Abstract—Tian et al. proposed a novel ultralightweight RFID mutual authentication protocol [4] that has recently been analyzed in [1], [2], [5]. In this letter, we first propose a desynchronization attack that succeeds with probability almost 1, which improves upon the 0.25 given by the attack in [1]. We also show that the bad properties of the proposed permutation function can be exploited to disclose several bits of the tag’s secret (rather than just one bit as in [2]), which increases the power of a traceability attack. Finally, we show how to extend the above attack to run a full disclosure attack, which requires to eavesdrop less protocol runs than the attack described in [5] (i.e., $192 << 2^{30}$).

Index Terms—RFID, Authentication, Attacks.

I. INTRODUCTION

Tian et al. recently proposed a permutation based mutual authentication protocol called RAPP [4]. In this scheme, tags only use three simple operations: bitwise XOR, left rotation and a very lightweight permutation function $Per(·)$, defined as:

Definition: Let $(A)_i$ denote the $i^{th}$ bit of $A$, $x = (x_1 || x_2) \cdots || (x_l)$ and $y = (y_1 || y_2) \cdots || (y_l)$, where $(x_i), (y_i) \in \{0, 1\}$ and $\parallel$ denotes concatenation. Assume too that $(y)_1 = (y)_k_2 = \cdots = (y)_k_m = 1$ and $(y)_k_{m+1} = (y)_k_{m+2} = \cdots = (y)_k_l = 0$, where $1 \leq k_1 < k_2 < \cdots < k_m \leq l$ and $1 \leq k_{m+1} < k_{m+2} < \cdots < k_l \leq l$. Then the permutation $x$ based on $y$, denoted as $Per(x, y)$, is as follows: $Per(x, y) = (x)_{k_1} || (x)_{k_2} \cdots || (x)_{k_m} || (x)_{k_{m+1}} || (x)_{k_{m+2}} \cdots || (x)_{k_l}$.

The protocol is sketched in Fig. 1. To prevent desynchronization attacks (e.g., if the adversary blocks the last message), the reader keeps two sets of the tuple $\{IDS^x, K^x_1, K^x_2, K^x_3\}$ for $x = old$ (previous values) and $x = new$ (updated values).

Contributions: RAPP is claimed to be resistant against common attacks, including desynchronization and traceability. In this letter, we first propose a desynchronization attack that is independent of the mechanism used to build the messages and has a success probability of almost 1, which improves a 75% the success probability of the previous known attack [1]. Secondly, we introduce a traceability attack that exploits the weak properties of $Per(·)$. Our attack is much more powerful than the one described in [2], since we disclose several bits of the $ID$.

Nasour Bagheri is with the Department of Electrical Engineering, Shahid Rajaee Teachers Training University, Tehran, Iran.

Masoumeh Safkhani is with the Department of Electrical Engineering, Tehran, Iran.

Pedro Peris-Lopez and Juan E. Tapiador are with COSEC Lab, Universidad Carlos III de Madrid, Spain.
instead of just one. The attack can be extended to disclose
the whole ID (as in [5]) at the expense of requiring
various protocol runs.

II. DESYNCHRONIZATION ATTACK ON RAPP

In this attack, the adversary misleads both the tag and
the reader to update their common values to different
values. The attack is based on the observation that the two
random numbers used in the protocol are both generated
by the reader. In a first step, the adversary A eavesdrops
one protocol run between the reader \( R_i \) and the tag \( T_i \),
stores the exchanged values, and blocks \( D \) and \( E \) to
prevent \( T_i \) from updating its internal secret. Note that
the captured values are:

\[
A = \text{Per}(K_2, K_1) \oplus n_1, \\
B = \text{Per}(K_1 \oplus K_2, \text{Rot}(n_1, n_1)) \oplus \text{Per}(n_1, K_1), \\
C = \text{Per}(n_1 \oplus K_1, n_2 \oplus K_2) \oplus ID, \quad D = \text{Per}(K_3, K_2) \oplus n_2 \\
\text{and } E = \text{Per}(K_3, \text{Rot}(n_2, n_2)) \oplus \text{Per}(n_1, K_3 \oplus K_2).
\]

