Evaluation and Cryptanalysis of the Pandaka Lightweight Cipher

Yuval Yarom, Gefei Li, and Damith C. Ranasinghe

The University of Adelaide
yval@cs.adelaide.edu.au
gefei.li@student.adelaide.edu.au
damith@cs.adelaide.edu.au

Abstract. There is a growing need to develop lightweight cryptographic primitives suitable for resource-constrained devices permeating in increasing numbers into the fabric of life. Such devices are exemplified none more so than by batteryless radio frequency identification (RFID) tags in applications ranging from automatic identification and monitoring to anti-counterfeiting. Pandaka is a lightweight cipher together with a protocol proposed in INFOCOM 2014 for extremely resource limited RFID tags. It is designed to reduce the hardware cost (area of silicon) required for implementing the cipher by shifting the computationally intensive task of cryptographically secure random number generation to the reader. In this paper we evaluate Pandaka and demonstrate that the communication protocol contains flaws which completely break the security of the cipher and make Pandaka susceptible to de-synchronisation. Furthermore, we show that, even without the protocol flaws, we can use a guess and determine method to mount an attack on the cipher for the more challenging scenario of a known-plaintext attack with an expected complexity of only 2^{55}. We conclude that Pandaka needs to be amended and highlight simple measures to prevent the above attacks.

1 Introduction

Lightweight cryptography has received extensive coverage in recent years due to the growth in low cost pervasive computing technologies such as Radio Frequency Identification (RFID) propelled by significant progress in low power microelectronics and lower manufacturing costs. Batteryless RFID tags are extremely cheap, typically less than 50 cents, and enable remote and precise identification of objects or people using wireless communication between *readers* connected to back-end servers and *tags* attached to the objects or people [17]. The growing ubiquity of RFID systems and their deployment in sensitive and in high-value environments, such as their use in national passports, continue to stimulate research into the security of these low cost computing devices. However, the limited resources available at the tags, as a consequence of the desire to drive tag costs down [30], present new challenges to the provision of security mechanisms for RFID systems [9, 10].

Multiple lightweight ciphers have been proposed for such resource limited environments in the recent literature [4, 11–13, 18, 20, 22, 23, 31, 32], since exiting standard cryptographic primitives, such as AES (Advanced Encryption Standard), are much too area and power intensive to be practicable for implementation on low cost batteryless RFID tags [10, 19, 29]. However, since these ciphers are designed at the limits imposed by technology such as the number of gate equivalents (GEs) and available harvested power on batteryless tags, they are incapable of incorporating considerable security margins built into standard cryptographic mechanisms. Therefore, it is unsurprising that attacks that successfully break lightweight ciphers are frequently reported [2, 7].

Recently, Chen et al. [8] suggested Pandaka—a stream cipher tigether with a communication protocol that exploit the resource imbalance between the tags and the back-end server, based on the concept of secure server-aided computations [1, 21], to develop a lightweight cryptographic mechanism. Essentially, Pandaka shifts the bulk of the cryptographic operations to the reader, thereby reducing the implementation footprint at the tag.

At its core, Pandaka is a stream cipher that combines a secret state with a random seed generated by the reader to create a pseudo-random *derived key* which is subsequently XORed with a message block to encrypt it. The random seed, *indicators* in the Pandaka nomenclature, is also used for perturbing the state prior to the next round of encryption. For block integrity, Pandaka uses the 16 bit Cyclic Redundancy Check (CRC) generator, already available on a typical RFID tag [15]. The cipher has two suggested configurations, a 16 bit version, Pandaka(16,6), that has 96 bits of state, and a 32 bit version, Pandaka(32,6), with 192 bits of state.

We analyse the Pandaka cipher and protocol and make the following contributions:

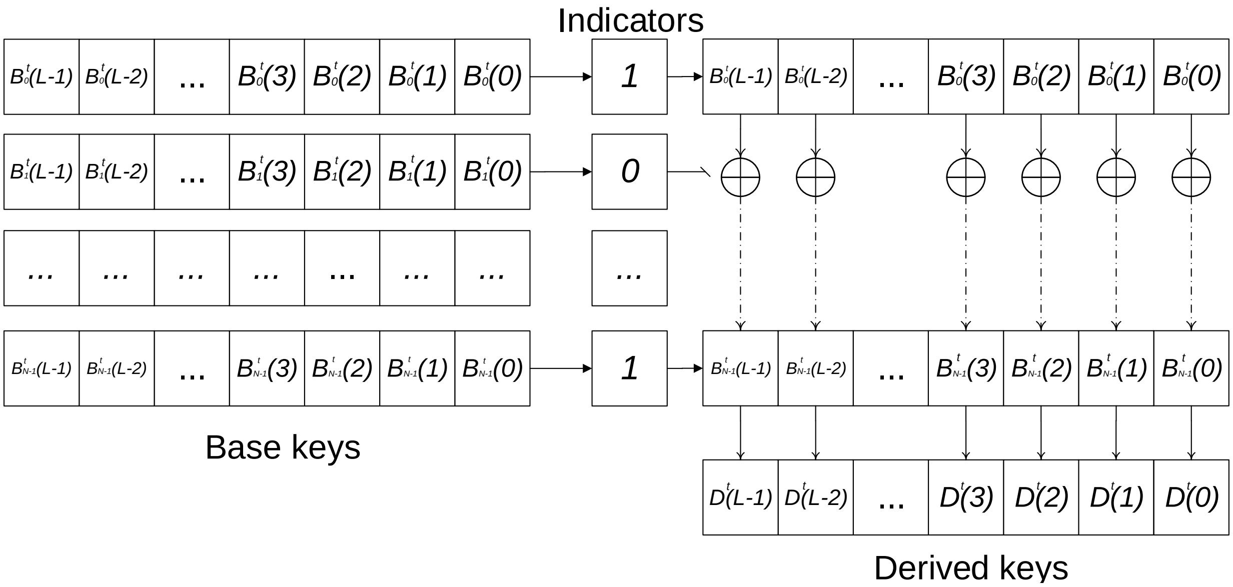
Indicators
$B_0^t(L-1)$ $B_0^t(L-2)$ ... $B_0^t(3)$ $B_0^t(2)$ $B_0^t(1)$ $B_0^t(0)$
1
$B_1^t(L-1)$ $B_1^t(L-2)$ ... $B_1^t(3)$ $B_1^t(2)$ $B_1^t(1)$ $B_1^t(0)$
0
... ...
...
$B_{N-1}^t(L-1)$ $B_{N-1}^t(L-2)$ ... $B_{N-1}^t(3)$ $B_{N-1}^t(2)$ $B_{N-1}^t(1)$ $B_{N-1}^t(0)$
1
Base keys
$D^t(L-1)$ $D^t(L-2)$ ... $D^t(3)$ $D^t(2)$ $D^t(1)$ $D^t(0)$
Derived keys

