An Ultra-Efficient Key Recovery Attack on the Lightweight Stream Cipher A2U2

Qi Chai, *Student Member, IEEE,* Xinxin Fan, and Guang Gong, *Senior Member, IEEE*

***Abstract*—In this letter we report on an ultra-efficient key recovery attack under the chosen-plaintext-attack model against the stream cipher A2U2, which is the most lightweight cryptographic primitive (i.e., it costs only 284 GE in hardware implementation) proposed so far for low-cost Radio Frequency Identification (RFID) tags. Our attack can fully recover the secret key of the A2U2 cipher by only querying the A2U2 encryption twice on the victim tag and solving 32 sparse systems of linear equations (where each system has 56 unknowns and around 28 unknowns can be directly obtained without computation) in the worst case, which takes around 0.16 second on a Thinkpad T410 laptop.**

***Index Terms*—Stream Cipher, Key Recovery, RFID**

I. Introduction

Radio Frequency Identification (RFID), which enables automatic remote identification of objects, is one of the most promising technologies in the field of ubiquitous computing. Although RFID technology provides many attractive and exclusive characteristics, the constrained computational and storage capabilities as well as the extremely low manufacture cost of RFID tags have posed a new challenge that goes beyond the traditional cipher design paradigm and stimulates the brand-new design of lightweight stream/block ciphers. Typical examples include PRESENT [1], KATAN/KTANTAN [3], Hummingbird [7], PRINT [9], Grain [8] and WG-7 [10]. Among them, PRESENT and KATAN/KTANTAN are the most promising ones because of their compact hardware implementation[1], satisfactory throughput and widely-accepted security (after extensive cryptanalysis).

More recently, David *et al.* [5] proposed a stream cipher called A2U2, which is the most lightweight cipher proposed so far, in terms of its hardware footprint of 284 GE. Besides its compactness, the throughput of A2U2 is approximately five times greater than that of PRESENT, KTANTAN32 and PRINT. The security analysis of the A2U2 shows: (1) The output sequence of the A2U2 can pass the NIST's statistical tests for pseudorandom number generators; (2) The period and the linear complexity of the output sequence are around 2^{70}; (3) Particular attacks are thwarted since variable number of clock cycles used for the initialization ensures that the cipher outputs different ciphertexts for identical plaintexts.

In this letter, we address the security of the lightweight stream cipher A2U2. Our cryptanalytic results show that the A2U2 is insecure under a simple chosen-plaintext attack, which enables the full key recovery of the A2U2 through executing two encryptions with particular plaintexts of 653 bits and solving 32 sparse systems of linear equations (where each system has 56 unknowns and around 28 unknowns can be directly obtained without computation) with around 0.16 second on a Thinkpad T410 laptop.

The authors are with the Communication Security Lab, Department of Electrical and Computer Engineering, University of Waterloo, ON, N2L 3G1 Canada (e-mail: {q3chai, x5fan, ggong}@uwaterloo.ca).

[1]PRESENT and KTANTAN32 cost 1075 GE and 462 GE in hardware implementation, respectively.

II. A2U2: A Lightweight Stream Cipher for RFID Tags

A. A Short Description of A2U2

As illustrated in Fig. 1, the stream cipher A2U2 is composed of four building blocks: a 7-stage linear feedback shift register (LFSR), a combination of two nonlinear feedback shift registers (NFLRs), a key schedule module, and a filter function. In the following description, we use $+$ to denote an XOR operation, $\odot$ an NAND operation, and $\cdot$ an AND operation.

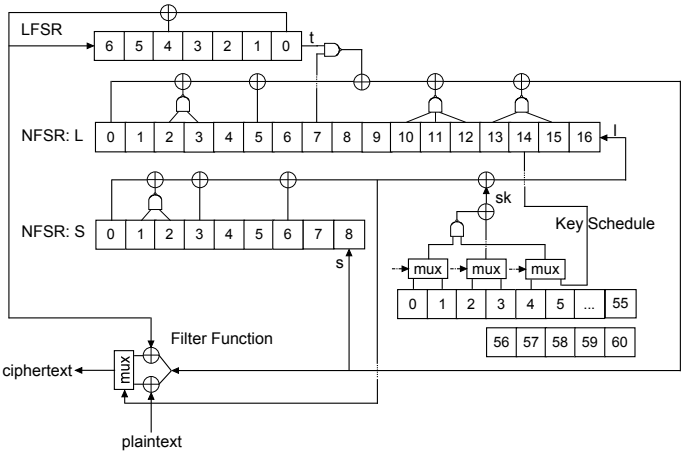

Fig. 1. The Architecture of the Stream Cipher A2U2

After a 61-bit secret key $(k_0, ..., k_{60})$ is burnt into an RFID tag, the tag is capable of encrypting plaintext bits $p_\delta, \ldots, p_n$ to the corresponding ciphertext bits $c_\delta, \ldots, c_n$, where δ is the number of clock cycles required to initialize the internal state of the A2U2 and is determined by the following procedure:

