NLHB : A Light-weight, Provably-secure Variant of the HB Protocol Using Simple Non-linear Functions

Mukundan Madhavan
Electrical Engg. Department,
IIT Madras.
Email: mukundanm@gmail.com

Andrew Thangaraj
Electrical Engg. Department,
IIT Madras.
Email: andrew@iitm.ac.in

Kapali Viswanathan
HP Labs India, Bangalore.
Email: kapali@hp.com

Yogesh Sankarasubramaniam
HP Labs India, Bangalore.
Email: yogesh@hp.com

***Abstract*—In this paper, we propose a light-weight provably-secure authentication protocol called the NLHB protocol, which is a variant of the HB protocol [6]. The HB protocol uses the complexity of decoding linear codes for security against passive attacks. In contrast, security for the NLHB protocol is proved by reducing the provably hard problem of decoding a class of non-linear codes to passive attacks. We demonstrate that the existing passive attacks([10],[3]) on the HB protocol family, which have contributed to considerable reduction in its effective key-size, do not work against the NLHB protocol. From the evidence, we conclude that smaller-key sizes are sufficient for the NLHB protocol to achieve the same level of passive attack security as the HB Protocol. Further, for this choice of parameters, we provide an implementation instance for the NLHB protocol for which the Prover/Verifier complexity is lower than the HB protocol, enabling authentication on very low-cost devices like RFID tags.**

Keywords: HB protocol, LPN problem, Secure and Efficient Authentication Protocol, Passive attacks, RFID tags.

I. Introduction

The HB [6] protocol is a low-complexity RFID authentication algorithm whose security is based upon the hardness of the "Learning Parity in Noise" (LPN) problem [1], which is known to be NP-Hard. Cryptanalysis of the HB authentication protocol has resulted in efficient solutions to the LPN problem. Notable among these are the LF2 algorithm [10] and the algorithm proposed by Carrijo *et al.* [3]. These new solutions have significantly reduced the effective key-size of the HB protocol family that depends on the hardness of decoding linear codes for security against passive adversaries.

In this paper, we define and consider the UNLD problem, which is a decoding problem for a specific class of non-linear codes. We prove hardness of UNLD by reducing the LPN problem to the UNLD. Following this, we propose the NLHB protocol, which is a carefully constructed variant of the HB protocol. Security of NLHB against passive attacks is proved by reduction of UNLD to the passive attacks.

The basic idea behind the NLHB protocol is the use of a carefully-chosen non-linear Boolean function on the linear parities generated in the HB protocol. The use of the non-linear function considerably weakens the effectiveness of passive attacks like LF2 [10] that depend on the linearity of the parities. Therefore, key efficiency is higher in NLHB when compared to HB.

For implementation, we demonstrate a certain quadratic form chosen from the general family of functions that we propose for the NLHB, which presents a specific low-cost candidate for the protocol. Further, we show that the Prover/Verifier complexity of NLHB protocol can be lower than that of the HB protocol because of the use of smaller keys.

In summary, the main contribution of this paper is a low-cost, provably-secure extension of the HB protocol through the use of simple non-linear functions on parities. Because of the non-linearity, the proposed NLHB protocol has better resistance to known passive attacks on the HB family resulting in higher key efficiency and cheaper implementations.

The paper is organized as follows. In Section 2, we give a brief introduction on the HB protocol, related security models and the LPN problem. In Section 3, we describe the UNLD problem and prove its NP-Hardness. This is followed by a description of the NLHB protocol and its security proofs. Section 4 contains discussions on the resistance of NLHB to passive attacks and its Prover complexity. Section 5 concludes the paper.

II. The HB Protocol Literature

A. HB Protocol

The HB protocol is a symmetric-key authentication protocol. The Prover and Verifier share a random k-bit secret key $\mathbf{s}$. The protocol has two public probability parameters $\epsilon, \epsilon' \in \;]0, \frac{1}{2}[$ such that $\epsilon < \epsilon'$. To authenticate, the Verifier challenges the Prover with a random $k \times n$ matrix, to which the Prover responds with $\mathbf{z} = \mathbf{s}A + \mathbf{v}$. Here, the bits of the vector $\mathbf{v}$ are all i.i.d Bernoulli random variables with parameter ϵ,the multiplication between the vector $\mathbf{s}$ and A is over the binary field $GF(2)$and + denotes XOR addition. The response vector $\mathbf{z}$ is a n-bit vector and the Verifier responds with "Accept" iff $d(\mathbf{z}, \mathbf{s}A) \leq \epsilon' n$, where $d(.)$ denotes Hamming distance. This process, which constitutes one authentication session is shown in Figure 1.

