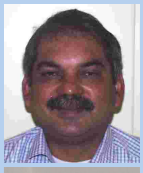
The University of Adelaide

AUTO–ID LABS

One Time Codes

Raja Ghosal, Manfred Jantscher, Alfio Grasso and Peter H. Cole

Auto-ID Labs White Paper WP-HARDWARE-030



Mr. Raja Ghosal
PhD Student, Auto-ID Lab, ADELAIDE
School of Electrical and Electronics Engineering,
The University of Adelaide

Mr Manfred Jantscher
Overseas Visiting Student, Auto-ID Lab, ADELAIDE
School of Electrical and Electronics Engineering,
The University of Adelaide

Mr. Alfio R. Grasso
Deputy Director, Auto-ID Lab, ADELAIDE
School of Electrical and Electronics Engineering,
The University of Adelaide

Prof. Peter H. Cole
Research Director, Auto-ID Lab, ADELAIDE
School of Electrical and Electronics Engineering,
The University of Adelaide

Contact:

E-Mail: rghosal@eleceng.adelaide.edu.au, m.jantscher@student.tugraz.at, alf@eleceng.adelaide.edu.au or cole@eleceng.adelaide.edu.au

Internet: www.autoidlabs.org

Hardware

Abstract

One time codes, originated from the works of Vernam (Vernam cipher (1917)), and Capt Joseph Maubourne around the same time. They have been used in military for the highest secret communications. One-time codes are the most secure means of encryption. They were proven to be ideal secret codes by Shannon (1949).

When using one-time codes, both the sender and the receiver must use the same key sequence, which must be destroyed after the message is decrypted, and a new key sequence used for the next message. As the one-time code can only be used once, and the generation of the code must be truly random, it is not practically feasible to use one-time codes for heavy traffic applications. In such high traffic application a pseudo-random but cryptographically secure code sequence is used.

In day to day operations such as internet banking and other applications where multiple users generate requests, one-time codes or any variant would not be scaleable due to the high volume of transacting sessions. Hence, in internet banking and other applications, public key cryptography is used where users mutually agree to some key exchange protocol and the public key decryption process is computationally intractable (to the intruder). Also in most circumstances such internet traffic is hard wired, and in some limited cases part wireless, employing well defined mobile, bluetooth wireless security protocols. Such systems, while lightweight, are not suitable for RFID, as they require greater computational power.

In RFID systems, due to the publicly available wireless medium information, and shorter lifetime, one-time codes would be ideal for security. In RFID systems, one-time codes require simple xor operations for encryption and decryption. A perfect random code generator is practically infeasible, but a stream cipher such as RC4, which has a very long period time (10^{1000}) is a sound option. Stream ciphers use LFSR (Linear Feedback Shift Registers).

Quantum cryptography based on quantum states of electrons, photons, and other subatomic particles, can generate random sequences and hence be used to generate one-time codes. Radio astronomy based noise from distant galaxies is another source of random codes, assuming that both the sender and receiver are very well synchronized. The receiver must be informed of the pattern, or perhaps obtain the pattern at the same time say from the same physical source. Another source of random codes is based on the Internet. Synchronization, to the specific one-time code by both sender and receiver is crucial. There is considerable scope for research in the generation, synchronization, recovery methods, of one-time codes.

This paper proposes a new RFID system that can be used in security and authentication applications, based on one-time codes. Many of the definitions in this document have come from Wikipedia (http://en.wikipedia.org/), and are licensed under the GNU Free Documentation License.

1. Introduction

One-time pads have been used by military and other highly confidential intelligence based security operations. This originated from Gilbert Vernam, known as Vernam cipher (1917) [22] . Vernam used a 5-bit baudot code. US military captain, Joseph Maubourne [27], proposed in addition a paper tape that had random digits. The two ideas when combined became known as one-time-pads [21] and [27]

Stream cipher (sometime known as a state cipher), which use a pseudo-random number generator, such as the LFSR techniques, [8] and [19], while often used to generate one-time codes, are not as secure as purely random numbers, but nevertheless need to be very robust, i.e. have long cycle times. Most so called random number generators generate pseudo random sequences; where over a large cycle some correlation patterns begin to emerge. These ciphers cannot be used for highly secure one-time code message transactions. In such secure message transactions one-time codes must be at least of the same length as the original message, and must be purely random.

