2007

Cryptanalysis of Two Lightweight RFID Authentication Schemes

Benessa Defend
University of Massachusetts - Amherst

Kevin Fu
University of Massachusetts - Amherst

Ari Juels

Cryptanalysis of Two Lightweight RFID Authentication Schemes

Benessa Defend   Kevin Fu
University of Massachusetts Amherst
{defend,kevinfu}@cs.umass.edu

Ari Juels
RSA Laboratories, Bedford, MA, USA
ajuels@rsasecurity.com

Abstract

Vajda and Buttyán proposed several lightweight authentication protocols for authenticating RFID tags to readers, and left open the quantifiable cryptographic strength. Our cryptanalysis answers this open question by implementing and measuring attacks against their XOR and SUBSET protocols. A passive eavesdropper can impersonate a tag in the XOR protocol after observing only 70 challenge-response transactions between the tag and reader. In contrast, the theoretical maximum strength of the XOR protocol could have required $16! * 2$ observed transactions to break the key. Our experiments also show that a passive eavesdropper can recover the shared secret used in the XOR protocol by observing an expected 1,092 transactions. Additionally, a nearly optimal active attack against the SUBSET protocol extracts almost one bit of information for each bit emitted by the tag.

1 Introduction

Low-cost RFID tags are being increasingly used in widespread applications such as inventory control, transit systems, livestock management, and building access. Cryptography is needed to prevent unauthorized communication between tags and readers. Vajda and Buttyán [3] developed several lightweight cryptographic protocols for low-cost tags. We show that their XOR and SUBSET protocols provide inadequate protection from passive and active adversaries.

1.1 Background

For their XOR protocol, Vajda and Buttyán [3] detail a possible passive attack that involves guessing the session keys by a brute force attack af-ter observing two consecutive runs of the protocol. The attacker is able to learn the difference between consecutive session keys and forumlate guesses on the subsequent session keys. In approximately $1/16^{th}$ of the cases, the session key will have a special property. Our attack exploits certain statistical properties of the bitstring and determines the correct key value with high probability.

Vajda and Buttyán present an active attack against their SUBSET protocol, which requires more than 256 queries for their parameters. We detail an active attack that requires only 9 queries under the same parameters. The attacker sends the tag a specifically formatted query and sends a subsequent query formulated based on the previous response. We also present a passive attack that will recover all bits with exponentially high probability.

Grain [2] is a lightweight stream cipher that was designed for hardware applications with restricted resources, such as memory and power consumption, and may be suitable for use in RFID tags. [1] presents a cryptanalysis of Grain that recovers the 80-bit key.

2. Vajda and Buttyán Protocol 1

Protocol 1 in Figure 1 is a challenge-response protocol in which the tag and reader share a secret, $k^{(0)}$. To construct a challenge, the reader selects a bitstring x uniformly at random. The reader transmits $a^{(i)} = x^{(i)} \oplus k^{(i)}$ to the tag, where i is the ith transaction between the reader and tag. $k^{(i)}$ is calculated by a permutation of $k^{(0)}$. Because $x^{(i)}$ is random, $a^{(i)}$ is also random. In an information-theoretic sense, $a^{(i)}$ reveals nothing about the secret $k^{(0)}$.