$$\begin{array}{ccc} & \text{Secret Shared } \mathbf{s} & \\ \text{Prover} & & \text{Verifier} \\ & \xleftarrow{\quad A \quad} & \text{Choose } A \in \{0,1\}^{k \times n} \\ \mathbf{z}_{1 \times n} = \mathbf{s}A + \mathbf{v} & \xrightarrow{\quad \mathbf{z} \quad} & \text{“Accept” iff } d(\mathbf{z}, \mathbf{s}A) \leq \epsilon' n \end{array}$$

Fig. 1. Parallelized version of the HB protocol

The parameters ϵ,ϵ', and n are fixed so that both the probability of rejecting an honest Prover as well as the probability of positively authenticating an attacker giving random responses are negligible ([10], Figure 2). The HB Protocol has been proven secure in the Passive attack model which is defined below.

Definition 1 (Passive attack model ([7],[9])): In this model, the adversary algorithm is two-phased. In the first phase (called the query phase), the adversary has access to the transcripts from q authentication sessions between an honest Prover and Verifier. In the second phase (called the cloning phase), the adversary tries to impersonate an honest Prover to the Verifier.

B. The LPN Problem and Passive Attacks

Definition 2 (LPN Problem [7]): Let $\mathbf{s}$ be a random binary k-bit vector. Let $\epsilon \in]0, \frac{1}{2}[$ be a constant error parameter. Let A be a random $k \times n$ matrix, and let $\mathbf{v}$ be a random n-bit vector such that $\mathrm{wt}(\mathbf{v}) \leq \epsilon n$, where $\mathrm{wt}(\mathbf{v})$ denotes the Hamming weight of $\mathbf{v}$. Given A, ϵ and $\mathbf{z} = (\mathbf{s}A) + v$, find a k-bit vector $\mathbf{s}'$ such that $d(\mathbf{z}, \mathbf{s}'A) \leq \epsilon n$.
For large n, this is equivalent to finding the vector $\mathbf{s}$. The LPN problem has been proven to be both NP-Hard [1] and average-case hard [6]. The LF2 algorithm [10], which was proposed after the HB protocol, is currently the best-known algorithm for solving the LPN problem. The LF2, which is an improvement over the BKW algorithm [2] requires far fewer challenge-response pairs and lesser complexity than the BKW algorithm. The key idea behind the LF2 algorithm is to do column additions to the query matrix (and adding the corresponding response bits) so that the resulting columns are all zero in certain row positions, making the corresponding key-bits irrelevant to the matrix product $\mathbf{s}A$. Then the remaining key-bits are found through exhaustive search in the diminished key-space (that does not contain the irrelevant bits). This process is repeated, doing exhaustive search over various subsets of the key-bits, until the whole secret key is found. A second new passive attack effective at low key-sizes and low noise-probabilities was also proposed by Carrijo *et al.* [3]. This attack tries to pick noise-free bits from the response vector and find the key through Gaussian elimination on the system of equations formed from these bits alone. Both attacks heavily depend on the Prover's response being a noisy codeword of the linear code generated by A.

As a consequence of these attacks, a LPN instance using as many as 512 bits of secret can be attacked with a complexity of just 2^{80} operations (as opposed to 2^{512}).

The main idea in this paper is to replace the linear parity generation part sA in the HB protocol with a non-linear version $f(sA)$ for a suitable public function f. The following characteristics are desirable for such a function f:

1) The public function f must allow for security proofs through reduction of hardness of decoding problems to the passive attacks.
2) The function f must be simple enough to implement on low-cost devices.
3) The function f must provide better resistance to known passive attacks that solve the LPN problem.

In the next section, we describe a specific class of non-linear Boolean vector functions and discuss some of its properties that will be used in the security reductions. We discuss the other characteristics like implementation-cost and passive attack resistance in later sections.

III. The UNLD Problem and the NLHB Protocol

A. The Function f

Let D and p be positive integers such that $D = n - p$ (n is as described in the HB protocol). The NLHB function f is constructed as follows. Each bit $y_i; i \in [1, .., D]$ of the output $\mathbf{y} = f(\mathbf{x}); \mathbf{y} \in \{0,1\}^D, \mathbf{x} \in \{0,1\}^n$ will be computed as

$$y_i = x_i + g([x_{i+1}, .., x_{i+p}]), \tag{1}$$

where x_i are the n-bits of $\mathbf{x}$ and $g : \{0,1\}^p \Rightarrow \{0,1\}$ is a Boolean function containing only non-linear terms. Below, we list some important properties for this class of functions.