In military intelligence use during World War 1 and II, pages of specific books, known to both sender and receiver were used as one-time pads (or one-time codes). Once a message was sent, a randomly different page, and/or another book, but both mutually known to both parties were used. The natural extrapolation of this process, in today's internet era, would be to use web pages. However, using the pages of books, or web pages cannot be said to be a purely random process. Language characteristics (what Shannon quantified as unicity distance [17], eg: for a given language like English), may mean that using a book in literature, philosophy or physics may have some specific correlations, and hence each occurrence of a symbol on a page is not truly random. The generation of a truly random source is the challenge of one-time codes.

Quantum cryptography, based on spins, quantum states of subatomic particles, like electrons, photons, is emerging as a possible avenue. One of the current applications, of quantum cryptography is not in data encryption, but only for regular changing of keys randomly in a session in public key cryptography, where earlier the keys were fixed for the duration of the session. Quantum cryptography may be able to provide truly random one-time codes in future. Some of the latest techniques have been the use of radio-noise in astrophysics, as random one-time codes. Both sides must record the noise activities and have very sound synchronization methods, for correct encryption and decryption. Another method is to use Schneier's 2-factor authentication to lead to a possible one-time code source [18], where there is a fixed part like a password, that does not change, and there is a variable part such as time of day, or something dynamically changing, making it difficult for an intruder to attack.

The objective of this paper is to provide an overview of one-time codes, random and pseudo-random generators used to provide sets of one-time codes, with respect to use within an RFID system. In addition, a new RFID system, exploiting one-time codes is described and analysed.

The following section outlines why one-time codes are amenable to be used in RFID systems. Sections 3 and 4 describe various methods of generating random codes from stream and block ciphers. Users of one-time codes and RFID uses are discussed in Section 5. The difference between one-time codes and one-way functions are discussed in Section 6, while Section 7 addresses management issues of one-time codes. Section 8 proposes a new RFID scheme in which one-time codes are used to offer both privacy and authentication, while Section 9 lists other research that may be exploited to generate one-time codes in the future. Finally conclusions are drawn in Section 10, followed by references in Section 11.

2. One-Time Codes and RFID

One-time code is the most secure from of encryption [17], [20], [21],[22] and [27]. It is proven to be the most secure and ideal communication scheme by Shannon's theory and analysis [17]. Based on purely random set of numbers or bits or patterns, they are very amenable to RFID applications due to very simple xor operations, for encryption and decryption. It is assumed the receiver for decryption, has the same set of one-time codes, random bits, and simply xor's the received ciphertext, C to retrieve the original message in plaintext, P.

3. One-Time Codes and Stream Ciphers

One-time codes (also referred to as one-time pads) are currently being used to generate provide ideal secrecy [11], [23] and [27]. The one-time code is used to xor with the plaintext, P, to give the ciphertext, C. Theoretically, any intruder with the best of computing capabilities, would not be able to attack such a system, as the codes are fully random. Hence they offer perfect secrecy [17]. As the codes are used only once, the generation, deletion of the current, and updating to a new set of codes, the distribution of such codes (the receiver and sender must have exactly the same one-time codes at any time), is not a scaleable operation in management. Hence in large public systems, with many users, eg: internet banking, other applications, one-time codes are not used. These networks also use the business's dedicated networks and hence have some security at the physical level being "wired or cabled" networks, and properly shielded so that the intruder cannot passively listen to the radio signals. RFID systems are very vulnerable, sharing the open public wireless environment. Also because the use of a tag is very short, when scanned, they are very strong candidates for one-time codes, provided the latter is feasible.

Stream ciphers are developed as approximation to the behaviour or action of the one-time code. The current stream ciphers are unable to generate the ideal random numbers, hence unable to fulfil Shannon's ideal secrecy condition [11],[17] and [27]. However they are of interest as they are the most widely used methods in practice.

RC4 is a pseudo random cipher used fairly commonly, and strong candidate for one-time codes. The RC4 cipher, has a period of 10^{1000} [15]. Hence this serves as a good pseudo-random sequence.

There are two types of stream ciphers [11]—

- synchronous,
- self-synchronous

A stream cipher generates the next set and the following set of keys based on the internal state (this is a limitation with regards to one-time purely random codes, as there should be no correlation with any state).

The internal state is updated in two different ways:

- If the state changes independently of the plaintext or ciphertext messages, the cipher is called a synchronous cipher.
- If the stream changes its state based on previous cipher digits they are called, self-synchronizing stream ciphers. Self synchronizing ciphers have the advantages that after receipt of the N digit block, the synchronization is automatically done. An example of this is the Block cipher in binary cipher-feedback mode (CFB).

Operations performed on stream ciphers, are xor, the binary exclusive or. They are also known as binary additive stream cipher [23].