Fig. 1: Generating the derived key

- Describe a known-indicators attack on Pandaka, which exposes a weakness in the linear relationship between the state and the derived key. (Section 3.)
- Present an effective known-plaintext only attack on Pandaka, demonstrating that the security of the cipher depends on the size of the indicators rather than on the size of the reported internal state. (Section 4.)
- Highlight two weaknesses in the protocol's integrity mechanism: information disclosure; and a potential for de-synchronisation even in the absence of an active attacker. In the case of the Pandaka(16,6) configuration, the former completely reveals the plaintext in each block. (Section 5.)
- Analyse the weaknesses of the cipher and suggest directions for addressing them. (Section 6.)

2 Pandaka

Pandaka is a stream cipher that uses a shared secret between a tag and a reader, which we refer to as the *base keys*, and a random seed called *indicators* to generate a pseudorandom *derived key*. Subsequently, the derived key is XORed with the plaintext to produce the ciphertext. After generating the derived key, Pandaka updates the base keys based on the contents of the indicators. This update creates new base key material for the encryption of the next block.

Following Chen et al. [8], we use Pandaka(L,N) to denote an instance of Pandaka with a block size of L bits and N base keys. Each base key has a length of L bits, hence the size of the state of Pandaka(L,N) is $L \times N$. The length of the indicators is $N+2$.

The rest of this section describes the derived-key generation and the base-key update (or in other words state update) procedure. We also describe the protocol Pandaka uses for transferring the indicators from the reader at the heart of the base-key update procedure.

2.1 Derived key generation

Pandaka uses N bits of the $N+2$ indicator bits to select base keys. Each of these base-key selection bits corresponds to one of the base keys. As illustrated in Figure 1, Pandaka computes the bitwise XOR of the base keys whose corresponding bits in the indicators are set to generate the derived key.

More formally, let $B_k^t(i)$ denote bit $i \bmod L$ of the value of the k^{th} base key, where $k = \{0, 1, ..., N-1\}$, used for the t^{th} encryption and let $I^t(k)$ denote the k^{th} bit of the indicator used for the t^{th} encryption. The derived key for the encryption $D^t(i)$ is calculated using

$$D^t(i) = B_0^t(i) \cdot I^t(0) \oplus B_1^t(i) \cdot I^t(1) \oplus \ldots \oplus B_{N-1}^t(i) \cdot I^t(N-1) \tag{1}$$

where $\oplus$ and $\cdot$ are the XOR and the AND operations. We recall that these are also the addition and multiplication operations in $GF(2)$.

2.2 Base key update procedure

In order to avoid using the same base keys for multiple encryptions, Pandaka perturbs the base keys after generating the derived key. The base key update procedure only modifies the base keys used for the current encryption, i.e. those selected by the N base-key selection bits of the indicators. Subsequently, each of the selected base keys is rotated one bit to the left.

Following a rotation operation, Pandaka flips select set of base key bits. The decision on which bits to flip is based on the values of the additional two bits, i.e. bits N and $N+1$, of the indicators. If the value of these two bits is 00, no bits in the base keys are flipped, otherwise, for bit patterns 01, 10 and 11, Pandaka flips the base key bits whose position i modulo 3 is 0, 1, and 2, respectively.

Thus,

$$B_k^{t+1}(i) = \begin{cases} B_k^t(i) & \text{if } I^t(k) = 0 \\ B_k^t(i-1) \oplus FL(i, I^t) & \text{if } I^t(k) = 1 \end{cases} \tag{2}$$

where FL is the flip function defined as:

$$FL(i, I) = \begin{cases} 1 & \text{if } I(N) = 1, I(N+1) = 0 \text{ and } i \bmod 3 = 0 \\ 1 & \text{if } I(N) = 0, I(N+1) = 1 \text{ and } i \bmod 3 = 1 \\ 1 & \text{if } I(N) = 1, I(N+1) = 1 \text{ and } i \bmod 3 = 2 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

2.3 Communication protocol

The indicators used for encryption and decryption are generated by the reader. We assume that these are generated by a cryptographically secure random number generator. These indicators need to be communicated securely to the tag. The Pandaka protocol relies on a pre-agreed initial secret key between a tag and a reader to initiate communication. The Pandaka protocol is designed such that after each communication round the tag and the reader both share a secret derived key they can use to continue the communication.