The tag uses its knowledge of $k^{(i)}$ to extract $x^{(i)}$. The tag then responds to the reader with $b^{(i)} = x^{(i)} \oplus k^{(0)}$. Knowing $x^{(i)}$ and $k^{(0)}$, the reader
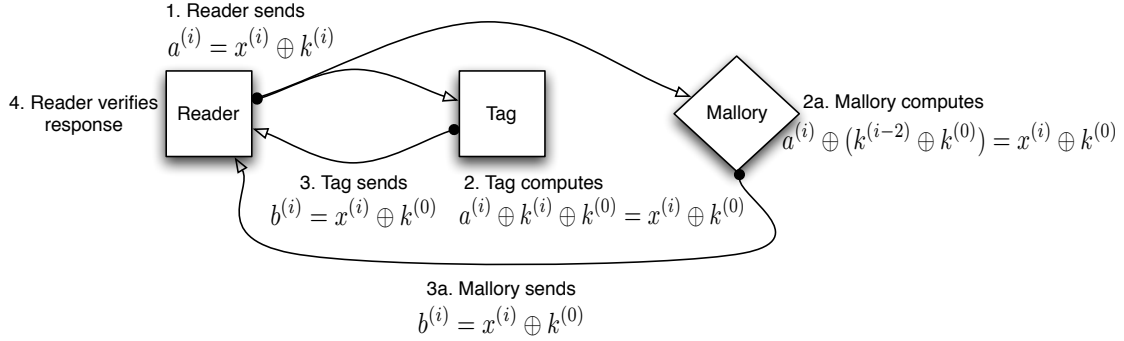
Figure 1. Steps 1-4 of VB protocol 1. The tag knows $k^{(0)}$ and the permutation to calculate $k^{(i)}$. The tag extracts $x^{(i)}$ from the challenge to form a valid response. Steps 2a and 3a show how an active adversary can implement the Repeated Keys Attack to successfully impersonate a tag. Mallory knows $k^{(i-2)} \oplus k^{(0)}$ after observing a challenge/response pair from an earlier transaction between the reader and tag. In this example, the session key is repeated every 2 cycles. Thus, $k^{(i-2)} = k^{(i)}$, and Mallory can form a valid response *without* **knowing $x^{(i)}$, $k^{(i)}$, or $k^{(0)}$.**

can verify the correctness of the tag's response.

The protocol is considered broken when an adversary can send a valid $b^{(i)} = x^{(i)} \oplus k^{(0)}$ or learn the value of $k^{(0)}$. [3] notes that a passive attacker can learn $k^{(i)} \oplus k^{(i+1)}$ after observing two consecutive transactions of the protocol. However, they suggest that an attacker must use a brute force attack to guess the session key $k^{(i)}$ and completely break the protocol, which requires as many as $16! * 2$ guesses, for the 128-bit example.

We demonstrate two types of attacks against Protocol 1. First is an active attack based on key sequence cycles that obtains the value $x^{(i)} \oplus k^{(0)}$ and can successfully impersonate a tag after observing an average of 70 transactions. The second attack is independent of key cycles and can fully recover $k^{(0)}$ in 1092 expected guesses.

2.1 Implementation

We implemented the 128-bit key length example from [3] and generated 1,000 sessions with 10,000 transactions per session, i.e. we generated 1,000 different $k^{(0)}$ values and permuted each one 10,000 times. Figure 2 shows that the session keys $k^{(1)}, \ldots, k^{(i)}, \ldots, k^{(10,000)}$ cycle after an average of 68 permutations. That is, the permutation resulted in a repeating pattern of session keys after an average of 68 sessions. For a cycle period of c,

$k^{(i)} = k^{(i+c)}$. The average cycle period is 2, meaning that $k^{(i)} = k^{(i+2)}$. The minimum cycle period was 1, which occurred in 31.9% of our results. All of the observed keys eventually repeat, and the maximum cycle period was 36, which occurred in only 1 out of the 1,000 sessions.

Session key cycles lead to our first attack, which allows an active adversary to successfully impersonate a tag. We also developed an active and passive attack that, independent of session key cycles, allows an adversary to gain full knowledge of $k^{(0)}$.