In the synchronous cipher, the sender and receiver must be in step, synchronized for the encryption and decryption to be successful. If any extra text is inserted or is lost, then the synchronization is lost, and the encryption-decryption is not possible. One approach to attempt to resynchronise tries different offsets within the set of ciphers to obtain the correct synchronization, an iterative approach until the output resembles a valid plaintext message. Another approach is to use a marker code (which is recognised by the receiver), and allows synchronisation. Binary additive cipher is another possible approach [23].

Stream ciphers are prone to active attacks. Stream ciphers have the property that if a bit is inverted in the ciphertext, then after decryption the corresponding bit in the plaintext output is also inverted. If an attacker receives a ciphertext, and tries various combinations of inverting different ciphertext bits, they maybe able to predict the bit inversion in the resultant plaintext and hence determine the key used to encrypt the original message.

Generation of Stream Ciphers: Stream ciphers use LFSR (Linear Feedback Shift Register). Highly secure one-time codes should not use feedback shift registers. Such LFSR systems can only generate pseudo-random number generators. Non-linear circuits are generated by adding some Boolean logic. Non-linear circuits due to non-availability of mathematical model cannot be proven to be more secure. There are some examples given [17] where such circuits are shown to be less secure.

Analysis: Welsh [20] shows the Goldwaser and Micali (1982) scheme which is based on the secrecy of all partial information about transmitted messages, and "cryptographically safe" pseudo random numbers, is a possible secure one-time application.

Further Welsh [20] analyses the one-time pad: Vernam pad, on 26 letter English alphabet. If any random sequence (Z1, Z2,..., Zn) of letters is chosen for the key K, for a plaintext

message of length n. The ciphertext (y1, y2, .., yn), is obtained by adding each, randomly chosen, Zi, is added to plaintext M =(x1, x2,.., xn) to produce ciphertext:

$y_i = x_i + Z_i \pmod{26}$

For a message of length, n, there are 26^n equally probably keys, hence Welsh [20] obtains the entropy, for the random key system as, H(K) = n log26

For a given ciphertext, C, the probability K = (Z1,Z2,.., Zn), or that an arbitrary set of Z's as above chosen by intruder, is $P(K \mid C) = 1/(26^n)$

Sometimes to distribute equally language dependent features eg: ‘e’ and ‘a’ have the most occurrences in English, the alphabet is mapped to {1,2,..., 100}.The first “a” is 1 plaintext. The 2nd occurrence of “a” maybe 15, 3rd 35 etc.., so that in the mapped domain {1,2,.., 100} each symbol equally likely.

Welsh [20] points out that the one-time pad has perfect secrecy. Any other cipher based, LFSR, pseudo-random scheme has a lower entropy and therefore not as secure.

The shortcomings of the cipher stream systems are that:

(a) there is no mathematical way of generating independent random bit patterns. One would have to rely on a "pseudo-random" sequence generated by one of the many standards (eg: ciphers, via LFSR, most common). There is no guarantee that such pseudo-random sequences would give the same level of security.

(b) length of key is exactly same as that of message, and if a 'genuine random sequence' obtained by some physical process (such as radioactive decay) is used as key, then this key must also be communicated to the receiver before decryption can be carried out.

General Random Number Generator: A general pseudo-random number generator is:

r(n+1) = a*r(n) + b (mod m), r(0) = c, with a, b, m, and c chosen initial integers, c = seed. a, b, c in (0, m-1).

Binary Generators: Welsh [20] lists some of the following useful results, for binary generators:

- The output sequence of a linear shift register is periodic and if there are n registers the maximum period is $2^n - 1$.
- The output sequence of a linear feedback shift register on a nonzero input has maximum period if and only if, its characteristic polynomial is primitive.
- If a linear shift register with n registers has maximum period $2^n - 1$, then any output sequence of length $2^n - 1$ has the following properties:
  - it contains exactly $[2^{(n-1)} - 1]$ zero’s and $[2^{(n-1)}]$ones;
  - for any block of length t bits, defined as a sequence of the form 011…10 containing exactly t bits, where 1 <= t <= (n-2), it contains, $2^{(n-t-2)}$ blocks of length t and the same number of gaps, defined as a sequence of the form 100…01.

The use of output sequences from a linear feedback shift register as a pseudo-random one-time pad is insecure against known plaintext attack.

4. Block Ciphers as Possible Generators

Currently there are no stream ciphers (a sequence of random bits generated one at a time) that have emerged as a de facto standard. One of the most widely used stream cipher is RC4 [15] and [17]. Certain modes of operations of block ciphers (a sequence of random blocks generated n-bits at a time) [10] and [17] can effectively transform it to a key stream generator. Thus a block cipher may be used as a stream cipher.