The protocol uses three data block formats. F_1 blocks are used for transferring data from the reader to the tag. F_2 and F_3 blocks are used for transferring data from the tag to the reader. The protocol also includes protection against communication error.

***F*₁ blocks** The F_1 blocks consist of two sections. The $N+2$ least significant bits (LSBs) of the block contain the indicators used for encrypting and decrypting the next block. The other $L-N-2$ bits are for data. To send a message, the reader splits it into groups of $L-N-2$ bits, and sends each of these groups in the data section of an F_1 block.

***F*₂ blocks** The reader sends F_2 blocks to provide the tag with indicators for the blocks the tag sends. Each F_2 block contains $\lfloor \frac{L}{N+2} \rfloor$ sets of indicators. Of these, $\lfloor \frac{L}{N+2} \rfloor - 1$ are used for encrypting F_3 blocks sent from the tag to the reader, and the last set of indicators is used for encrypting the next F_2 or F_1 block sent by the reader. If $(N+2) \nmid L$, then $L \bmod (N+2)$ most significant bits (MSBs) of the F_2 block are set to zero.

***F*₃ blocks** Data is sent from the tag to the reader in F_3 blocks. Each block contains L data bits encrypted using indicators previously sent to the tag in an F_2 block. The Pandaka protocol does not specify how the reader and the tag agree on the number of F_3 blocks needed for a tag message.

Message integrity To ensure message integrity, each block is transmitted with a 16-bit cyclic redundancy check (CRC) code, using the specification outlined in the air interface protocol used by RFID tags, as specified in [15]. The CRC code is calculated on the block before the block is encrypted. On receipt, the block is decrypted and the CRC code is calculated again and matched against the transmitted code to help detect bit errors.

3 Known-Indicators Attack

First we analyse the cipher under a known-plaintext attack scenario where the corresponding indicators for a number of consecutive F_1 messages are also know. This is a simple extension of the common known-plaintext threat model to Pandaka. We show how an adversary can successfully use this information to recover the base keys of the tag. Successfully obtaining the base keys will allow an attacker to completely decipher all future communications. This attack exploits the weakness that the cipher relies completely on linear operations to generate the derived key.

We use $P^t(i)$ to denote the i^{th} bit of the t^{th} decrypted message and $C^t(i)$ to denote the i^{th} bit of the corresponding encrypted message. We note that the derived key, D^t, used for encrypting the message can be calculated using $D^t = P^t \oplus C^t$ where $\oplus$ is a bitwise XOR operation.

For each base key we compute two values: i) Q_k^t — the number of times that the k^{th} base key has been used for generating the derived key; and ii) $\mathscr{F}_k^t(i)$ — the flips applied to the i^{th} bit of base key k since the first messages, i.e. when $t = 0$. More formally,

$$\begin{aligned} Q_k^0 &= 0 \\ Q_k^{t+1} &= Q_k^t + I^t(k) \end{aligned} \tag{4}$$

and

$$\begin{aligned} \mathscr{F}_k^0(i) &= 0 \\ \mathscr{F}_k^{t+1}(i) &= \begin{cases} \mathscr{F}_k^t(i)) & \text{if } I^t(k) = 0 \\ \mathscr{F}_k^t(i-1) \oplus FL(i, I^t) & \text{if } I^t(k) = 1 \end{cases} \end{aligned} \tag{5}$$

We note that $B_k^t(i) = B_k^0(i - Q_k^t) \oplus \mathscr{F}_k^t(i - Q_k^t)$. Hence,

$$P^t(i) \oplus C^t(i) = D^t(i) = \bigoplus_{k=0}^{N-1} (B_k^0(i - Q_k^t) \oplus \mathscr{F}_k^t(i - Q_k^t)) \cdot I^t(l) \tag{6}$$

or equivalently,

$$\bigoplus_{k=0}^{N-1} B_k^0(i - Q_k^t) \cdot I^t(l) = P^t(i) \oplus C^t(i) \oplus \bigoplus_{k=0}^{N-1} \mathscr{F}_k t(i - Q_k^t) \cdot I^t(l) \tag{7}$$

Recall that in F_1 blocks, for $0 \le i < N+2$ we have $P^t(i) = I^{t+1}(i)$. Hence, given plaintext, ciphertext and indicators of T consecutive F_1 blocks, an attacker can construct $TL - N - 2$ linear equations in $B_k^0(i)$ over $GF(2)$. Solving this linear system reveals the values of the base key.

The resultant linear system has a very distinctive and sparse structure which can be exploited to rapidly evaluate a solution to the system of equations. Figure 2 shows the matrix representation of a system of equations created for a choice of six sets of indicators for Pandaka(16,6), where shaded blocks indicate the value 1 and clear blocks indicate the value 0. The base-key selection bits of these indicators are 110100, 110010, 011100, 110100, 100100 and 001000.

As Figure 2 demonstrates, the matrix is divided into N groups of L columns, each group corresponding to a base key. The rows are also divided into groups of L, each group corresponding to a set of indicators. An $L \times L$ block is empty if the corresponding indicator bit is 0, otherwise, the block contains a possibly rotated $L \times L$ identity matrix. The size of the rotation is determined by the number of instances the corresponding base key has been selected by previous indicators.

For $T \le N$, the number of equations in the system is less than the size of the base-key bits NL, hence at least $N+1$ blocks are required to solve the system. However, having $N+1$ blocks does not guarantee a solution, to solve the system its rank must be equal to NL. In other words, the system should have NL independent equations.