2.2 Repeated Keys Attack

An active adversary, Mallory, can learn $k^{(i)} \oplus k^{(0)}$ after observing one challenge/response pair. As shown in Figure 1, Mallory learns $a^{(i)} = x^{(i)} \oplus k^{(i)}$ and $b^{(i)} = x^{(i)} \oplus k^{(0)}$, and can calculate their bitwise difference to learn $k^{(i)} \oplus k^{(0)}$. He builds a table with $k^{(i)} \oplus k^{(0)}, k^{(i+1)} \oplus k^{(0)}, k^{(i+2)} \oplus k^{(0)}, \ldots$ rows. Two rows will have the same value when the session key repeats, allowing the attacker to determine the key cycle period. Without loss of generality, assume that the key cycle period is 2. Thus, $k^{(i-2)} = k^{(i)}$. As Figure 1 shows, when the reader sends $x^{(i)} \oplus k^{(i)}$, the attacker can calculate $(x^{(i)} \oplus k^{(i)}) \oplus (k^{(i-2)} \oplus k^{(0)}) = x^{(i)} \oplus k^{(0)}$. This forms a valid response which the attacker can then broadcast to the reader, thus successfully impersonating a valid tag.
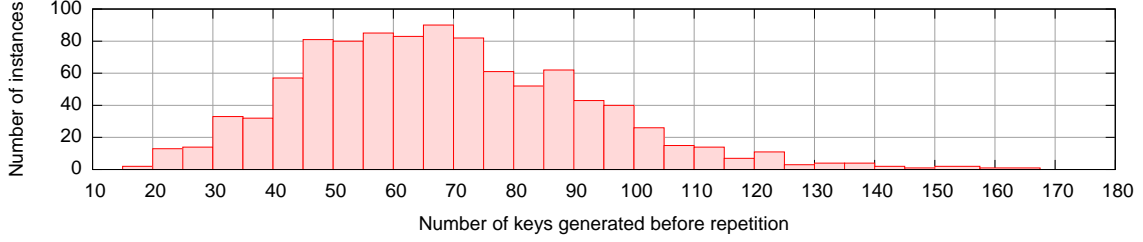
Figure 2. This histogram shows that keys repeat much earlier than the $16! * 2$ theoretical maximum. 1,000 instances of the VB protocol 1 with random $K^{(0)}$'s execute until a key repeats. A key repeats on average after 68 transactions between a tag and reader.

In 68.8% of the sessions we generated, the key cycle period is 2 or less. If the attacker begins eavesdropping with the first transaction between the reader and tag, he can detect a repeated key cycle and impersonate a tag after 70 transactions. If the adversary begins eavesdropping after 68 transactions, then, on average, he can impersonate the tag after observing just 3 transactions.

2.3   Nibble Attack

Passive attack: [3] gives an example using 128-bit key lengths. The permutation $\Pi(k^{(i)}) = k^{(i+1)}$ is defined as follows. First cut each byte of $k^{(i)}$ in half to obtain two nibbles. The left nibbles $k_{0,L}^{(i)}, k_{1,L}^{(i)}, \ldots, k_{15,L}^{(i)}$ form $k_L^{(i)}$ and the right nibbles $k_{0,R}^{(i)}, k_{1,R}^{(i)}, \ldots, k_{15,R}^{(i)}$ form $k_R^{(i)}$. $k_R^{(i+1)}$ is formed by swapping the 0th and $k_{0,L}^{(i)}{}^{th}$, 1st and $k_{1,L}^{(i)}{}^{th}$, $\ldots$, 15th and $k_{15,L}^{(i)}{}^{th}$ elements of $k_R^{(i+1)}$. $k_L^{(i+1)}$ is formed in a similar way using $k_R^{(i)}$.

Observe that if $k_{0,L}^{(i)} = 0$, then the first four bits of $k^{(i)}$ are equal to 0. The 0th and the $k_{0,L}^{(i)}{}^{th}$ elements of $k_{0,R}^{(i)}$ are switched. Hence, $k_{0,R}^{(i)} = k_{0,R}^{(i+1)}$ and the permutation Π resulted in no change to $k_{0,R}^{(i)}$, and thus we know that $k_{0,L}^{(i)} = 0$. This event will happen for roughly a $1/16^{th}$-fraction of values i. Knowing $k_{0,L}^{(i)} = 0$, we can compute $x_{0,L}^{(i)}$ and therefore $k_{0,L}^{(0)}$.