Matrix Transformations: An analogy, is Hill's 1929 [17] and [19] matrix transformation method (which is not a cipher method), working on a block of n x n plaintext characters, multiplying by an encoding matrix, E, of same size n x n, to give a ciphertext n x n matrix, which is transmitted sequentially per character basis. The receiver correspondingly, multiplies the ciphertext c x n matrix, with the inverse of matrix, E to recover the plaintext. Data Encryption Standard DES [13] and [17] in cipher-feedback mode (CFB), or (output-feedback mode) OFB also can be used. Stream ciphers are inherently much faster than block ciphers.

DES: DES uses block ciphers, typically 128 bits; however, the block cipher should not be a mathematical group structure [14] and [17]. Likewise any block cipher designer avoids a group structure. This avoids the "meet-in-the middle attacks" [2] and [17].

Group Weakness: A mathematical group structure ensures closure, following mathematical operations. If a message is encoded using one key (k1), then this encrypted message further encode by another key (k2), we may think we have a very secure system. If k1 and k2 are part of the a group, then there is no advantage in the second encoding, as this can be shown to be equivalent to encryption by a single key (k3). By group closure properties, k1*k2 = k3 (where "*" is the group operation, maybe addition, multiplication, exponentiation, modulo (n), etc). Triple DES or 3*DES is a more secure version of DES. This first encrypts the plaintext with key k1, then decrypts this with key k2, then re-encrypts with a third key k3. There are some variations such as, k1=k2=k3, or k1=k3 etc. [17] and [19].

5. Users of One-Time Codes

One-time codes are used in applications where very crucial messages are interchanged that require the highest level of security. Example of the interchange of such messages include but are not limited to military information, sensitive government information, where interception could have very severe ramifications, such as in a war, terrorism, or some other major catastrophe, require the highest level of security. These are best protected by one-time codes. The enormous efforts required in generation and ensuring both receiving and

transmission parties in the communication are synchronized to the correct one-time codes (or one-time pads) is justified in such situations.

Scalability Issues: The difficulty of the process of generation, distribution and synchronisation of one-time codes does not make them easily scaleable as the number of users communicating, increases. We cannot extend the same concept to worldwide exchange of information, as the generation and key interchange amidst highest security would be a very complex process and grow exponentially [23] with number of users in the system.

Control Requirements: The one-time codes must be generated, and deactivated, with utmost caution to ensure complete randomness. This requires that central authorities control their use, so that the same set of one-time codes are not multiply used. Hence in common day-day areas like internet banking, public key cryptography systems are used. These are less secure, but are more practical, as individual users can randomly generate their own keys, have a mutually, acceptable key exchange protocol and verify with a digital signature authority, and start communications, with very little input from the central authorities. The public key cryptography method is easily scaleable as the number of users increasing in the system. Also, each user session has multiple message transfers and the duration of the session can be very long, such as several internet banking transactions, long GSM mobile calls [24], [25] and [26], and internet “chat” sessions.

RFID tags have a short “application life”. Hence RFID tags are very suited to the use of one-time codes, due to the need of lightweight xor computations. Also the limitation of the RFID tag makes the message short compared with the internet applications above. Mobile and other wireless devices have greater compute and signal processing capabilities to ensure the messages are sufficiently well coded, unlike RFID devices where the computing and bandwidth is very limited. At physical wireless medium layer, unlike mobile GSM, or bluetooth, it is not possible to perform frequency hopping as a very rudimentary guard against an eavesdropper.

5. One-Time Codes and One-Way Functions

Before further discussions on applications of one-time codes, the difference with a closely related area with similar mathematic basis, one-way functions (also known as one-way hash functions) is provided. In an ideal situation, ideal secrecy, truly random one-time codes are very different from one-way functions [17] and [20].

One-time Codes: One-time codes share many common properties with one-way functions, but there is a crucial difference. Most cryptography applications based on one-way functions have some form of trap door one-way functions (TOF), while one-time codes do not have a TOF [6], [8], [17], and [19]. As an example the one-time function maybe exponentiation (mod (n)), where the one-way function is exponentiation and the trapdoor is modulo (n). Unless the

intruder knows the exact exponential power, they cannot recover the value. The modulo (n), is an extra piece of information (also called trap door), which helps receiver decrypt the message. One-way functions are such that the forward process, encryption should be easy, but the reverse process, decryption should be difficult, and impossible or computationally intractable for an intruder.