1) The RFID reader and the tag generate and exchange two 32-bit random numbers $RND_R = (a_0, \ldots, a_{31}) \in \mathbb{F}_2^{32}$ and $RND_T = (b_0, \ldots, b_{31}) \in \mathbb{F}_2^{32}$, respectively;
2) The value $(RND_R + RND_T)$ is then loaded into the LFSR and two NFSRs of the A2U2 cipher;
3) Both the LFSR and two NFSRs run δ clock cycles for initialization without any output until the state of the LFSR reaches all ones. From the next clock cycle, the stream cipher A2U2 outputs the ciphertext bits.

To demonstrate our attack we further detail the building blocks of the stream cipher A2U2 below.

The LFSR: The LFSR has seven stages denoted by a binary vector $(t_{i+6}, \ldots, t_i) \in \mathbb{F}_2^7, i = 0, \ldots, n$. Moreover, the following recursive relation holds:

$$t_{i+7} = t_i + t_{i+4}, \text{ for } i = 0, \ldots, n.$$

Note that the generated sequence is an m-sequence [6] with the maximum period $2^7 - 1 = 127$. In the above Step 2, $((a_0, \ldots, a_4) + (b_0, \ldots, b_4) + (k_{56}, \ldots, k_{60}))$ is loaded into $(t_4, \ldots, t_0)$ of the LFSR, while leaving t_5 to be a constant one and t_6 to be a constant zero.

The Two NFSRs: This block borrows part of the design from the lightweight block cipher KATAN/KTANTAN [3]. For $i = 0, \ldots, n$, the feedback functions can be represented as follows:

$$\begin{aligned} s_{i+8} &= l_i + l_{i+2} \odot l_{i+3} + l_{i+5} + l_{i+7} \odot t_{i+6} \\ &\quad + l_{i+10} \odot l_{i+11} \odot l_{i+12} + l_{i+13} \odot l_{i+15}, \\ l_{i+16} &= s_i + s_{i+1} \odot s_{i+2} + s_{i+3} + s_{i+6} + sk_i, \end{aligned} \quad (1)$$

where sk_i is the subkey bit generated by the key schedule.

The Key Schedule: This module derives a sequence of subkeys $(sk_0, \ldots, sk_n)$ from a portion of the secret key $(k_0, ..., k_{55})$, where sk_i is used by the NFSR during the i-th clock cycle and is computed as follows:

$$\begin{aligned} sk_i &= mux(k_{mod(5i,56)}, k_{mod(5i+1,56)}, t_{i+1}) \odot \\ &\quad mux(k_{mod(5i+4,56)}, l_{i+14}, t_{i+5}) + \\ &\quad mux(k_{mod(5i+2,56)}, k_{mod(5i+3,56)}, t_{i+3}), \end{aligned} \quad (2)$$

where $mod(x, y)$ returns x modulo y provided that x, y are non-negative integers and $mux()$ is a multiplexer such that, given $x, y, z \in \{0, 1\}$, $mux(x, y, z) = x$ iff $z = 0$ or $mux(x, y, z) = y$ iff $z = 1$.

The Filter Function: The filter function is essentially a variant of the *shrinking generator* [4], which replaces the XOR operation of keystream bits and plaintext bits in a classical stream cipher. Regardless of its multiplexer-based implementation, the filter function can be simply written as (see Eq. (6) in [5]):

$$\textbf{Case I}: c_i = \begin{cases} s_{i+8} + t_i, & \text{if } l_{i+16} = 0, \\ s_{i+8} + p_i, & \text{if } l_{i+16} = 1, \end{cases} \text{ for } i = \delta, \ldots, n, \quad (3)$$

where p_i is the plaintext bit fed into the A2U2 cipher during the i-th clock cycle, and c_i is the corresponding ciphertext bit. Let us denote the above expression of c_i as "**Case I of $\mathbf{c}_i$**".