As noted in [3], it is possible for a passive adver-

sary to learn $k^{(i)} \oplus k^{(i+1)}$ after observing two consecutive runs of the protocol. Mallory constructs a table as in Figure 3 and looks at the Indicator column for '0000' in the second nibble. When this occurs, he knows that $k_{0,R}^{(i)} = k_{0,R}^{(i+1)}$ because their bitwise difference is '0000.' Thus, he also knows that $k_{0,L}^{(i)} = 0$. He can use column 1 to calculate $x_{0,L}^{(i)}$ and then use column 2 to determine $k_{0,L}^{(0)}$.

Using similar reasoning, he can find rows in the Indicator column where the fourth nibble is '0000,' which indicates that $k_{1,R}^{(i)} = k_{1,R}^{(i+1)}$. Thus $k_{1,L}^{(i)} = 1$ and he can use the table to calculate $k_{1,L}^{(0)}$. We can use this reasoning to find all nibbles of $k_L^{(0)}$ and $k_R^{(0)}$ and learn the full value of $k^{(0)}$, thus breaking the scheme completely.

Active attack: Our passive attack algorithm can also be employed as an active attack. Mallory first sends the tag a string of $0's$. When a tag receives a challenge $a^{(i)}$, it always responds with $a^{(i)} \oplus k^{(i)} \oplus k^{(0)}$. Thus Mallory will learn $k^{(i)} \oplus k^{(0)}$ by sending a challenge of all $0's$ to the tag. He can continue to sends challenges of all $0's$ to learn $k^{(i+1)} \oplus k^{(0)}, k^{(i+2)} \oplus k^{(0)}$, etc. and construct a table similar to Figure 3. The same analysis from the passive attack can be employed to determine the full value of $k^{(0)}$.

Remark   We note that there are cases in which a nibble is swapped *twice*, such that $k_{0,R}^{(i)} = k_{0,R}^{(i+1)}$

3

$R \to T$	$T \to R$	Leak	Indicator
$x^{(i)} \oplus k^{(i)}$	$x^{(i)} \oplus k^{(0)}$	$k^{(i)} \oplus k^{(0)}$	$k^{(i-1)} \oplus k^{(i)}$
$x^{(i+1)} \oplus k^{(i+1)}$	$x^{(i+1)} \oplus k^{(0)}$	$k^{(i+1)} \oplus k^{(0)}$	$k^{(i)} \oplus k^{(i+1)}$
$x^{(i+2)} \oplus k^{(i+2)}$	$x^{(i+2)} \oplus k^{(0)}$	$k^{(i+2)} \oplus k^{(0)}$	$k^{(i+1)} \oplus k^{(i+2)}$

Figure 3. Information Leaked by Protocol 1. The first and second columns are the observed challenge and response, respectively. The Leak column is the bitwise difference between the first two columns, and the Indicator column is the bitwise difference between rows in the Leak column. When our algorithm finds a '0000' nibble in the Indicator column, it combines this with information from the Leak column to calculate a nibble of $k^{(0)}$.
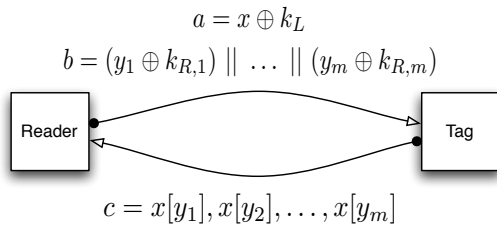
and $k_{0,L}^{(i)} \neq 0$. Thus, Mallory needs to find two cases with a '0000' nibble. If the values calculated for $k_{0,L}^{(0)}$ agree, then this is the correct value with high probability. Otherwise, he must find a third nibble and take the majority value. A false positive occurs in $\frac{1}{15}$ cases, because the '0000' actually occurs in one of the other 15 positions. Mallory must find two '0000' nibbles (16 expected trials to find each) and find a third in $\frac{1}{15} + \frac{1}{15}$ of the cases. This results in $32 * (16 + 16 + 16 * (\frac{1}{15} + \frac{1}{15})) = 1092$ expected trials to fully recover all 32 nibbles of $k^{(0)}$.