One-way Functions: Unlike one-time codes, one-way functions are not truly random in the bit stream generated. One-way functions generate pseudo-random bit patterns. Their advantage lies in shorter logic, and functional operations to generating the bits the plaintext is xor'd bit shifted with to produce the ciphertext, C. One-time codes cannot be generated by any function as above (generated pseudo-random numbers), but are generated as a purely random stream at least of the same length as the plaintext, P.

One-Way functions are also known as trap-doors, (analogy, it should be easy to open and get inside, but difficult to come out). There is vast literature on hash functions which are very extensively used. If hash functions are supplemented with some extra constraints to ensure collision-free codes we obtain, one-way hash functions.

Requirements: One-way hash functions are fairly extensively used in cryptography. A hash function, H, is defined, by h = H(m), where m is the message, and h, is the hash value.

The requirements for cryptographic hash functions are [17]:

(a) The input may be of any length.
(b) The output must have a fixed size.
(c) Hash function value z = H(k) is easy to calculate, for any given k.
(d) H(m) is one-way.
(e) H(m) is collision free.

A hash function is one way if it is hard to invert. That is, it should be infeasible, computationally, to obtain,

$$k = z^{-1}, \text{ or } H^{-1}(k) \text{ , where } z = H(k).$$

Hash functions also must be collision-free as far as possible, in cryptography applications. A weakly collision-free hash function is one, when given a message,
m, it is computationally infeasible to another message, n, different to m, such that H(m) = H(n). A strongly collision-free hash function, H, is one, where it is computationally intractable to find any two messages, m, and n, such that H(m) = H(n).

6. Management of Codes

There are various issues with respect to code (sometimes called key) management [17] which are important, and while not directly related to one-way codes, or RFID systems, may be worth noting:

- How does one store the codes? The weakness of some systems is that the data storage of encrypted message, and the code, is on the same disk.
- Where are duplicate or backup codes kept?
- In the choice of codes, is there sufficient "difficulty", eg: a simple name, date of birth easy for someone to guess, contrasted to a complex combination of letters and numbers.
- Other issues are, code or key escrow, key management, compromised key, lifetime of keys etc. The more complex key management systems eg: systems in public key cryptography use a layered key structure. They use a master key, or key-encryption-key (KEK). Users encrypt keys using the KEK [17].The master keys are most securely delivered to the main exchanges. In some ways they are equivalent of one-time codes, at the highest layer, as detailed below.

ANSI X.17 [16] - Financial Institute Key Management (wholesale) standard is used for public key internet traffic. Methods of distribution of secret keys, is on a layered approach. Financial Institutions exchange their keys, on a daily, or per-session basis based on the volume of encryptions performed. Due to the high volumes, the most secure method, namely, manual distribution and transfer of keys is considered fairly inefficient, and expensive. Hence there is a three-tiered hierarchy of key distribution:

- Highest level is the master key (KKM). This is always manually distributed for maximum security.
- The next level consists of key-encrypting keys (KEKs). These are distributed on-line.
- The lowest level is data keys (KDs). These are also distributed on-line.

KD – data keys are used for bulk encryption, for users, and changed per session basis, and are encrypted using the KEK keys.

KEK- are changed periodically and changed encrypted with the master key

The KKM- master key is changed less often, but must always be distributed in the most secure manner, hence always distributed manually. In some ways KKMs are equivalent to one-time codes, to guarantee highest level of security.

In the case of one-time codes, we do not require duplicate or backup keys. Hence the above issues need not be considered. There is an issue, though, what if the receiver loses their one-time key, or goes out of synchronization? Is it possible to recover the one-time code? The very nature of the one-time system, with backup, may defeat its purpose.

8. A Design for Random One-Time Codes for RFID

Jantscher [6] presents an extremely lightweight mutual authentication scheme for RFID based on random tag identifiers and XOR-padding.

Objectives: The main objective of the scheme described herein is to provide privacy in terms of preventing tags from being traced and security in terms of mutual authentication between RFID readers and tags.

System Model: The general model consists of an RFID reader with an interface to be connected to a computer or a network and RFID tags. For the considerations in the security and privacy context a few details should be added to this general model. The detailed model consists of a backend database, a reader and tags. Communication between reader and tags is conducted via a wireless RF-link. This wireless communication channel is assumed to be insecure. The reader is connected to the backend database wired or wirelessly. This connection is assumed to be secure. Taking into account modern network security measures based on established cryptographic primitives this assumption ought to be legitimate. Furthermore, a majority of other works in the privacy and security context of RFID is also based on the same premise. Therefore, the following considerations solely address security and privacy issues of the RFID air-interface whereas reader and backend database are treated as single entity.