It is worthy to point out that we were recently informed by Mohamed Ahmed Abdelraheem, Julia Borghoff and Erik Zenner that the initial intention of the authors of [5] is in fact to construct a multiplexer (as shown below) behaving like a *stop-and-go sequence generator* [2], which is different from the descriptions of A2U2 in [5]. Let us call the below expression of c_i as "**Case II of $\mathbf{c}_i$**". Nevertheless, as shown later, our attack is applicable to both designs.

$$\textbf{Case II}: c_i = \begin{cases} s_{i+8} + t_i, & \text{if } l_{i+16} = 0, \\ s_{i+8} + p_{f(i)}, & \text{if } l_{i+16} = 1, \end{cases} \text{ for } i = \delta, \ldots, \hat{n} \quad (4)$$

where $f(i) = \delta + \sum_{j=\delta+16}^{i+16} l_j, \delta \leq i \leq n$, $f(\delta) = \delta$ and $\hat{n}$ equals the sum of n and the number of inserted bits from the m-sequence.

III. A Lightweight Key Recover Attack

A. Attacker Model

We consider the classical chosen-plaintext attack against the stream cipher A2U2 that is implemented on an RFID tag with a fixed and high-entropy 61-bit secret key burnt inside. An attacker, equipped with a programmable RFID reader, queries the victim tag with a particular random number RND_R and plaintext bits $p_\delta, ..., p_n$. The attacker's goal is to recover the secret key $(k_0, ..., k_{60})$. Note that in our attack, the reader, manipulated by the attacker, can always adaptively choose RND_R to make $(RND_R + RND_T)$ a constant, e.g., $RND_R + RND_T = 0$.

B. Step 1: Recover Sequences of s_{i+8} And l_{i+16}

A cryptographic primitive is only as strong as its weakest module, which is the general principle to break a cryptosystem. We noticed that the filter function in the A2U2 is quite weak, which enables an attacker to easily recover the internal state of the NFSRs by the following steps.

1) At the δ-th clock cycle, the LFSR reaches the all-one state and the A2U2 starts the ciphertext output. As a result, the attacker knows the sequence $(t_\delta, t_{\delta+1}, ..., t_n)$, which is just a repetition of the m-sequence with period 127 as shown in Table I.

TABLE I
One Period of $(t_\delta, t_{\delta+1}, ..., t_n)$

1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0,
1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0.

2) **Case I of $\mathbf{c}_i$**: The attacker first chooses the plaintext bits

$$(p_\delta, \ldots, p_n) = (t_\delta, t_{\delta+1}, \ldots, t_n),$$

and sends them to the RFID tag for encryption. Since p_i and t_i are equal for $i = \delta, \ldots, n$, Eq. (3) now becomes

$$c_i = s_{i+8} + t_i, \text{ for } i = \delta, \ldots, n.$$

Consequently, the variable l_i that controls the multiplexer is nullified and s_{i+8} can be recovered (i.e., both c_i and t_i are known to the attacker). Next, the attacker chooses a new set of plaintext bits

$$(p_\delta, \ldots, p_n) = (1 + t_\delta, 1 + t_{\delta+1}, \ldots, 1 + t_n),$$

and launches another encryption session with the RFID tag. In this case, Eq. (3) becomes

$$c_i = \begin{cases} s_{i+8} + t_i, & \text{if } l_{i+16} = 0 \\ s_{i+8} + t_i + 1, & \text{if } l_{i+16} = 1 \end{cases} \quad \text{for } i = \delta, \ldots, n.$$

Given t_i, s_{i+8} and c_i, the attacker can easily distinguish $l_i = 0$ from $l_i = 1$.

Case II of $\mathbf{c}_i$: The above procedure could be easily transferred to attack c_i in the second case, based on the observation that the output c_i of the multiplexer is a linear function, i.e., either $(s_{i+8} + t_i)$ or $(s_{i+8} + p_{f(i)})$. To launch the attack, the attacker chooses two complimentary plaintexts $(p_\delta, \ldots, p_n)$ and $(p'_\delta, \ldots, p'_n)$ for encryption, i.e., $p_i + p'_i = 1, \delta \le i \le n$. Letting the corresponding ciphertexts be $(c_\delta, \ldots, c_{\hat{n}})$ and $(c'_\delta, \ldots, c'_{\hat{n}})$, we have

$$c_i + c'_i = \begin{cases} 0, & \text{if } l_{i+16} = 0, \\ 1, & \text{if } l_{i+16} = 1, \end{cases} \quad \text{for } i = \delta, \ldots, \hat{n},$$

which reveals the sequence $\{l_{\delta+16}, \ldots, l_{\hat{n}+16}\}$. Consequently, the sequence $\{s_{\delta+8}, \ldots, s_{\hat{n}+8}\}$ can be simply recovered from Eq. (4) once each l_{i+16} is known.