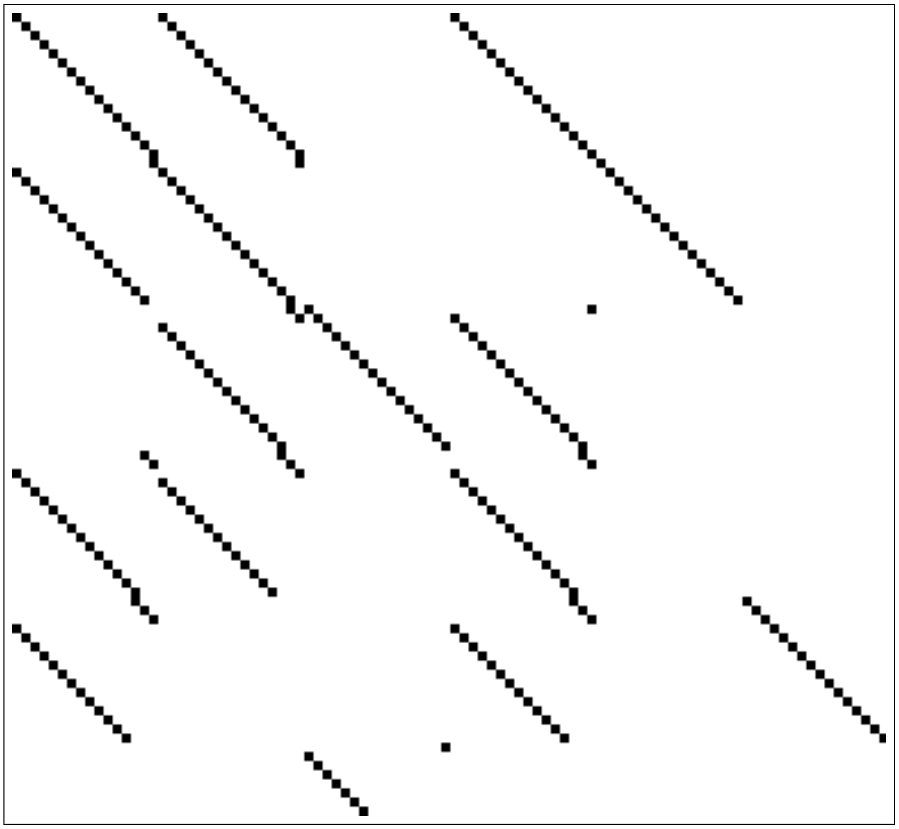

Fig. 2: Structure of a linear system

The number of blocks required depends on the values of the base key selection bits in the indicators in each block. Figure 3 shows the distribution of the number of blocks required over 1,000,000 random instances of the attack. On average Pandaka(16,6) and Pandaka(32,6) require 7.76 and 8.12 blocks, respectively, with a worst case scenario of 27 blocks.

4 Known-Plaintext Attack

The attack we describe in the previous section assumes the attacker knows the indicators. However, since the indicators are assumed to be randomly chosen, such an assumption may be unrealistic. Furthermore a known-indicators attack is also not included in the threat models considered by Chen et al. [8].

In this section, we remove the notion of known indicators and instead consider the more challenging known-plaintext attack. We describe a known-plaintext attack which allows an adversary to completely recover the base key using the plaintext (excluding the indicators) and the corresponding ciphertext of only a handful of consecutive F_1 blocks. By breaking the base key, such an attacker can successfully decrypt further message blocks exchanged between the tag and the reader.

Here we use a guess and determine [3, 16, 27] approach where we guess the values of the indicators of some of the blocks and subsequently apply the procedure from the known-indicators attack in Section 3 to determine the values of the base keys.

The attack uses the recursive algorithm shown in Algorithm 1. It scans all possible values of the indicator sets, starting with I^0. For each value, the attack builds a system of linear equations using the technique discussed in Section 3 and, based on the properties of the linear system, decides on one of three options to proceed. If the linear system is inconsistent, it is clear that the current guess is wrong and the attack moves to the next guess. If the system is consistent,