A Simple Scheme: As mentioned above, the backend database and the reader are treated as single entity which communicates with a tag via the RFID air-interface. The scheme uses three different encryption strings (T, R, and A) which are used in conjunction with an XOR operation to disguise transmissions of identifiers. These encryption strings are random binary sequences that are shared between each individual tag and the backend database.

During the **setup-phase** of the scheme (phase before first identification of a tag) the backend database, for each tag, generates five random numbers which are stored in the database as well as in the tag. The first number represents the current identifier of the tag, denoted curID in the figure above. The second number represents the first old identifier denoted oldID. The remaining three numbers are stored as encryption string for the current identifier (T), encryption string for the new identifier (R), and encryption string for the old identifier (A). For efficient look-ups during the identification-phase the sequence curID XOR T should be also stored in the database. Finally, the database stores the allocation between tags and items the tags are affixed to or tags and EPCs respectively.

Fig 1 shows a single protocol run during the **identification- and update-phase** of the scheme. The symbol ♦ represents the XOR operation. The example shows the identification of a single tag without consideration of the anti-collision mechanism which would be required to identify a single tag out of a group of tags. The shown procedure, however, can easily be applied after an anti-collision mechanism in which case the Query might be replaced by another command (e.g. an ACK command if using the EPCglobal Class 1 Generation 2 air-interface [3]).

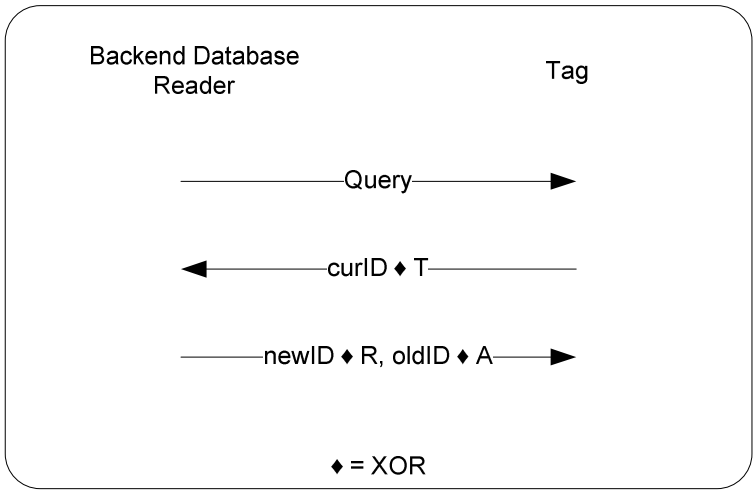


Fig 1: Basic XOR Authentication Scheme

The **identification-phase** starts with a query command issued by the reader. In response to this command the tag replies with its current identifier (curID) disguised by the encryption string for current identifiers (T) which is XORed with the curID. In order to identify a tag the database just needs to look-up the received sequence. If this sequence could be found the tag is assumed to be legitimate and has successfully authenticated to the backend database.

During the **update-phase** a message including two sequences is sent by the reader. The first sequence represents the new tag identifier (newID), randomly generated by the backend database, XORed with the encryption string for new identifiers. The second sequence represents the old tag identifier (oldID), which was used as curID during the previous identification, XORed with the encryption string for old identifiers. The database exchanges the old identifier by the current identifier and stores the new identifier. The tag, upon receipt of the update-message, compares the old identifier sent as second part of this message to its stored value. If the comparison matches the old identifier is replaced by the current identifier and the new identifier, which is included in the first part of the update-message, is stored as the current identifier. This two-stage update procedure ensures that tag identifiers can be updated by legitimate backend-databases and readers only since external devices do not have knowledge about old identifiers and the corresponding encryption strings. Therefore, during the update-phase the reader authenticates to the tag.

Production of Uncertainty – Decoys: Decoys are special tags deliberately introduced into a system in order to confuse eavesdroppers. Decoy tags, just as ordinary tags, participate in singulation as well as identification. They, however, respond with completely random sequences. Legitimate readers are able to detect responses from decoys because they don't find the replied sequences in the backend database but eavesdroppers cannot differentiate between replies from decoys and legitimate tags. With decoys having access to the backend database it can be guaranteed that the random replies do not appear in the database.