3   Vajda and Buttyán Protocol 2

Protocol 2 is a challenge-response protocol in which the tag and reader share two secrets, k_L and k_R.

To construct a challenge, the reader selects two bitstrings x and y uniformly at random. The reader transmits $a = x \oplus k_L$ and $b = y \oplus k_R$ to the tag. Because x and y are random, a and b are also random. In an information-theoretic sense, the pair (a, b) reveals nothing about the secrets k_L and k_R.

The secrets k_L and k_R effectively act as "masks" to conceal the challenge values x and y. The tag uses its knowledge of k_L and k_R to extract $x =$ $a \oplus k_L$ and $y = b \oplus k_R$. The tag then responds to the reader with selected portions of x indexed by y, as detailed below. Knowing x and y, the reader can verify the correctness of the tag's response.

While a challenge alone in this protocol leaks no information, a challenge-response pair *does* leak a considerable amount. Vajda and Buttyán note this leakage, but hypothesize that it is about one bit per protocol invocation. They suggest, therefore, that their challenge-response protocol may be suitable for practical scenarios in which hundreds of accesses to a tag are impractical for an attacker.

We demonstrate an active attack against Protocol 2 that recovers k_L and k_R almost optimally, in the sense that the attack extracts nearly one bit of information from every bit emitted by the tag. In other words, the security of Protocol 2 is nearly no better than that of a protocol in which the tag directly reveals a portion of its key in response to a challenge.

Protocol details: Let l and m be security parameters. The secret k_L has bit-length l, a power of 2. The other secret, k_R, has bit-length $m \log_2 l$. Let $k_R = k_{R,1} \parallel \ldots \parallel k_{R,m}$, i.e., we partition the secret into m substrings, each of bit-length $\log_2 l$.

As we have explained, x and y are random bit strings. By analogy with our notation for k_R, let $y = y_1 \parallel \ldots \parallel y_m$. A challenge consists of a pair $(a = x \oplus k_L, b = y \oplus k_R)$. The response of the tag comprises selected bits of x; the tag determines which bits of x to return to the reader by treating $y_1, \ldots y_m$ as indices into x. Let $x[i]$ denote the ith bit of x for $0 \leq i \leq \log_2 l$ (with either big-endian or little-endian notation). See Figure 4 for a concise protocol specification.

$$a = x \oplus k_L$$
$$b = (y_1 \oplus k_{R,1}) \parallel \ldots \parallel (y_m \oplus k_{R,m})$$

Reader → Tag

$$c = x[y_1], x[y_2], \ldots, x[y_m]$$

Figure 4. VB protocol 2

Overview of active attack: Vajda and Buttyán describe an active attack involving l queries to the

4

tag that recovers k_L. We refer the reader to [3] for details. As an example, they consider $l = 256$ and $m = 16$. They hypothesize that an active attacker requires at leastS 256 queries to break their scheme. We show that considerably fewer queries suffice.

The active attack that we describe first recovers k_R in $\log_2 l + 1$ queries—9 queries for the suggested parameters $l = 256$ and $m = 16$. The attack then fully recovers k_L with at most $\lceil l/m \rceil$ additional queries—i.e., 16 queries for the suggested parameters in [3], amounting to a total of 25 queries for the full attack.

The attack is nearly optimal in the following sense. The total bit length of the shared secrets k_R and k_L is $D = l + m \log_2 l$, while our attack involves a total bit output from the tag of $(\lceil l/m \rceil + \log_2 l + 1)m \leq D + 2m$ bits. Viewed another way, our attack is optimal to within two queries—and only one query when l is divisible by m. (The attack could be further optimized somewhat, but the gains would be small, of course.)