After that, A waits until a legitimate reader initiates
a new session with \( T_i \). In this session \( R_i \) sends
\( A' = \text{Per}(K_2, K_1) \oplus n_1' \) and \( B' = \text{Per}(K_1 \oplus \\
K_2, \text{Rot}(n_1', n_1')) \oplus \text{Per}(n_1', K_1) \) to \( T_i \) and \( T_i \) sends
\( C' = \text{Per}(n_1' \oplus K_1, n_1' \oplus K_3) \oplus ID \) to \( R_i \). In
response, \( R_i \) computes \( D' = \text{Per}(K_3, K_2) \oplus n_2' \) and
\( E' = \text{Per}(K_3, \text{Rot}(n_2', n_2')) \oplus \text{Per}(n_1', K_3 \oplus K_2) \) and
sends \( D' \) and \( E' \) to \( T_i \), which are both blocked by A.
Thus, the tag does not execute the updating mechanism
but the reader does it. In particular, \( R_i \) gets:

\[
IDS^{old} = IDS, K_1^{old} = K_1, K_2^{old} = K_2, K_3^{old} = K_3, \\
IDS^{new} = \text{Per}(IDS, n_1' \oplus n_2') \oplus K_1 \oplus K_2 \oplus K_3, \\
K_1^{new} = \text{Per}(K_1, n_1') \oplus K_2, K_2^{new} = \text{Per}(K_2, n_2') \oplus K_1, \\
K_3^{new} = \text{Per}(K_3, n_1' \oplus n_2') \oplus IDS.
\]

whereas \( T_i \) does not update its internal values and these
remain as \( \{IDS, K_1, K_2, K_3\} \).

Finally, A supplants a legitimate \( R_i \) by replying old
values. A sends \textit{Hello} message to \( T_i \) and \( T_i \) replies with
its pseudonym \( IDS \). Then, A sends \( A \) and \( B \) (captured
in the first step) to \( T_i \). Then \( T_i \) extracts \( n_1 \) from \( A \), verifies
\( B \), and sends \( C \) to \( R_i \) (i.e., to \( A \)). Upon receiving \( C \), A
replies with the values \( D \) and \( E \) eavesdropped in step 1.
Finally, \( T_i \) extracts \( n_2 \) from \( D \), verifies \( E \), and updates
its secrets:

\[
IDS^* = \text{Per}(IDS, n_1 \oplus n_2) \oplus K_1 \oplus K_2 \oplus K_3, \\
K_1^* = \text{Per}(K_1, n_1) \oplus K_2, K_2^* = \text{Per}(K_2, n_2) \oplus K_1, \\
K_3^* = \text{Per}(K_3, n_1 \oplus n_2) \oplus IDS, \\
IDS = IDS^*, K_1 = K_1^*, K_2 = K_2^*, K_3 = K_3^*.
\]

Thus, given that \( n_1, n_1', n_2 \) and \( n_2' \) are chosen randomly,
by the end of the attack the reader keeps two records for
the tag, none of which match the values stored in the tag
(see Table I for details). Note that the success probability
for this attack is \( 1 - 2^{-2l} \), where \( l \) is the bit length of
the random numbers. For instance, for \( l = 96 \) as in [2],
the success probability is almost 1.

III. TRACEABILITY ATTACK ON RAPP

In RAPP, a desynchronized tag never updates its \( IDS \).
Therefore, such a constant value can be used to carry out
a straightforward traceability attack. We next describe a
different attack based on the uniqueness of ID and some
weak properties of the permutation function \( \text{Per}(\cdot) \):

- **Observation 1**: \( \text{Per}(x, y) = (P)_1 || \ldots || (P)_t \) is
  not a perfect permutation. More precisely, \( (x)_1 \) will
  be assigned to either \( (P)_1 \) or \( (P)_2 \); \( (x)_2 \) will
  be assigned to either \( (P)_1 \) (or \( (P)_2 \) \) or \( (P)_3 \) and
  so on.