Analysis – Privacy: The main goal of privacy protection in RFID is to provide 'unlinkability', i.e. non-traceability of items bearing RFID tags. An obvious approach to attain this goal is to have tags with changing identifiers. For attackers the various identifiers of a single tag appear like random sequences whereas legitimate readers (and backend databases

respectively) still are able to clearly link these identifiers with a certain tag and therefore with the item this tag is affixed to. There are two groups of methods for changing tag identifiers, tag-driven ID-update and reader-driven ID-update.
The group of reader-driven ID-update mechanisms comes along with a general privacy weakness. Although tag identifiers are updated by legitimate readers, they are not updated by active attackers which allow such attackers to trace tags from the time of a legitimate identification to the time of the next legitimate identification.
Another weakness of the above described scheme is that there is no response from the tag after its identifier was updated. Hence, the reader does not know if the update was successful. If a tag identifier is not updated, the tag will respond with unchanged identifier during the next query. This presents a threat on unlinkability. Reasons for unsuccessful updates might reach from flipped bits in the update message to intended disruptions of the update message by active attackers.

Analysis – Security: Besides the discussed privacy weaknesses there are also a security weaknesses in the scheme allowing attackers to change tag identifiers without actually knowing the encryption strings (T, R, and A). In order to allow an attacker to authenticate to a tag and to change its identifier it has to know the authentication sequence oldID XOR A. Since the encryption strings (T, R, and A) as well as the random identifiers are secret, attackers normally do not have knowledge about the authentication sequence. However, there are two security weaknesses that allow attackers, in the first case, to construct the authentication sequence and, in the second case, to record and replay this sequence. Since sequences in the original scheme are just encrypted using XOR padding and identifiers at different times are XORed with different encryption strings, attackers may exploit the properties of the XOR function in order to create new, useful sequences. The example following shows how an attacker can create the useful sequences A XOR T, R XOR T and R XOR A just by listening to three successive runs of the authentication scheme. These sequences can be used to change tag identifiers.

First, let us assume a single tag which is identified and updated by legitimate readers at several points. An eavesdropper which is able to capture three successive and successful authentication runs learns the following sequences. Short notation is used from here. For example, the authentication run in Fig 2 using short notation is “Q curID♦T newID♦R oldID♦A”. Same indices indicate same identifiers, e.g. $newID_3 = curID_3 = oldID_3$.

Q $curID_2$♦T $newID_3$♦R $oldID_1$♦A
Q $curID_3$♦T $newID_4$♦R $oldID_2$♦A
Q $curID_4$♦T $newID_5$♦R $oldID_3$♦A

Fig 2: Authentication Sequences for 1 Tag

With knowledge of the three successive authentication runs in Fig 2 the attacker simply applies XOR operations to generate the useful sequences like in Fig 3.

A♦T = (oldID2♦A) ♦ (curID2♦T)
R♦T = (newID3♦R) ♦ (curID3♦T)
A♦R = (oldID3♦A) ♦ (newID3♦R)

Fig 3: Derived Sequences from Fig 2

Next, let us have a look what happens if more than one tags are identified and updated at the same time. Since the order of the identifications of tags is not deterministic, the attacker faces the problem of not knowing which sequences belong to which tag. In the following example Fig 4 three tags form a group which is identified and updated at different points. An eavesdropper which is able to capture three successive and successful authentication runs for the whole group learns the following sequences. The letters A, B, and C are indicators for the three different tags.

$Q\ curID_{A2}♦T_A\ newID_{A3}♦R_A\ oldID_{A1}♦A_A$
$Q\ curID_{B2}♦T_B\ newID_{B3}♦R_B\ oldID_{B1}♦A_B$
$Q\ curID_{C2}♦T_C\ newID_{C3}♦R_C\ oldID_{C1}♦A_C$

$Q\ curID_{B3}♦T_B\ newID_{B4}♦R_B\ oldID_{B2}♦A_B$
$Q\ curID_{A3}♦T_A\ newID_{A4}♦R_A\ oldID_{A2}♦A_A$
$Q\ curID_{C3}♦T_C\ newID_{C4}♦R_C\ oldID_{C2}♦A_C$

$Q\ curID_{A4}♦T_A\ newID_{A5}♦R_A\ oldID_{A3}♦A_A$
$Q\ curID_{C4}♦T_C\ newID_{C5}♦R_C\ oldID_{C3}♦A_C$
$Q\ curID_{B4}♦T_B\ newID_{B5}♦R_B\ oldID_{B3}♦A_B$

Fig 4: Authentication Sequences for a Group of 3 Tags

In order to construct the useful sequences A♦T, R♦T, and A♦R the attacker first needs to figure out which of the above lines belongs to which tag. Hence, the attacker needs to compare identifiers. Since identifiers are always disguised using encryption strings it is not possible to directly compare them, but, exploiting the properties of the XOR operation again, pairs of identifiers can be compared.

