROCRYPT'93, Lecture Notes in Computer Science 765: 344-359, 1993.

[8] S. Capkun and J.-P. Hubaux. Secure Positioning in Wireless Networks. IEEE Journal on Selected Areas in Communications, 24(2), February, 2006.

[9] G.P. Hancke and M.G. Kuhn. An RFID Distance Bounding Protocol. Proceedings of the IEEE/Create-Net Secure Comm, 67- 73, 2005.

[10] C. Meadows, R. Poovendran, D. Pavlovic, L.W. Chang, and P. Syverson. Distance Bounding Protocols: Authentication Logic Analysis and Collusion Attacks. Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks, 279- 298, Springer-Verlag, 2007.

[11] J. Reid, J.M.G. Nieto, T. Tang, and B. Senadji. Detecting Relay Attacks with Timing-Based Protocols. Proceedings of the 2nd ACM Symposium on Information, Computer, and Communications Security, 204-213, 2007.

[12] C. Meadows, R. Poovendran, D. Pavlovic, L.W. Chang, and P. Syverson. Distance Bounding Protocols: Authentication Logic Analysis and Collusion Attacks. Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks, 279- 298, Springer-Verlag, 2007.

[13] S. Brands and D. Chaum. Distance-Bounding Protocols. Advances in Cryptology - EUROCRYPT'93, Lecture Notes in Computer Science 765: 344-359, 1993.

[14] D. Singelee, B. Preneel, Distance bounding in noisy environments, in: F. Stajano et al. (Eds.), ESAS 2007, LNCS, vol. 4572, Springer, Heidelberg, 2007, pp. 101–115.

[15] Julia Abrahams, \Huffman-Type Codes for Infinite Source Distributions," Journal of the Franklin Institute, 331B (3) (1994) 265-271.

Authors

Mohammad Arjmand received the Bachelor's degree in Telecommunication Engineering from I.H.University, Tehran, Iran, in 2008 and Master's degree in Telecommunication in the field of Cryptography from IHU, Tehran, Iran (2011). Currently, he is research assistant (RA) at the research centre of cryptography, IHU, Tehran, Iran.
His research interest includes: Cryptography, Applications of RFID.

Mahmoud Gardeshi received his Erudition Degree in applied mathematics from Amir Kabir University, Islamic Republic of Iran in 2000. Currently, he is a researcher at the I. H. University. His research interest includes: cryptography and information security.

Reza Taheri Zohur received his Bachelor's Degree in Electronic engineering from Sari Technical University, Islamic Republic of Iran in 2009. Currently, He is a student of Information and Telecommunication course at the Sharif Technical University. He interested in Programming and RFID Security**.**

Mohammad Kazemi received his Bachelor's Degree in Software engineering from Chalous Technical University, Islamic Republic of Iran in 2009. Currently, He is a student of Information and Telecommunication course at the Sharif Technical University. He interested in RFID System Management.