- **Observation 2**: Given the location of \( (x)_i \) in
  \( \text{Per}(x, y) \), there are only two possible choices for
  the location of \( (x)_{i+1} \).

- **Observation 3**: Given \( y^1, y^2, \ldots, y^j \) and \( P^j = 
  \text{Per}(x, y^j) \), for \( j \leq i \), if \( (y^1)_1 = (y^2)_1 = \ldots = 
  (y^j)_1 \), either \( (P^1)_1 = (P^2)_1 = \ldots = (P^j)_1 \) or
  \( (P^1)_1 = (P^2)_1 = \ldots = (P^j)_1 \).

We next show how to obtain \( \{IDS\}_{i=(i-1)} \oplus 
\text{ID} = \{IDS\}_{i=1}^{1/2} \). The adversary A eavesdrops t
protocol runs, stores the exchanged values and blocks message
\( C \) to abort the protocol. The values captured in the
\( j \)-th round are \( A^j = \text{Per}(K_2, K_1) \oplus n_1', B^j = 
\text{Per}(K_1 \oplus 
K_2, \text{Rot}(n_1', n_1')) \oplus \text{Per}(n_1', K_1), \) and \( C^j = \text{Per}(n_1' \oplus 
K_1, n_1' \oplus K_3) \oplus ID \).

It is easy to see that \( A^j \oplus A^i = n_1' \oplus n_1^i \), for
\( j, f \leq t \). Hence, it is possible to determine whether \( (n_1^i)_m = 
(n_1')_m \) for \( 1 \leq m \leq t \) and group \( n_1^i \), \( n_1' \) into
two groups, denoted by \( G_1 \) and \( G_2 \), respectively, where any
entry in a group holds the same value in its \( m \)-th bit.
Now, consider the case where \( G_1 \) and \( G_2 \) are grouped
by the Least Significant Bit (LSB) of each entry. Then,
for any \( n_1^i, n_1' \in G_1 \) one can state that \( (n_1')_1 = (n_1^i)_1 \)
and for \( n_1^i \in G_1 \) and any \( n_1' \in G_2 \) we have \( (n_1')_1 \oplus 
(n_1^i)_1 = 1 \). A can also group the values of \( C \) based
on how \( n_1 \) has been grouped. In addition, given that in
each group the LSB of all \( n_1 \)s are the same, \( K_3 \) remains
constant (non-updating condition by the blockage of $C$), and $ID$ is fixed, then either LSB or, equivalently, the Most Significant Bit (MSB) of all $C$s in that group must be the same. For instance, if for all $C^j \in G1$ the LSBs are the same and for all $C^j \in G2$ the MSBs are the same then $\mathcal{A}$ deduces that for any $n_1^j \in G1$, $(n_1^j \oplus K_3)_j = 1$ and $(C^j)_1 = (n_1^j)_1 \oplus (K_1)_1 \oplus (ID)_1$ and for any $n_1^j \in G2$, $(n_1^j \oplus K_3)_j = 0$ and $(C^j)_1 = (n_1^j)_1 \oplus (K_1)_1 \oplus (ID)_1$. In this case, it is easy to deduce that for any $C^j \in G1$ and any $C^j \in G2$, $(C^j)_1 \oplus (C^j)_2 = 1 \oplus (ID)_1 \oplus (ID)_2$. Otherwise, for all $C^j \in G1$ the MSBs are the same and for all $C^j \in G2$ the LSBs are the same, then $\mathcal{A}$ deduces that for any $n_1^j \in G1$, $(n_1^j \oplus K_3)_j = 0$ and $(C^j)_2 = (n_1^j)_2 \oplus (K_1)_2 \oplus (ID)_2$ and for any $n_1^j \in G2$, $(n_1^j \oplus K_3)_j = 1$ and $(C^j)_2 = (n_1^j)_2 \oplus (K_1)_2 \oplus (ID)_2$. In this case, for any $C^j \in G1$ and any $C^j \in G2$, we have $(C^j)_2 \oplus (C^j)_1 = 1 \oplus (ID)_2 \oplus (ID)_1$. Therefore, in both of the above cases, $\mathcal{A}$ reveals one bit of information about the $ID$ (in particular, the XOR of two of its bits), which remains unchanged during all the tag’s life.