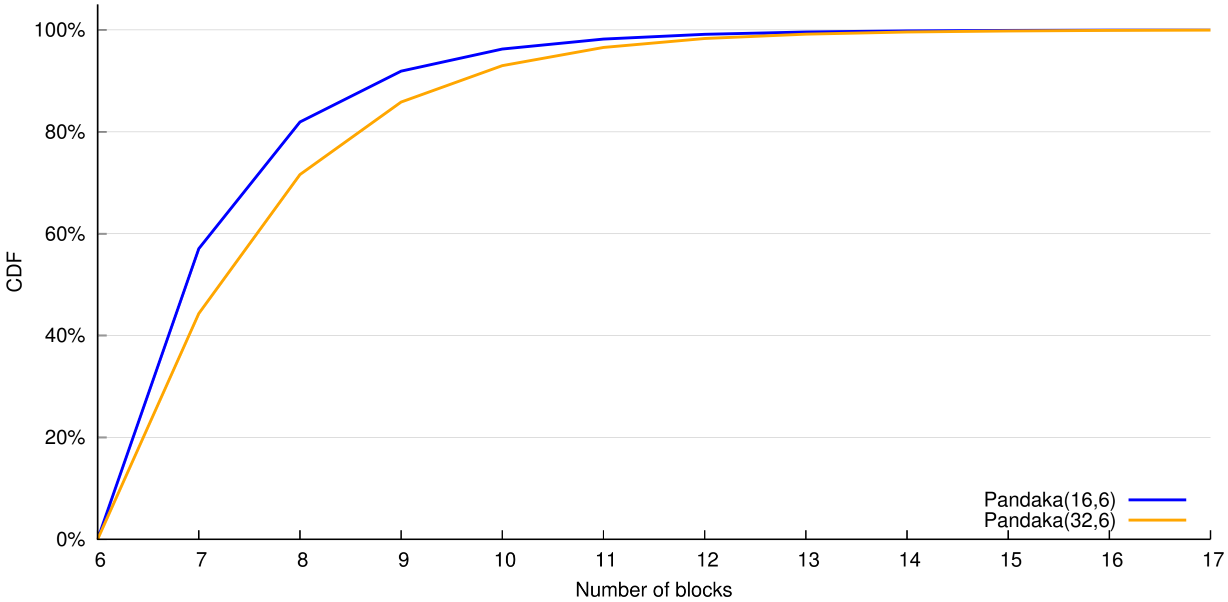

Fig. 3: Distribution of the number of blocks required for a known-indicators attack

there are two possibilities: i) the rank of the system is NL, in which case we can solve the system, find the initial value of the base keys and verify the solution; or ii) the rank of the system is less than NL where we do not have a solution and need to recursively guess the next set of indicators.

We use Gauss elimination with implicit row pivoting to test for consistency and to calculate the rank of the linear system. We note that the $(T-1)L-N-2$ first equations in the system do not depend on the value of the indicators of the T^{th} round. Consequently, we do not need to apply the Gauss elimination process to the entire matrix for each guess. Instead, we can pre-compute the result of the elimination on the first $(T-1)L-N-2$ equations once. We can then use the pre-computed value to complete the elimination process on the L rows that are affected through the recursive process of guessing the indicators.

It is important to note that the value of the flipped bits (see Equation 3) of the selected base keys does not affect the structure of the system of equations. That is, bit flips only affect the right-hand side of Equation 7. Therefore, we can reuse the results of one Gauss elimination to all four indicator values that share the base-key selection bits and only differ in the value of the two flip bits. (i.e. indicator bits that define the four possible bit flips given in Equation 3.)

Furthermore, we can optimise the guess and determine approach by halting the guessing of indicators when the rank of the system is greater than $NL-N$ and instead adding an adequate number of equations of the form $B_k^0(i) = x_j$ to obtain a full-ranked system and evaluate its solution. The number of equations we add is smaller than the number of base-key selection bits in an indicator; hence, this approach reduces the number of cases we need to evaluate.

In order to calculate the expected number of guesses to completely recover the base keys of Pandaka(16,6) and Pandaka(32,6) we first examine the structure of the linear system of equations created by the guesses of the first four rounds. At this stage, the linear system has $4L-N-2$ equations, or 56 equations for Pandaka(16,6) and 120 for Pandaka(32,6). The rank of the system is not necessarily the same as the number of equations. Table 1 summarises the distribution of ranks over all possible combinations of indicators for the first four rounds.

As discussed above, the values of the base-key selection bits in the indicators determine the structure of the linear system. Using the same sequence of base-key selection bits in Pandaka(16,6) and Pandaka(32,6) produces similar systems of equations. Consequently, the distribution of the ranks of the linear system of equations indicated by the probability in in Table 1 are the same in both versions of Pandaka where the only difference is the numeric value of the ranks.

```
input  : $I$: guessed indicators for the first $T$ rounds
         $P$: plaintexts of the first $n$ rounds
         $C$: ciphertexts of the first $n$ rounds
output: Initial base-keys and indicators for the first round, if found
if $T > 0$ then
    Use Equation 7 with $I$, $P$, and $C$ to create a system of $TL - N - 2$ linear equations ;
    if the system is not consistent then
        return false;
    if the rank of the system is $NL$ then
        Solve the linear system;
        if the solution matches all $n$ known rounds then
            Output solution;
            return true;
        else
            return false;
        end
    end
end
foreach possible indicators value $i$ do
    $I' =$ add $i$ to $I$;
    Recursively call this algorithm with $I'$, $P$ and $N$;
    if result found then
        return true;
end
return false
```

Algorithm 1: Known-Plaintext Attack

We now estimate the number of guesses required for completely scanning all possible values of the indicators given the indicator bits of the first four rounds. Summing the estimate over all possible combinations of the first four indicators gives an estimate of the size of the search space for the attack. There are 252 possible indicator values (there are 256 possible 8 bits combinations, of which the four with no base-key selection are illegal). Consequently, there are 252^4 possible combinations of four indicators.

We first look at the case where the indicators result in a system with a maximal rank, i.e. 56 for Pandaka(16,6) and 120 for Pandaka(32,6) and estimate the number of indicator guesses required to completely scan all of the combinations of indicator bits that result in a consistent non-full ranked system. For that, we generate 1,000 random instances of Pandaka and evaluate the number of guesses required for solving each. For Pandaka(16,6), we require an average of 9.21 million guesses, with a 99% confidence interval of 0.20 million. For Pandaka(32,6), the average is 9.23 million and the 99% confidence interval is 0.18 million. We note that, due to the overlap of the confidence intervals, the estimates for Pandaka(16,6) and for Pandaka(32,6) are statistically indistinguishable. Hence, we conclude that the number of cases required does not depend on the block length L.