1) $f : \{0,1\}^n \Rightarrow \{0,1\}^D$
2) f is a non-linear function.
3) For uniformly distributed $\mathbf{x} \in \{0,1\}^n$, $f(\mathbf{x})$ is uniformly distributed in $\{0,1\}^D$.

A proof of Property 3 is provided in Appendix A. As a specific example, the function

$$y_i = x_i + x_{i+1}x_{i+2} + x_{i+2}x_{i+3} + x_{i+3}x_{i+1}, 1 \leq i \leq D \tag{2}$$

is a part of this function family for $p = 3$ (i.e $D = (n-3)$). As we can see, using functions like the one in Equation 2 in a protocol requires very low additional complexity(only 3 AND and 3 XOR gates here) and can easily be accomodated into any cheap device like RFIDs. In the next section, we describe how this function family can be used to create a robust protocol and prove the protocol's security for any given f in this family. In later sections, we use our specific candidate to demonstrate that the new protocol has increased passive attack resistance while still maintaining low implementation complexity.

B. UNLD Problem

Suppose $A_{k\times n}$ is the generator matrix of a linear code. Then all vectors of the form $\mathbf{s}A$ are codewords of this code. When we apply the function f to these codewords, i.e, we compute $f(\mathbf{s}A)$, the output vectors $\{f(\mathbf{s_i}A)\}_{i=1}^{2^k}$ can be viewed as a non-linear code. We now define the UNLD problem, which (in words) is the problem of decoding the class of non-linear codes defined by f and A as $\{f(\mathbf{s_i}A)\}_{i=1}^{2^k}$.

Definition 3 (UNLD Problem): Let $\mathbf{s}$ be a random k-bit binary vector. Let $\epsilon \in]0, \frac{1}{2}[$ be a constant error parameter. Let A be a random $k \times n$ binary matrix and let $\mathbf{v}$ be a random D-bit vector such that $\mathrm{wt}(\mathbf{v}) \leq \epsilon D$, where $\mathrm{wt}(\mathbf{v})$ denotes the Hamming weight of $\mathbf{v}$. Given A, ϵ and $\mathbf{z} = f(\mathbf{s}A) + \mathbf{v}$, find the k-bit vector $\mathbf{s}$.

We prove the hardness of the UNLD problem by reducing a random instance of the LPN problem to the UNLD. To show the reduction, we assume an existential algorithm X that can solve the UNLD problem. We construct an algorithm S, which can solve a random LPN instance, when given access to X.

Theorem 1 (LPN reduces to UNLD): Let A be a random $k \times n$ matrix, $\mathbf{v}'$ be a $(n-p)$-bit Bernoulli noise vector, and $\mathbf{s}$ be a random k-bit vector. Suppose there exists a probabilistic polynomial-time (PPT) algorithm X with input $\langle A, \mathbf{y} = f(\mathbf{s}A) + \mathbf{v}' \rangle$ that can output $\mathbf{s}$ with probability atleast δ. Then, there also exists a PPT algorithm S that can solve a random LPN problem instance $\langle G_{k\times n'}, \mathbf{z} = \mathbf{m}G + \mathbf{v} \rangle$ for randomly chosen $\mathbf{m}$, Bernoulli noise vector $\mathbf{v}$ and $n' \leq \frac{(n-1)}{p}, k < n'$ with probability at least δ.

Proof: Let $\mathbf{z} = [z_1, ..., z_{n'}]$ and $\mathbf{v} = [v_1, ..., v_{n'}]$ be the constituent bits of the vectors described above. The algorithm S, having access to algorithm X works as follows to solve a random LPN instance $\langle G, \mathbf{z}\rangle$ passed to it.

1) Pick $r_i; i \in [1, n'-1]; r_i \geq (p-1)$,such that $\sum_{i=1}^{n'-1} r_i = n - p - n'$.
2) Insert r_i zeros between bit z_i and z_{i+1} of $\mathbf{z}$. This gives the vector $\mathbf{y}_{(n-p)} = [z_1, 0..0, z_2, 0..0, z_3, 0.....0, z_{n'}]$.
3) Similarly insert r_i columns of zeros in between columns i and $i+1$ of G to get the matrix A. Also insert p columns of zeros after the last column of A. Now, the dimension of A is $k \times n$ and A is of the form $A = [\mathbf{g_1}\underline{00}..\underline{0}\mathbf{g_2}\underline{00}..\underline{0}.....\mathbf{g_n}\underline{00}..\underline{0}]$, where $\mathbf{g_i}$ are the columns of G.
4) Pass $\langle A, \mathbf{y}\rangle$ to X and get back $\mathbf{m}'$.
5) Return $\mathbf{m}'$ as the estimate of the LPN secret $\mathbf{m}$.