Given the location of $(n_1^j)_1 \oplus (K_1)_1$ for $C^j \in G1$, e.g., $(C^j)_1$, there are two possible locations for $(n_1^j)_2 \oplus (K_1)_2$ in $C^j$, e.g., $(C^j)_2$ or $(C^j)_3$. Following the given procedure, we can group the values of $n_1$ and $C$ in $G1$ based on the second bits of $n_1$s and determine $(ID)_2 \oplus (ID)_1$. This process can be repeated to retrieve $(ID)_3 \oplus (ID)_2 \oplus (ID)_1$ based on the grouping of the $n_1$s and $C$s in $G2$. These new information increases the adversary’s probability of tracing the tag. The adversary fails to determine the correct value of $ID_1 \oplus ID_2$ if all $(n_1^j)_1$, for $j \leq t$, has the same value, which occurs with a probability of $2^{-t}$, or for any $C^j \in G1$ and $C^j \in G2$ we have $(C^j)_1 = (C^j)_2$, which happens with a probability of $2^{-|G1|} \times 2^{-|G2|} = 2^{-t}$, where $|G1|$ and $|G2|$ denotes the cardinality of $G1$ and $G2$ respectively. Hence, the total success probability is $(1 - 2^{-t})^2$.

We note here that formal traceability models (e.g., [3]) require the disclosure of just one bit. Nevertheless, more bits are needed to trace a constellation of tags. Our attack facilitates this at the expense of requiring several protocol runs. For instance, if the attacker wants to disclose 3 bits, the success probability is approximately $(1 - 2^{-t})(1 - 2^{-t/2})^2$. For example, for $t = 8$ the success probability is 0.77, which is greater than 0.5. For the given attack, the attack complexity increments exponentially with the number of bits recovered. However, a more efficient disclosure is possible, as described next.