Table 1: Rank distribution after four rounds of guesses

Pandaka(16,6)	Pandaka(32,6)	Probability
16	32	.000004
24	56	.0002
32	64	.0016
40	88	.0440
47	95	.0008
48	96	.0532
55	119	.0149
56	120	.8852

We argue that using using the higher estimate above (9.23 million) for the case of a system with a lower rank is an overestimate of the number of guesses required for covering the whole search space. In a nutshell, when the rank of the system is lower, more iterations are required for solving the system and thus increasing the number of guesses, in contrast, when the rank of the system is lower, the probability of the attack ignoring the case due to linear-system inconsistencies is higher. We argue that the latter grows faster than the former, so that the expected number of guesses is lower than for a case of a fully-ranked system.

More specifically, we postulate that for each L dependent equations in the system we need to guess another round of indicators to get a system of degree NL. For simplicity we assume that because we have $N+2$ indicator bits, adding another round increases the number of guesses by a factor of 2^{N+2}, or 256 for the two Pandaka configurations.

We validate this assumption by counting the number of guesses required for solving consistent systems of rank $3L-N-2$, i.e. systems in which L equations are dependent. The results are 2,039 million and 2,085 million for Pandaka(16,6) and Pandaka(32,6), respectively. These numbers are about 225 times larger than our estimate of the number of guesses required for solving the case of fully-ranked systems. Hence the assumption we used, increasing the guesses by a factor of 256, is an overestimate of the number of guesses required.

Chen et al. [8] demonstrates that each bit of the derived key is equally likely to be 0 or 1. Consequently, each dependent equation in the linear system we produce has a $1/2$ probability of resulting in an inconsistency. Thus, for a given guess of four indicators, if the difference between the rank of the system and the number of equations is r, the probability of the attack proceeding beyond these four indicators is 2^{-r}. Thus, the expected factor is in the order of $2^{\frac{(N+2)r}{L}} \cdot 2^{-r} = 2^{\frac{(N+2-L)r}{L}}$ and because $N+2 < L$ using the estimate of the fully-ranked system is an overestimate of the number of guesses required for non-fully-ranked systems.

With 252^4 possible combinations of indicators for the first four rounds, where 9.23 million guesses are required for each combination, the size of the search space is estimated at $252^4 \cdot 9.23 \cdot 10^6 \approx 2^{55}$. It is important to highlight that the same attack complexity applies to both Pandaka(16,6) and Pandaka(32,6). More significantly, the complexity we have evaluated is significantly lower than that postulated by Chen et al. [8] where they claim an attacker will need to guess all the values of the base keys, or 2^{96} and 2^{192} for Pandaka(16,6) and Pandaka(32,6), respectively.

The amount of plaintext required depends on the number of blocks required for solving the system. In Section 3 we see that up to 27 blocks may be required, with a typical number of 7 (Pandaka(16,6)) or 8 (Pandaka(32,6)) blocks. (See Figure 3.) Additionally, because we are trying a large number of guesses, we need further plaintext bits to have a sufficiently high confidence that we have found the right key. Each additional bit halves the probability of accepting a wrong guess. Hence, with 2^{55} guesses and 55 additional bits, we have a probability of $1/e$ of accepting a wrong guess. With 74 additional bits the probability drops to below one attack in a million. Hence, for the typical case, we require 130 and 266 bits of plaintext for Pandaka(16,6) and Pandaka(32,6), respectively. For the worst case we require 290 and 722 bits.

5 Targeting The Protocol Flaws

Analysis of the communication protocol in Pandaka reveals a key design flaw related to the integrity check employed using CRCs[28]. The CRC code used for checking the integrity of messages reveals excessive amounts of information on the contents of the encrypted message.

CRC is a standard method of ensuring message integrity in network communication. RFID tags already include the circuity for calculating the proposed 16 bit CRC [15] and the CRC is used to identify bit erroneous communications as a result of bit errors. Pandaka reuses this circuity to ensure its messages' integrity and thus avoiding the cost of a dedicated circuity for evaluating the CRC.

5.1 Ciphertext-Only Attack

While the 16 bit CRC offers a high probability of error detection, for example detect any single error burst less than 16 bits, it is designed to protect against unintentional errors and is not cryptographically secure. As described earlier, Pandaka calculates the CRC on the message before the encryption and transmits it together with the encrypted message. Thus, the CRC in Pandaka reveals 16 bits of information on the plaintext. For Pandaka(16,6), the 16 bit version of the protocol, the CRC effectively discloses the whole plaintext, negating the protection of the encryption. Therefore the current description of Pandaka(16,6) is completely broken.

For Pandaka(32,6), the CRC could be used to elevate a known-plaintext attack to a known-indicators attack. (See Section 3.) It is also possible that the CRC could be used as a source of information of the plaintext and subsequently facilitating a ciphertext only attack on Pandaka(32,6). One possibility for implementing the attack is to guess the 8 indicators bit in F_1 blocks. From this information and from the CRC, the attacker can create a set of 24 linear equations and subsequently use the attack in Section 4 to break the cipher. This attack has the potential of reducing the complexity of a ciphertext-only attack on Pandaka(32,6) from 2^{192} to approximately 2^{72}.

5.2 Active Attacks

Not being cryptographically secure also means that the CRC does not protect against malicious modifications of messages. The CRC code is linear, that is, given two messages A and B, $CRC(A \oplus B) = CRC(A) \oplus CRC(B)$. Thus, an active attacker can modify transmitted messages by flipping bits in the encrypted message and then calculate the correct CRC for the modified message even without knowing the contents of the message. However, it should be noted that such an attacker is beyond the threat model considered by Chen et al. [8].