Analysis: Consider the vector $\overline{\mathbf{x}} = \mathbf{m}A$. We can see that $\overline{\mathbf{x}} = [x_1 00..0x_2 00..0x_3 00.0....x_n 00..0]$, where $x_1, x_2, .., x_n$ are the bits of $\mathbf{x} = \mathbf{m}G$. We also see that, since g is a function with only non-linear terms (i.e each term in g is some kind of product of its input bits) and $r_i \geq (p-1)$, the vector $f(\overline{\mathbf{x}})$ can be written as $f(\overline{\mathbf{x}}) = [x_1 00..0x_2 00..0x_3.....00x_n]$ as all the product terms from g go to zero.
So, the vector $\mathbf{y}$ is of the form $f(\overline{\mathbf{x}}) + \mathbf{v}'$ where $\mathbf{v}' = [v_1, 00..0, v_2, 00..0, v_3, 00....., v_n]$, v_i being the bits of the LPN noise vector $\mathbf{v}$. In other words, $\mathbf{y} = f(\mathbf{m}A) + \mathbf{v}'$. Hence, by definition, X, since will return $\mathbf{m}$. Since S succeeds whenever X succeeds, its probability of success is atleast δ. So, if δ is non-negligible, the LPN problem can be solved easily. ■
We can always pick r_i satisfying the condition in Step 1 for any n and $n' \leq \frac{n-1}{p}$. One could initially fix $r_i = (p-1); \forall i$. Then $\sum r_i = (n'-1)(p-1) = n'p - p - n' + 1$. Then one can add the difference $(n-p-n') - (n'p-p-n'+1) = n - n'p - 1$ (which is always positive) to say, r_1, giving us a new set $\{r_i\}$ satisfying the conditions in Step 1 for any n.

C. NLHB Protocol

Having established the hardness of the UNLD problem, we now propose the NLHB protocol that is based on this problem. Figure 2 shows one session of the NLHB protocol. The protocol is similar to the HB protocol except that the Prover response is now $\mathbf{z}_{1\times D} = f(\mathbf{s}A) + \mathbf{v}$, $\mathbf{v}$ being the Bernoulli noise-vector with parameter ϵ. Here, D has to be large enough ($\approx$1000) and (D,ϵ,ϵ') have to satisfy the conditions satisfied by the HB protocol parameters (n,ϵ,ϵ')(for instance, see [10][Figure 2]). For example, $D = 1164$, $\epsilon = .25$ and $\epsilon' = .348$ is a possible parameter set.

$$\begin{array}{lcl} & \text{Secret Shared } \mathbf{s} & \\ \text{Prover} & & \text{Verifier} \\ & \xleftarrow{\quad A \quad} & \text{Choose } A \in \{0,1\}^{k\times n} \\ \mathbf{z}_{1\times D} = f(\mathbf{s}A) + \mathbf{v} & \xrightarrow{\quad \mathbf{z} \quad} & \text{“Accept” iff} \\ & & \quad d(\mathbf{z}, f(\mathbf{s}A)) \leq \epsilon' D \end{array}$$

Fig. 2. Parallelized version of the NLHB protocol

Since $f(\mathbf{s}A)$ is some unknown codeword of the random non-linear code $\{f(\mathbf{s_i}A)\}_{i=1}^{2^k}$. To find the secret $\mathbf{s}$, the attacker now has to decode this random non-linear code (from Bernoulli noise) instead of the linear code with generator matrix A. We will show in Section IV that none of the existing passive attacks on HB protocol family work on our protocol and that, for certain choices of f within this family, the protocol complexity is very low.

D. Security Proofs For NLHB In Passive Model

The proof of security for NLHB in the passive model involves reductions to the forging of the protocol in the passive model from the solving of the UNLD problem. It is detailed in Theorems 2 and 3. Theorem 2 shows that it is NP-hard to distinguish between the output of a NLHB protocol oracle $\mathcal{A}_{\mathbf{s},\epsilon,f}$(an oracle gives out the NLHB transcript $\langle A, \mathbf{z}\rangle$ for some unknown secret) and an oracle U_{kn+D} which outputs a stream of $(kn + D)$ uniformly distributed bits. It does this by reducing the UNLD problem to this distinguishing problem. Similarly, Theorem 3 proves the hardness of forging NLHB protocol by reducing the forging problem from the problem of distinguishing the above-said oracles. Together, these Theorems imply the existence of a UNLD solver if an algorithm capable of forging NLHB in the passive model exists, which is in contradiction to the hardness of UNLD. Thus, we prove the security of NLHB by contradiction.