### IV. Disclosure Attack

From Section III, we can generally state that to determine $(n_1^j)_f \oplus (K_3)_f$, given the location of $(n_1^j)_f \oplus (K_3)_f$, there are two possible locations for $(n_1^j)_f \oplus (K_3)_f$, i.e., $m$ for $(n_1^j)_f \oplus (K_3)_f = 0$ and $n$ for $(n_1^j)_f \oplus (K_3)_f = 1$. An adversary $\mathcal{A}$ can calculate such $m$ and $n$ for all the stored records. Since $t > l$, there are at least $t-l$ collisions for those cases where $(n_1^j)_f \oplus (K_3)_f = 0$ and also $t-l$ collisions for those where $(n_1^j)_f \oplus (K_3)_f = 1$, with $j \leq t$. Assume that for $(n_1^j)_f \oplus (K_3)_f = 1$ and $(n_1^j)_f \oplus (K_3)_f = 0$, $(n_1^j)_f \oplus (K_3)_f$ and $(n_1^j)_f \oplus (K_3)_f$ are grouped into $(C^j)_m$ and $(C^e)_m$, respectively. Then, for $(n_1^j)_f = (n_1^j)_f$ we have $(C^j)_m = (C^e)_m$ because $(C^j)_m = (n_1^j)_f \oplus (K_3)_f \oplus (ID)_m$ and $(C^e)_m = (n_1^j)_f \oplus (K_3)_f \oplus (ID)_m$. This may be used to verify whether the assumption is correct or not. Thus, to determine $(n_1^j)_f \oplus (K_3)_f$, given $(n_1^j)_f \oplus (K_3)_f$, $\mathcal{A}$ groups the records based on the $f^{th}$ bit of $n_1$s, i.e., $G1$ and $G2$, where all entries in each group hold the same value in that bit of $n_1$. Now, it is obvious that for any $n_1^j \in G1$ and $n_1^j \in G1$ one can state that $(n_1^j)_f \oplus (K_3)_f = (n_1^j)_f \oplus (K_3)_f$ and $(n_1^j)_f \oplus (K_3)_f = (n_1^j)_f \oplus (K_3)_f$. (The same is applicable for entries in $G2$). Next $\mathcal{A}$ guesses $(n_1^j)_f \oplus (K_3)_f = 1$ for entries in $G1$ and checks its correctness by verifying the collision in the expected location of $C$s, e.g., $(n_1^j)_f = (n_1^j)_f$ and also the adversary expects to have $(C^j)_m = (C^e)_m$ as previously
explained. If \((n^l_1)f \oplus (K_3)_f = 1\) passes the verification for \(G1\), then \((n^l_1)f \oplus (K_3)_f = 0\) must pass the verification for entries in \(G2\). However, it is possible that a wrong guess for \((n^l_1)f \oplus (K_3)_f\) passes all verifications, with a probability upper bounded by \(2^{l-t}\), where the probability of determining the correct value is \(1-2^{l-t}\). Summarizing, the success probability of extracting a correct value for \(K_3\) is \(2^{l-t}\), and \(1 \leq j \leq t\), is \((1-2^{l-t})^l \approx 1 - l \cdot 2^{l-t}\). For instance, if \(l = 64\) and \(t = 2l\), then it is \(1 - 2^{-58} \approx 1\).

By the end of the above process, for any \(n^l_1\) we have \(l\) equations of the form \((K_3 \oplus n^l_1)f\), for \(f \leq l\). Moreover, for any bit of \(C^j\) we have \((C^j)_m = (n^l_1)f \oplus (K_1)f \oplus (ID)_m\). These equations are not all linearly independent – collided values to the same position in a group generate the same equation. Hence, assuming that there are \(n\) possible locations for \((n^l_1)f \oplus (K_1)_f\)s that are confirmed, \(A\) has \(n\) distinct equations of the form \(\{n^l_1\}_f \oplus (K_1)f \oplus (ID)_m\), \(\ldots\), \(\{n^l_1\}_f \oplus (K_1)f \oplus (ID)_m\), where \(1 \leq m_1 < \ldots < m_n \leq l\). For instance, if \(A\) determines \((K_3 \oplus n^l_1)_2\), she has the following information: \(\{(n^l_1)_2 \oplus (K_1)_2 \oplus (ID)_1\}, \{(n^l_1)_2 \oplus (K_1)_2 \oplus (ID)_1\}, \ldots\). The above equations can be used to compute \(\{n^l_1\}_1 \oplus (K_1)_1 \oplus (ID)_1\), \(\{n^l_1\}_2 \oplus (K_1)_2 \oplus (ID)_2\), \ldots. Therefore, when \(A\) discloses \((K_3 \oplus n^l_1)_1\), she is expected to have enough information to reveal \((n^l_1 \oplus K_1)_f \oplus (ID)_m\) for any given \(e \leq t\) and \(f, m \leq l\).