The weakness of the CRC also results in a vulnerability to de-synchronisation attacks. With a 16 bit CRC, there is a probability of 2^{-16} of an arbitrary message having the correct CRC. If an attacker generates enough random messages, one of them is likely to have the correct CRC. When a tag receives such a message, it is accepted and Pandaka updates the cipher base-keys. At this stage, the base-keys at the tag diverges from that at the reader, preventing any further communication between the two. The de-synchronisation attack, Like the message modification vulnerability described above, is outside the threat model of Chen et al. [8].

6 Discussion

Pandaka aims to reduce the complexity of the tag by shifting the random number generation logic to the reader. While the idea is appealing and is worth further investigation, the implementation fails to meet the desired security level. In this section we review the main weaknesses of the implementation and suggest measures for addressing them.

Confusing randomness with security For a stream cipher to be secure its random number generator must have good statistical properties. The converse, however, does not hold. A "good" random number generator that passes many standard tests for randomness is not necessarily cryptographically secure.

We recommend that, in addition to statistical tests, Pandaka is subjected to known and successful cryptanalysis techniques employed with stream and block ciphers such as linear cryptanalysis [24–26], differential cryptanalysis [5, 6], and guess and determine [3, 16, 27].

Linearity Linear systems are easy to reverse because they can be efficiently solved. More significantly, sparse systems of equations can be stored using less memory and solved extremely rapidly. Consequently, cipher designs aim to avoid linearity by including non-linear state update functions. Pandaka, however, only uses linear operations to update the state of the cipher.

The introduction of non-linear state update functions, both for generating the derived key and for perturbing the base-keys while increasing diffusion, would significantly increase the security of the cipher and provide protection against our attacks.

Limited base-key perturbation The purpose of the base-keys (or state) update is to provide new key material for following rounds. Pandaka uses a simple base-key update algorithm whose implementation only requires a small number of gates. However, the key material is hardly mixed, in fact mixing between base keys are non-existent. In Pandaka, key material of a base key is only used within the base key and the update (bit flip and rotation) of a given base-key bit depends only on the state of a single indicator bit and all base keys are updated using the same algorithm.

Unfortunately, despite the simplicity in the hardware implementation, the algorithm is extremely easy to analyse and break. By combining the values of multiple key bits from multiple base keys to determine each updated base-key bit will increase the security of the procedure.

Synchronisation Pandaka is a synchronous cipher with no mechanism for re-synchronisation of the sender (e.g. tag) and receiver (e.g. reader) in the event of lost messages. While the problem of bit errors have been addressed with the CRC, the cipher will easily self de-synchronise during: i) packet loss that often occurs in RFID communication networks due to packet collisions resulting from basing the air interface protocol of RFID system on the Slotted ALOHA protocol for facilitating simultaneous communications with multiple other tags[15]; ii) packet corruption and packet loss due to interferences from other readers communicating nearby that interfere and increase the noise in the communication channel between a reader and a tag referred to as the reader collision problem[14]; and iii) more rarely, a CRC collision (bit errors producing a message block with an identical CRC to the original value calculated by the sender)[28].

Thus Pandaka is vulnerable to self de-synchronisation even without the presence of an active attacker simply from corrupt messages and packet loss due to the wireless propagation environment and the nature of the communication protocol between RFID readers and tags.

7 Conclusions

Pandaka is designed for resource limited RFID tags. In order to reduce the hardware (area) cost of implementing the cipher in silicon chips Pandaka has used short shift registers and linear operations. The computationally intensive task of generating random numbers is allocated to the more resourceful RFID readers and overcomes the need to implement such a generator on the tag. Therefore Pandaka manages to significantly reduce the cost of implementing the cipher on a tag. Together with three message types and reader generated random numbers, Pandaka develops a state update mechanism that requires minimal hardware at the tag.

In this article we discuss several practical breaks of the Pandaka lightweight stream cipher. In particular, we show that in the more challenging known-plaintext scenario, using a guess and determine attack approach, Pandaka can be broken with an attack complexity of 2^{55} guesses using a known plaintext length of approximately 170 bits for Pandaka(16,6) and approximately 270 bits for Pandaka(32,6). Furthermore, we show that the information leak in the protocol by way of the CRC completely removes any protection provided by Pandaka(16,6) and dramatically reduces the attack complexity of a ciphertext-only attack on Pandaka(32,6).

We conclude our analysis by pointing out some of the most severe weaknesses of the cipher. The most obvious weakness is that the CRC value computed to improve message integrity exposes information on unencrypted blocks. Then, secondly, the lack of non-linearity in the design of the state update function. Although we suggest avenues for improving the cipher, any new design may be vulnerable to different attacks from those we have analysed and therefore a full analysis of the cipher would need to be performed in order to assess the strengths of any potential changes.

Acknowledgements

The authors wish to thank Dr Sylvan Elhay for the useful advice and productive discussions. This research was supported by a grant from the Australian Research Council (DP140103448).

Bibliography

[1] Martín Abadi, Joan Feigenbaum, and Joe Kilian. On hiding information from an oracle. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 195–203, New York, New York, United States, 1987.

[2] Mohamed Ahmed Abdelraheem, Julia Borghoff, Erik Zenner, and Mathieu David. Cryptanalysis of the lightweight cipher A2U2. In *Cryptography and Coding*, volume 7089 of *Lecture Notes in Computer Science*, pages 375–390, Oxford, UK, December 2011.

[3] Hadi Ahmadi and Taraneh Eghlidos. Heuristic guess-and-determine attacks on stream ciphers. *IET Information Security*, 3(2):66–73, 2009.