These proofs are mostly based on the proof of security given for the HB protocol in ([9],[8]), with suitable modifications and additions to support the function f. So, we simply state the final result of the proofs below.

Theorem 2: (Distinguishing NLHB oracle from random oracle is equivalent to finding the NLHB protocol secret) Suppose there exists a probabilistic polynomial-time algorithm Y making q queries to the oracle and outputting $0/1$, running in time t, such that [1]

$$\left|Pr\left[\mathbf{s} \leftarrow \{0,1\}^k : Y^{\mathcal{A}_{\mathbf{s},\epsilon,f}} = 1\right] - Pr\left[Y^{U_{kn+D}} = 1\right]\right| \geq \delta$$

Then there exists X making $q' = O(q.\delta^{-2}log(k))$ queries running in time $t' = O(t.k.\delta^{-2}log(k))$ such that

$$Pr\left[\mathbf{s} \leftarrow \{0,1\}^k : X^{\mathcal{A}_{\mathbf{s},\epsilon,f}} = \mathbf{s}\right] \geq \delta/4$$

Theorem 3: (Forging NLHB Prover is equivalent to distinguishing NLHB Oracle from random oracle) If $Adv_Z^{NLHB-attack}(k, \epsilon, u, f) = \delta$ is non-negligible [2] for some polynomial time adversary Z, then the UNLD problem can be efficiently solved.

IV. Implementation and Efficiency

In this section, we use the specific low-cost candidate for f given in Equation 2 to demonstrate how existing passive attacks on the HB protocol fail against the NLHB protocol. Then, we compare the Prover complexity of NLHB and HB protocols and demonstrate that the NLHB Prover is required to carry out lesser operations when compared to a HB prover that achieves the same level of security.

[1] $Y^{\mathcal{A}_{\mathbf{s},\epsilon,f}}$ implies Y has oracle access to $\mathcal{A}_{\mathbf{s},\epsilon,f}$

[2] "Adv" is the difference between success probability of Z and an algorithm giving D random bits as Prover Response

A. Resistance Against Current Passive Attacks

Using the specific f in Equation 2, we will show how the existing LF2 attack on LPN is ineffective on the NLHB protocol. Let $\mathbf{x} = [x_1, ..., x_n] = \mathbf{s}A = [\mathbf{s}.\mathbf{a_1}, ..., \mathbf{s}.\mathbf{a_n}]$, where $[\mathbf{a_1}, ..., \mathbf{a_n}]$ are columns of A. Let $\mathbf{y} = f(\mathbf{x})$. Then, the passive adversary to NLHB has access to $\mathbf{z} = \mathbf{y} + \mathbf{v}$.

As explained in Section II, the LF2 (or BKW) algorithm works by repeatedly adding the columns of the matrix A and obtaining the response corresponding to this new matrix by adding the responses corresponding to the added columns. We examine the result when the attacker does one column addition. Let the attacker modify A into $A' = [\mathbf{a_1}, ..., \mathbf{a_j} + \mathbf{a_k}, ..., \mathbf{a_n}]$, i.e, he adds the k^{th} column to the j^{th} column. The corresponding matrix product between $\mathbf{s}$ and A' will be $\overline{\mathbf{x}} = [x_1, x_2, ..., x_j + x_k, ..., x_n]$, i.e $\overline{\mathbf{x}}$ is the same as $\mathbf{x}$ except that the j^{th} position value is $x_j + x_k$. Let $\overline{\mathbf{y}} = f(\overline{\mathbf{x}})$. Now let us compare the relation between the unnoised responses $\mathbf{y}$ and $\overline{\mathbf{y}}$. As can be seen, the only output bits getting affected by the change of matrix are the ones with indices $(j-3), (j-2), (j-1), j$. We readily see the following relationships.

$$\begin{aligned}
y_{j-3} &= x_{j-3} + x_{j-2}x_{j-1} + x_{j-1}x_j + x_jx_{j-2}.\\
y_{j-2} &= x_{j-2} + x_{j-1}x_j + x_jx_{j+1} + x_{j+1}x_{j-1}.\\
y_{j-1} &= x_{j-1} + x_jx_{j+1} + x_{j+1}x_{j+2} + x_{j+2}x_j\\
y_j &= x_j + x_{j+1}x_{j+2} + x_{j+2}x_{j+3} + x_{j+3}x_{j+1}.\\
\overline{y}_{j-3} &= x_{j-3} + x_{j-2}x_{j-1} + x_{j-1}(x_j + x_k) + (x_j + x_k)x_{j-2}.\\
\overline{y}_{j-2} &= x_{j-2} + x_{j-1}(x_j + x_k) + (x_j + x_k)x_{j+1} + x_{j+1}x_{j-1}.\\
\overline{y}_{j-1} &= x_{j-1} + (x_j + x_k)x_{j+1} + x_{j+1}x_{j+2} + x_{j+2}(x_j + x_k).\\
\overline{y}_j &= x_j + x_k + x_{j+1}x_{j+2} + x_{j+2}x_{j+3} + x_{j+3}x_{j+1}.
\end{aligned}$$