C. Step 2: Recover Internal States of NFSRs and Subkey sk_i

Given s_{i+8} and l_{i+16} ($i \ge \delta$) obtained from the Step 1, the internal states of the two NFSRs are completely exposed to the attacker after $\max(\delta + 9, \delta + 17) = \delta + 17$ clock cycles. Next, the attacker employs the following relation derived from Eq. (1) to recover the subkey bits sk_i for $i = \delta + 17, \ldots, n$,

$$sk_i = s_i + s_{i+1} \odot s_{i+2} + s_{i+3} + s_{i+6} + l_{i+16}.$$

We would like to point out that the attacker is already capable of decrypting any ciphertext with the obtained subkeys $(sk_{\delta+17}, \ldots, sk_n)$. However, the attacker could do even better by fully recovering the secret key as described below.

D. Step 3: Fully Recover Secret Key

Without loss of generality, we fix δ to be a specific integer by guessing, e.g., assume $\delta = 88$ hereafter, and first recover the partial secret key bits $k_0, ..., k_{55}$. To this end, the attacker generates $(n - \delta - 16)$ equations (see below) from Eq. (2) as well as the internal state l_i of the NFSRs, and derives the subkeys sk_i, when $i \ge \delta + 17$.

$$\begin{aligned} k_{22} \cdot l_{119} + k_{23} &= sk_{105} + 1 & k_{26} \cdot k_{30} + k_{29} &= sk_{106} + 1 \\ k_{31} \cdot l_{121} + k_{34} &= sk_{107} + 1 & k_{37} \cdot k_{40} + k_{38} &= sk_{108} + 1 \\ k_{42} \cdot k_{45} + k_{44} &= sk_{109} + 1 & k_{46} \cdot l_{124} + k_{48} &= sk_{110} + 1 \\ k_{52} \cdot l_{125} + k_{53} &= sk_{111} + 1 & k_0 \cdot l_{126} + k_3 &= sk_{112} + 1 \\ k_5 \cdot l_{127} + k_8 &= sk_{113} + 1 & k_{11} \cdot k_{14} + k_{13} &= sk_{114} + 1 \\ & \ldots & & \ldots \end{aligned}$$

Among these equations, approximately half of them are of the type

$$k_x \cdot l_y + k_z = sk_w + 1, \tag{5}$$

where x, y, z and w are integers. Since l_y is known to the attacker, he can select 56 such equations to form a sparse linear system with full rank and solve it to get $(k_0, ..., k_{55})$. Through extensive experiments, we found that when $n = 512 + \delta$, the attacker can obtain 56 linear independent equations out of 249 Eq. (5)-alike equations with probability 1, which implies that in a practical attack scenario the attacker should query the RFID tag with plaintexts of $n = 512 + \delta = 638$ bits (recall that $\delta \le 126$).

To recover the rest key bits $k_{56}, \ldots, k_{60}$, we make use of the fact that δ is indeed the number of clock cycles required to transit the LFSR's state from $(k_{60}, \ldots, k_{56}, 1, 0)$ to $(1, 1, 1, 1, 1, 1, 1)$ or the reverse direction (here we assume $RND_R + RND_T = 0$ for simplicity). In our example, transiting the state $(1, 1, 1, 1, 1, 1, 1)$ reversely for $\delta = 88$ clock cycles gives us $(k_{56}, ..., k_{60}) = (1, 0, 1, 0, 0)$. Due to the uncertainty of δ, the attacker could have 32 possible keys such that the recovered $(k_0, ..., k_{55})$ is a δ-bit shifted version of the right key and $(k_{56}, ..., k_{60})$ is the state determined by δ as listed in Table II. The attacker then utilizes the obtained one plaintext/ciphertext pair to test all 32 key candidates locally for retrieving the correct one.

TABLE II
DETERMINISTIC RELATION BETWEEN δ AND $(k_{56}, \ldots, k_{60})$

$(k_{56}, \ldots, k_{60})$	δ	$(k_{56}, \ldots, k_{60})$	δ	$(k_{56}, \ldots, k_{60})$	δ
$(0,0,0,0,0)$	29	$(0,0,0,0,1)$	91	$(0,0,0,1,0)$	36
$(0,0,0,1,1)$	113	$(0,0,1,0,0)$	26	$(0,0,1,0,1)$	108
$(0,0,1,1,0)$	40	$(0,0,1,1,1)$	75	$(0,1,0,0,0)$	111
$(0,1,0,0,1)$	73	$(0,1,0,1,0)$	96	$(0,1,0,1,1)$	98
$(0,1,1,0,0)$	20	$(0,1,1,0,1)$	54	$(0,1,1,1,0)$	48
$(0,1,1,1,1)$	100	$(1,0,0,0,0)$	59	$(1,0,0,0,1)$	43
$(1,0,0,1,0)$	22	$(1,0,0,1,1)$	66	$(1,0,1,0,0)$	88
$(1,0,1,0,1)$	70	$(1,0,1,1,0)$	56	$(1,0,1,1,1)$	117
$(1,1,0,0,0)$	120	$(1,1,0,0,1)$	78	$(1,1,0,1,0)$	50
$(1,1,0,1,1)$	10	$(1,1,1,0,0)$	14	$(1,1,1,0,1)$	83
$(1,1,1,1,0)$	102	$(1,1,1,1,1)$	126		