[4] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptology ePrint Archive*, 2013:404, 2013.

[5] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1): 3–72, 1991.

[6] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard.* Springer, 1993.

[7] Andrey Bogdanov and Christian Rechberger. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In *Proceedings of the 17th International Workshop on Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 229–240, Waterloo, Ontario, Canada, August 2011.

[8] Min Chen, Shigang Chen, and Qingjun Xiao. Pandaka: A lightweight cipher for RFID systems. In *Proceedings of IEEE INFOCOM 2014*, pages 172–180, Toronto, Ontario, Canada, April 2014.

[9] Peter H Cole and Damith C Ranasinghe. Networked RFID systems and lightweight cryptography: Raising barriers to counterfeiting. *London, UK: Springer. doi*, 1:978–3, 2008.

[10] Peter H. Cole, Leigh H. Turner, Zhonghao Hu, and Damith C. Ranasinghe. The next generation of RFID technology. In Damith C. Ranasinghe, Quan Z. Sheng, and Sherali Zeadally, editors, *Unique Radio Innovation for the 21st Century: Building Scalable and Global RFID Networks*, pages 3–23. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-03461-9.

[11] Mathieu David, Damith C. Ranasinghe, and Torben Larsen. A2U2: A stream cipher for printed electronics RFID tags. In *IEEE International conference on RFID*, pages 176–183, Orlando, Florida, United States, April 2011.

[12] Christophe De Cannière, Orr Dunkelman, and Miroslav Knežević. KATAN and KTANTAN—a family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288, Lausanne, Switzerland, September 2009.

[13] Daniel Engels, Markku-Juhani O. Saarinen, Peter Schweitzer, and Eric M. Smith. The Hummingbird-2 lightweight authenticated encryption algorithm. In *RFID. Security and Privacy*, volume 7055 of *Lecture Notes in Computer Science*, pages 19–31. Amherst, Massachusetts, United States, June 2012.

[14] Daniel W Engels and Sanjay E Sarma. The reader collision problem. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, volume 3, pages 6–pp. IEEE, 2002.

[15] *EPC™ Radio-Frequuency Identity Protocols Generation-2 UHF RFID Version 2.0.0 Ratified.* EPCGLOBAL, November 2013.

[16] Xiutao Feng, Jun Liu, Zhaocun Zhou, Chuankun Wu, and Dengguo Feng. A byte-based guess and determine attack on SOSEMANUK. In *Advances in Cryptology—Asiacrypt'2010*, pages 146–157, Singapore, December 2010.

[17] Klaus Finkenzeller. *RFID Handbook.* Wiley Online Library, 2003.

[18] Zheng Gong, Svetla Nikova, and Yee Wei Law. KLEIN: A new family of lightweight block ciphers. In *RFID. Security and Privacy*, volume 7055 of *Lecture Notes in Computer Science*, pages 1–18, Amherst, Massachusetts, United States, June 2012.

[19] Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. In *Advances in Cryptology—Crypto 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 293–308, Santa Barbara, California, United States, August 2005.

[20] F. Karakoç, H. Demirci, and A.E. Harmancı. AKF: A key alternating Feistel scheme for lightweight cipher designs. *Information Processing Letters*, 115(2):359–367, 2015.

[21] Shin-ichi Kawamura and Atsushi Shimbo. Fast server-aided secret computation protocols for modular exponentiation. *IEEE Journal on Selected Areas in Communications*, 11(5):778–784, June 1993.

[22] Lars Knudsen, Gregor Leander, Axel Poschmann, and Matthew J.B. Robshaw. PRINTCIPHER: a block cipher for IC-printing. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 16–32, 2010.

[23] Yiyuan Luo, Qi Chai, Guang Gong, and Xuejia Lai. A lightweight stream cipher WG-7 for RFID encryption and authentication. In *Global Telecommunications Conference (GLOBECOM 2010)*, pages 1–6, Miami, Florida, United States, December 2010.

[24] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology—Eurocrypt' 93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397, Lofthus, Norway, may 1994.

[25] Mitsuru Matsui and Atsuhiro Yamagishi. A new cryptanalytic method for FEAL cipher. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E77(1):2–7, January 1994.

[26] Kaisa Nyberg. Linear approximation of block ciphers. In *Advances in Cryptology—Eurocrypt'94*, volume 950 of *Lecture Notes in Computer Science*, pages 439–444, Perugia, Italy, May 1995.

[27] Enes Pasalic. On guess and determine cryptanalysis of LFSR-based stream ciphers. *IEEE Transactions on Information Theory*, 55(7):3398–3406, July 2009.

[28] W.W. Peterson and D.T. Brown. Cyclic codes for error detection. *Proceedings of the IRE*, 49(1):228–235, Jan 1961. ISSN 0096-8390. doi: 10.1109/JRPROC.1961.287814.

[29] Damith C. Ranasinghe, Daniel W. Engels, and Peter H. Cole. Low cost RFID systems: Confronting security and privacy. *Paper Auto-ID Labs White Paper Journal*, 1, 2005.

[30] Sanjay E. Sarma. Towards the 5 cent tag. *White Paper-MIT Auto-ID center*, 2001.

[31] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: an ultra-lightweight blockcipherlockcipher. In *Cryptographic Hardware and Embedded Systems–CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 342–357, Nara, Japan, September 2011.

[32] Wenling Wu and Lei Zhang. LBlock: A lightweight block cipher. In *Applied Cryptography and Network Security*, volume 6715 of *Lecture Notes in Computer Science*, pages 327–344, Nerja, Spain, June 2011.