Let us denote the errors between these corresponding bits as $E_{j-3}, E_{j-2}, E_{j-1}, E_j$. From the above equations, we get

$$\begin{aligned}
E_{j-3} &= y_{j-3} + \overline{y}_{j-3} = x_{j-1}x_k + x_kx_{j-2},\\
E_{j-2} &= x_{j-1}x_k + x_kx_{j+1},\\
E_{j-1} &= x_{j+1}x_k + x_kx_{j+2},\\
E_j &= x_k.
\end{aligned}$$

Each error term above is an unknown bit to the attacker, since he does not have access to either a noised or un-noised version of these terms. So, the attacker has to guess the error bits $E_{j-3}, E_{j-2}, E_{j-1}, E_j$ that need to be added to the new response to get the right responses corresponding to the new matrix. The amount of uncertainty involved in guessing these bits can be found from the entropy of $[E_{j-3}, E_{j-2}, E_{j-1}, E_j]$. Since the bits x_i are uniformly distributed, it can easily be seen that this entropy is equal to 2.5 bits. So each time a column is added, the attacker has to guess 2.5 bits on an average. Since there are many such additions needed in the LF2 attack, this attack is not feasible against the NLHB protocol. In Table I, we give some function choices that maximise this entropy for a given p. We can see from the table that the entropy increases with increase in p, meaning that higher values of p would make LF2 attacks harder. Similar arguments can be given for the infeasibility of the Imai [3] attack, that also relies heavily on linearity.

The infeasibility of these passive attacks indicates that the NLHB protocol can achieve 80-bit security using smaller keysizes than the HB protocol (which uses 512 bits) and the general lack of solutions to the problem of decoding the random non-linear codes implies that it is reasonable to use keysizes very close to 80 bits with this protocol. However, keeping in mind, the possibility of new attacks, we suggest using 128-bit keys as secrets.

B. Comparison of Prover Complexity of NLHB and HB

We calculate the Prover (or Verifier) algorithm's complexity in terms of binary (scalar) additions and scalar multiplications. Further, since the complexity involved in adding noise is the same in both protocols, we compare the complexity involved in the calculation of the un-noised responses in the Prover (or Verifier).

We recall that the HB protocol response for the challenge matrix $A_{k \times n}$ is given by $\mathbf{z} = \mathbf{s}A + \mathbf{v}$. The matrix product $\mathbf{s}A$ requires kn scalar multiplications and $(k-1)n$ (binary) additions for its calculation. Assuming that $\epsilon = .25$ and $\epsilon' = .348$, the recommended response length and key-size (for 80-bit security) for HB are $n = 1164$ [10] and around $k = 512$.

In the NLHB protocol, we have a $k \times (D+p)$ challenge-matrix A which we use to find $\mathbf{s}A$. This requires $k(D+p)$ scalar multiplications and $(k-1)(D+p)$ additions. If we assume that we use the function f in (2) (with $p = 3$), we require $3D$ scalar multiplications and $3D$ additions for evaluating the function f. So to calculate $f(\mathbf{s}A)$, we need $k(D+3) + 3D$ multiplications and $3D + (k-1)(D+3)$ additions. (Since we add a length-D noise vector in NLHB, D will be 1164 here.)

For the sake of comparing complexities, if we assume a high-security version of NLHB which uses $k = 512$, then we see that NLHB needs a total of 600996 multiplications and 599829 additions, whereas HB protocol requires 595968 multiplications and 594804 additions. This approximately represents a 0.85% increase in the both the number of multiplications and additions. However, with $k = 128$, the computation of noise-free NLHB response requires 152868 scalar multiplications and 151701 additions, which is far less than the number of computations needed for a HB protocol Prover to achieve the same level of security.

V. Conclusion And Future Work

In this paper, we have proven the hardness of a non-linear decoding problem that we call the UNLD problem and proposed the NLHB authentication protocol, which is a variant of HB. This new protocol has better passive attack security than the HB protocol and is very light-weight, having very low Prover complexity. Though we have not discussed active attacks here, it is straightforward to extend the NLHB protocol, in the spirit of the HB$^+$ protocol [7], to resist against the active attacks specified in [7]. However, such an extension, like the HB$^+$, would be vulnerable to Man-In-The-Middle attacks like the ones shown on HB$^+$ [4], [12](part of a prevention-based attack model). It would be interesting to see modifications to the NLHB protocol that can resist such attacks.