E. Computational Complexity of the Attack

Our attack is an ultra-efficient chosen-plaintext attack in terms of the computational overhead. Table III summarizes the computational complexity of the proposed attack.

TABLE III
COMPUTATIONAL COMPLEXITY OF THE PROPOSED ATTACK

Recovery	Bits	Computation Cost
s_i	$i = \delta + 9, \ldots, n$	one encryption of n bits
l_i	$i = \delta + 17, \ldots, n$	one encryption of n bits
sk_i	$i = \delta + 17, \ldots, n$	negligible
k_i	$i = 0, \ldots, 60$	solve 32 sparse systems of linear equations
		≈ 0.16 second on a Thinkpad T410

Generally speaking, solving a system of linear equations with m variables requires $O(m^3)$ steps by the Gaussian elimination. However, as observed in our experiment, the linear system in question is quite special such that approximately 28 equations are of type

$$k_z = sk_w + 1,$$

which immediately return the key bits. Moreover, the rest equations can be solved with around 0.005 second on a Thinkpad T410 laptop in our testing. In the worst case, the attacker has to solve 32 such systems of linear equations using around 0.16 second, which is negligible effort for the attacker.

IV. Conclusion

In this letter, we identified the security vulnerabilities of the A2U2 lightweight stream cipher and developed an ultra-efficient chosen-plaintext attack to fully recover the secret key of A2U2 through querying the encryption function twice on the victim tag and solving 32 sparse systems of linear equations with around 0.16 second. Our cryptanalysis implies that A2U2 has been completely broken and is not eligible to provide confidentiality and authenticity for RFID communications, which settled the concerns that are made in the conclusion of [5].

Acknowledgement

The authors would like to thank Mohamed Ahmed Abdelraheem, Julia Borghoff, Erik Zenner and the designers of the A2U2 cipher for clarifying the initial design intention of the filter function in A2U2 and for many valuable suggestions. This work is supported by an NSERC Strategic Project Grant.

References

[1] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin and C. Vikkelsoe, PRESENT: An ultra-lightweight block cipher, *Cryptographic Hardware and Embedded Systems, CHES'07*, pp. 450-466, 2007.

[2] T. Beth and F. Piper, The stop-and-go generator, *Advances in Cryptology, EUROCRYPT'84*, pp. 88-92, 1985.

[3] C. De Canniere, O. Dunkelman, and M. Knežević, KATAN and KTANTAN – a family of small and efficient hardware-oriented block ciphers, *Cryptographic Hardware and Embedded Systems, CHES'09*, pp. 272-288, 2009.

[4] D. Coppersmith, H. Krawczyk and Y. Mansour, The shrinking generator, *Advances in Cryptology, Crypto'93*, pp.22-39, 1994.

[5] M. David, D.C. Ranasinghe, T. Larsen, A2U2: a stream cipher for printed electronics RFID tags, *IEEE International Conference on RFID, RFID'11*, 2011.

[6] S.W. Golomb and G. Gong, *Signal design with good correlation: for wireless communications, cryptography and radar applications*, Cambridge University Press, 2005.

[7] D. Engels, X. Fan, G. Gong, H. Hu and E. M. Smith, Hummingbird: ultra-lightweight cryptography for resource-constrained devices, *14th International Conference on Financial Cryptography and Data Security, FC'10*, 2010.

[8] M. Hell, T. Johansson and W. Meier, Grain: a stream cipher for constrained environments, *International Journal of Wireless and Mobile Computing*, vol. 2, no. 1, pp. 86-93, 2007.

[9] L. Knudsen, G. Leander, A. Poschmann, and M. Robshaw, PRINTcipher: a block cipher for IC-printing, *Cryptographic Hardware and Embedded Systems, CHES'10*, pp. 16-32, 2011.

[10] Y. Luo, Q. Chai, G. Gong and X. Lai, WG-7, a lightweight stream cipher with good cryptographic properties, *IEEE Global Telecommunications Conference, GLOBECOM'10*, 2010.