Attack details: Let us denote by $a^{(j)}, b^{(j)}$, and $c^{(j)}$ the protocol values in the jth query, for $j = 0, 1, \ldots, \log_2 l$. Let $c^{(j)}[i]$ denote the ith bit of the tag response.

The attack is as follows. Let $j' = \log_2 l - j$. We construct the vector $a^{(j)}$ as a sequence of $2^{j'}$ '0' bits, followed by $2^{j'}$ '1' bits, then $2^{j'}$ '0' bits, etc., up to the full length of l bits. In other words, we let $a^{(0)} = 00 \ldots 00$, i.e., the all-0s string. We let $a^{(1)} = 00 \ldots 0011 \ldots 11$, i.e., the first half consists of 0s, then second half of 1s. The final query, $a^{(\log_2 l)}$, consists of alternating '0' and '1' bits.

For all j, we let $b^{(j)} = \vec{0}$, i.e., b is a string of 0 bits. (This is a matter of convenience. It is easy to modify the attack such that b is any desired value in any query.) In query q, we challenge the tag with the pair $(a^{(j)}, b^{(j)})$.

Since $b^{(j)} = \vec{0}$, for any i, we have $y_i = k_{R,i}$. Therefore, $c[i] = x[k_{R,i}]$. Now observe that for any $0 \leq i \leq \log_2 l$, if the leading bit $k_{R,i}[0] = 0$, then $c^{(0)}[i] = c^{(1)}[i]$, since $k_{R,i}$ indexes the first half of the vector a, which is constant across the 0th and 1st queries. Otherwise $c^{(0)}[i] \neq c^{(1)}[i]$. Similarly, if $k_{R,i}[1] = 0$, then we observe $c^{(0)}[i] = c^{(2)}[i]$; otherwise $c^{(0)}[i] \neq c^{(2)}[i]$. Similar comparisons across queries reveal the remaining bits of $k_{R,i}$. Thus, for any i, $\log_2 l + 1$ queries suffice to recover $k_{R,i}$ in its entirety. Furthermore, we may recover $k_{R,i}$ for every $1 \leq i \leq m$ independently in parallel.

Hence, with $\log_2 l + 1$ queries, we may recover k_R completely.

With knowledge of k_R, we can quickly recover k_L. Suppose (a, b, c) is a given challenge-response tuple. Using k_R, we can create a value b that corresponds to any sequence of indices $y_1, \ldots, y_m$ we desire. We know that $c[i] = x[y_i]$. Therefore, $c[i] = a[y_i] \oplus k_L[y_i]$, and hence $k_L[y_i] = a[y_i] \oplus c[i]$. In other words, by setting b as desired and using a random vector a, we may recover any m desired bits in k_L. Thus, $\lceil l/m \rceil$ queries suffices to recover all of x_L.

Remark: The authors propose a method of strengthening their scheme by using linear combinations of overlapping sets of y to select bits from x. Our hypothesis is that any such scheme would still not provide adequate cryptographic strength for most practical settings.

4   Conclusion

The Vajda and Buttyán protocols 1 and 2 have inherent weaknesses that render them inadequate for tag authentication. With few resources, an attacker can determine the session keys for both of the protocols, breaking the schemes completely.

References

[1] Henri Gilbert Cme Berbain and Alexander Maximov. Cryptanalysis of grain. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/019, 2006. http://www.ecrypt.eu.org/stream.

[2] Thomas Johansson Martin Hell and Willi Meier. Grain - a stream cipher for constrained environments. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/010, 2005. http://www.ecrypt.eu.org/stream.

[3] István Vajda and Levente Buttyán. Lightweight authentication protocols for low-cost RFID tags. In *Second Workshop on Security in Ubiquitous Computing – Ubicomp 2003*, Seattle, WA, USA, October 2003.