References

[1] E.R. Berlekamp, Robert.J.McEliece and Henk.C.A.van Tilborg. On the Inherent Intractability of Certain Coding Problems. *IEEE Transactions on Information Theory*, 1978.

[2] A. Blum, A. Kalai, and H.Wasserman. Noise-tolerant Learning, the Parity Problem, and the Statistical Query Problem Journal of the ACM 50,4, July 2003, pp. 506-519.

p	Function Achieving Maximum entropy for given p	Maximum Entropy Achieved for given p
2	$y_i = x_i + x_{i+1}x_{i+2}$	2
3	$y_i = x_i + x_{i+1}x_{i+2} + x_{i+1}x_{i+3}$	2.5
4	$y_i = x_i + x_{i+1}x_{i+4} + x_{i+2}x_{i+3}$	3

TABLE I
MAXIMUM ENTROPY ACHIEVED FOR A GIVEN p AND THE FUNCTION ACHIEVING THIS MAXIMUM

	k	ϵ	ϵ'	Size of Challenge Matrix	Length Of Prover Response	Scalar Multiplications	Scalar Additions
HB	512	.25	.348	512×1164	n=1164	595968	594804
NLHB	128	.25	.348	128×1167	D=1164	152868	151701

TABLE II
PROVER/VERIFIER COMPLEXITIES FOR HB AND NLHB FOR f WITH $p = 3$, FALSE-REJECT PROBABILITY $P_{FR} = 2^{-40}$ AND FALSE-ACCEPT PROBABILITY $P_{FA} = 2^{-80}$ AND 80-BIT SECURITY.

[3] Josè Carrijo, Rafael Tonicelli, Hideki Imai, Anderson C. A. Nascimento. A Novel Probabilistic Passive Attack on the Protocols HB and HB$^+$. Available from http://eprint.iacr.org/2008/231.pdf.

[4] H. Gilbert, M.J.B. Robshaw, and H. Sibert. An Active Attack Against HB$^+$: A Provably Secure Lightweight Authentication Protocol. *IEE Electronics Letters*, vol. 41, number 21, 1169-1170, 2005.

[5] H. Gilbert, M.J.B. Robshaw, and Y. Seurin. Good Variants of HB$^+$ are Hard to Find. In Proceedings of *Financial Crypto 2008*.

[6] N. Hopper and M. Blum. A Secure Human-Computer Authentication Scheme. Technical Report CMU-CS-00-139, Carnegie Mellon University, 2000.

[7] A. Juels and S. Weis. Authenticating Pervasive Devices with Human Protocols. *Adv. in Cryptology - Crypto 2005*, LNCS vol. 3621, Springer-Verlag, pp. 293-308, 2005.

[8] J. Katz and A. Smith. Analysing the HB and HB$^+$ Protocols in the "Large Error" Case. Available from http://eprint.iacr.org/2006/326.pdf.

[9] J. Katz and J. Shin. Parallel and Concurrent Security of the HB and HB$^+$ Protocols.*Advances in Cryptology - Eurocrypt 2006*,LNCS, vol. 4004, 73-87, Springer, 2006.

[10] E. Levieil and P.A. Fouque. An Improved LPN Algorithm. *Proceedings of SCN 2006*, LNCS vol. 4116, 348-359, Springer, 2006.

[11] Kishan Chand Gupta and Palash Sarkar. Construction of Perfect Nonlinear and Maximally Nonlinear Multiple-Output Boolean Functions Satisfying Higher Order Strict Avalanche Criteria. *IEEE Trans. On Information Theory*, Vol. 50, No. 11, Nov. 2004.

[12] Khaled Ouafi, Raphael Overbeck, Serge Vaudenay. On the Security of HB$^\#$ against a Man-in-the-Middle Attack. *Asiacrypt 2008*, 108-124.

APPENDIX

Theorem 4 (f is a Balanced Function): If the input to the function f is uniformly distributed, so is its output.