The adversary can also retrieve \(K_1\) as follows. If we assume that \((K_1)_1 = 1\) then \((A')_1 = (K_2)_1 \oplus (n^l_1)_1\). On the other hand, \(A\) already knows the value of \((n^l_1)_1 \oplus (K_3)_1\). So it can determine \((K_2)_1 \oplus (K_3)_1\) which should be the same for all \(j \leq t\). If all the values were equal then the guess is correct, with a probability of \(1 - 2^{-t}\), and \((K_1)_1 = 1\); otherwise \((K_1)_1 = 0\). In addition, \(A\) discloses another bit of information of the form \((K_2)_1 \oplus (n^l_1)_1\) or \((K_2)_1 \oplus (n^l_1)_j\). Next, for \(2 \leq m \leq l\), given the location of \((K_2)_m\) in \(A'\), there are two possible locations for \((K_2)_m\), i.e., \(e\) for \((K_1)_2 = 1\) and \(f\) for \((K_1)_2 = 0\) respectively. Assuming \((K_1)_m = 1\) then \((A')_e = (K_2)_m \oplus (n^l_1)_e\). On the other hand, \(A\) already knows the value of \((n^l_1)_e \oplus (K_3)_e\), so she can determine \((K_2)_m \oplus (K_3)_e\) too, which should be the same for all \(1 \leq j \leq t\) when the guessed value for \((K_1)_m\) is correct. If all the values are equal, then the guess is correct with a probability of \(1 - 2^{-t}\), and \((K_1)_m = 1\); otherwise \((K_1)_2 = 0\). This process can be repeated to retrieve all bits in \(K_1\) and auxiliary equations can also be deduced \((K_2)_3 \oplus (K_3)_e\), \ldots, \((K_2)_t \oplus (K_3)_e\), where \(e_j \in \{1, \ldots, l\}\). The success probability of extracting a correct value for \(K_1\) is \((1-2^{-t})^l \approx 1 - l2^{-t}\).

In the next step, the adversary combines the records of the form \((n^l_1 \oplus (K_1)_f \oplus (ID)_m\), for any given \(e \leq t\), \(f \leq l\) and \(K_1\) to determine the following information: \(\{(n^l_1)_1 \oplus (n^l_1)_2, (n^l_1)_1 \oplus (n^l_1)_2, \ldots, (n^l_1)_1 \oplus (n^l_1)_1 \oplus (ID)_1\}\). Then \(A\) guesses \((n^l_1)_1\). Given \((n^l_1)_1\) \(A\) can determine \((n^l_1)_2, \ldots, (n^l_1)_t\) (i.e., she discloses all bits in \(n^l_1\) and \(ID\) from the above equations. Given that \(A\) knows \(K_3 \oplus n^l_1\), she can easily get \(K_3\) now. Then, from the auxiliary equations \((K_2)_1 \oplus (K_3)_e, \ldots, (K_2)_1 \oplus (K_3)_e\), \(A\) discloses \(K_2\) too. Finally, \(A\) uses the stored \(B_3s\) to verify the correctness of the guessed value for \((n^l_1)_1\). If any \(B_3\) does not pass the verification, then \(A\) should flip its guess for \((n^l_1)_1\) and repeat the process to determine \(n^l_1s, ID, K_2\) and \(K_3\). Finally, \(A\) reveals all secrets. Assuming that the adversary has eavesdropped \(t\) sessions, the success probability of the given attack is \((1-2^{-t})^l \cdot (1-2^{-t})^t \approx (1-2^{-l})^t \times (1-2^{-t})\). For example, for \(l = 96\) and \(t = 2 \cdot l\) (which is much less than \(t = 2^{30}\) eavesdropped sessions required in [5]), then the success probability is almost 1.

V. CONCLUSIONS

We have described desynchronization, traceability, and disclosure attacks against RAPP more powerful than those previously published. RAPP’s major contribution is its permutation function, which is clearly insufficient. While the traceability and disclosure attacks depend on the permutation and might be fixed by proposing a stronger permutation, our desynchronization attack works for any permutation, showing that RAPP has major flaws that cannot be fixed just by replacing the permutation.

REFERENCES

2. G. Avoine and X. Carpent. Yet another ultralightweight authentication protocol that is broken. In Workshop on RFID Security – RFIDSec’12, Nijmegen, Netherlands, June 2012.