Proof: We first prove that each bit of the output is balanced. For this, we consider $\Pr[y_i = 1]$.

$$= \Pr[x_i + g(x_{i+1}, ..., x_{i+p}) = 1]$$

$$= \frac{1}{2}\Pr\left[g(x_{i+1}, ..., x_{i+p}) = 1 \mid x_i = 0\right] + \frac{1}{2}\Pr\left[g(x_{i+1}, ..., x_{i+p}) = 0 \mid x_i = 1\right]$$

Since the input vector is uniform, the bits of $\mathbf{x}$ are independent. So, this is equal to

$$\begin{aligned} &= \frac{1}{2}\Pr\left[g(x_{i+1}, ..., x_{i+p}) = 1\right] + \frac{1}{2}\Pr\left[g(x_{i+1}, ..., x_{i+p}) = 0\right] \\ &= \frac{1}{2} \end{aligned} \quad (3)$$

So each bit of the output is balanced. Now, we use this to prove our theorem. To this end, we first define the following vectors. Let $\mathbf{y^i} = [y_{D-i+1}, .., y_D]$ be the vector containing the last i bits of $\mathbf{y}$. So $\mathbf{y^D} = \mathbf{y}$. Let $\mathbf{a} = [a_1, ..., a_D]$ be an arbitrary constant D-bit vector. We also define $\mathbf{a^i} = [a_{D-i+1}, .., a_D]$ similar to $\mathbf{y^i}$. Now consider the probability $\Pr[\mathbf{y^i} = \mathbf{a^i}]$.

$$\begin{aligned} \Pr[\mathbf{y^i} = \mathbf{a^i}] &= \Pr[x_{D-i+1} = 0]\Pr[\mathbf{y^i} = \mathbf{a^i} \mid x_{D-i+1} = 0] \\ &\quad + \Pr[x_{D-i+1} = 1]\Pr[\mathbf{y^i} = \mathbf{a^i} \mid x_{D-i+1} = 1] \end{aligned} \quad (4)$$

Since the input is uniformly distributed, this is equal to

$$\begin{aligned} &= \tfrac{1}{2}\Pr[\mathbf{y^i} = \mathbf{a^i} \mid x_{D-i+1} = 0] + \tfrac{1}{2}\Pr[\mathbf{y^i} = \mathbf{a^i} \mid x_{D-i+1} = 1] \\ &= \tfrac{1}{2}\Pr[g(x_{D-i+2}, ..., x_{D-i+p+1}) = a_{D-i+1}, \mathbf{y^{i-1}} = \mathbf{a^{i-1}} \mid x_{D-i+1} = 0] \\ &\quad + \tfrac{1}{2}\Pr[g(x_{D-i+2}, ..., x_{D-i+p+1}) = a_{D-i+1} + 1, \\ &\qquad \mathbf{y^{i-1}} = \mathbf{a^{i-1}} \mid x_{D-i+1} = 1] \end{aligned}$$

We point out that in the vector $\mathbf{y^i}$, only the bit y_{D-i+1} is dependent on x_{D-i+1}. Since both $g(x_{D-i+2}, ..., x_{D-i+p+1})$ and $\mathbf{y^{i-1}}$ are independent of x_{D-i+1}, we can remove the conditioning from the above equation. So the above expression becomes,

$$\begin{aligned} &\frac{1}{2}\Big(\Pr[g(x_{D-i+2}, ..., x_{D-i+p+1}) = a_{D-i+1}, \mathbf{y^{i-1}} = \mathbf{a^{i-1}}] \\ &\quad + \Pr[g(x_{D-i+2}, ..., x_{D-i+p+1}) = a_{D-i+1} + 1, \mathbf{y^{i-1}} = \mathbf{a^{i-1}}]\Big) \end{aligned} \quad (5)$$

Now $g(x_{D-i+2}, ..., x_{D-i+p+1})$ takes binary values 0 and 1. So, by summing the joint probability of $g(x_{D-i+2}, ..., x_{D-i+p+1})$ and $\mathbf{y^{i-1}}$ over these values, we are effectively finding the marginal probability of $\mathbf{y^{i-1}}$. So, we have

$$\Pr[\mathbf{y^i} = \mathbf{a^i}] = \frac{1}{2}\left(\Pr[\mathbf{y^{i-1}} = \mathbf{a^{i-1}}]\right) \quad (6)$$

Plugging $i = D$ in the above equation, and expanding, we have

$$\begin{aligned} \Pr[\mathbf{y^D} = \mathbf{a^D}] &= \frac{1}{2}\left(\Pr[\mathbf{y^{D-1}} = \mathbf{a^{D-1}}]\right) = \frac{1}{2^2}\left(\Pr[\mathbf{y^{D-2}} = \mathbf{a^{D-2}}]\right) \\ &\ \ \vdots \\ &= \frac{1}{2^{D-1}}\left(\Pr[\mathbf{y^1} = \mathbf{a^1}]\right) = \frac{1}{2^{D-1}}\left(\Pr[y_D = a_D]\right) \\ &= \frac{1}{2^D} \end{aligned} \quad (7)$$

from Equation 3. Since this proof holds for any $\mathbf{a^i}$, the output of f is uniformly distributed. ■