If an attacker looks for the sequences belonging to one tag in the above example it just needs to XOR the parts including the current identifiers of the first two groups of authentication sequences and the parts including the old identifiers of the second two groups of authentication sequences and compare the resulting strings.

$curID_{A2}♦T_A♦curID_{B3}♦T_B$
$curID_{A2}♦T_A♦curID_{A3}♦T_A = curID_{A2}♦curID_{A3}$
$curID_{A2}♦T_A♦curID_{C3}♦T_C$

$curID_{B2}♦T_B♦curID_{B3}♦T_B = curID_{B2}♦curID_{B3}$
$curID_{B2}♦T_B♦curID_{A3}♦T_A$
$curID_{B2}♦T_B♦curID_{C3}♦T_C$

$curID_{C2}♦T_C♦curID_{B3}♦T_B$
$curID_{C2}♦T_C♦curID_{A3}♦T_A$
$curID_{C2}♦T_C♦curID_{C3}♦T_C = curID_{C2}♦curID_{C3}$

Fig 5: XORed Tag Responses from Fig 4

$$oldID_{B2} \blacklozenge A_B \blacklozenge oldID_{A3} \blacklozenge A_A$$
$$oldID_{B2} \blacklozenge A_B \blacklozenge oldID_{C3} \blacklozenge A_C$$
$$oldID_{B2} \blacklozenge A_B \blacklozenge oldID_{B3} \blacklozenge A_B = oldID_{B2} \blacklozenge oldID_{B3}$$

$$oldID_{A2} \blacklozenge A_A \blacklozenge oldID_{A3} \blacklozenge A_A = oldID_{A2} \blacklozenge oldID_{A3}$$
$$oldID_{A2} \blacklozenge A_A \blacklozenge oldID_{C3} \blacklozenge A_C$$
$$oldID_{A2} \blacklozenge A_A \blacklozenge oldID_{B3} \blacklozenge A_B$$

$$oldID_{C2} \blacklozenge A_C \blacklozenge oldID_{A3} \blacklozenge A_A$$
$$oldID_{C2} \blacklozenge A_C \blacklozenge oldID_{C3} \blacklozenge A_C = oldID_{C2} \blacklozenge oldID_{C3}$$
$$oldID_{C2} \blacklozenge A_C \blacklozenge oldID_{B3} \blacklozenge A_B$$

Fig 6: XORed Reader Authentication Strings from Fig 4

The attacker finds that equal strings appear in the sequences in Fig 5 and Fig 6 (e.g. $curID_{A2} \blacklozenge curID_{A3} = oldID_{A2} \blacklozenge oldID_{A3}$). These equal strings betray the lines in Fig 4 which belong together. It is the lines the equal strings resulted from. For example, $curID_{A2} \blacklozenge curID_{A3} = oldID_{A2} \blacklozenge oldID_{A3}$ betrays that line 1 in the first group, line 2 in the second group, and line 1 in the third group of the authentication sequences in Fig 4 belong to one tag. Having this information the attacker can again apply the method described in Fig 3 to derive the useful sequences.

After the attacker created the set of useful sequences for a tag it can use it in various ways. First, knowing the sequence R♦T allows the attacker to convert between reader-update-strings and tag-response-strings (curID♦T = (newID♦R) ♦ (R♦T)) and therefore allows subsequent tracking of tags. Second, knowing the sequence A♦T allows it to change the tag identifier (oldID♦A = (curID♦T) ♦ (A♦T)) and therefore to mount a denial of service attack. Once, a tag identifier was changed by an attacker, legitimate readers no longer are able to identify the corresponding tag. This means the tag cannot be updated any more which again allows the attacker to track it.

Another attack is based on the fact that an authentication session ends with the update message of the reader. As already mentioned in the privacy section of this analysis, this means that readers cannot be sure that the update was successful. Hence, it is possible that same identifiers appear at different identifying points and privacy cannot be guaranteed. Moreover, this allows attackers to record the authentication sequence (oldID♦A) and to reuse it for an ID-update in the case the update of the legitimate reader was unsuccessful (replay attack). Admittedly, this approach is based on trial-and-error but leads to a more sophisticated attack, the so-called relay or man-in-the-middle attack [7]. Assuming an attacker which is able to mount a relay attack, in the scheme described in Fig, this attacker, in order to change a tag identifier to an arbitrary value, just needs to alter the ID-update sequence (newID♦R) and forward all other messages. Juels mentioned that relay attacks generally are unavoidable but first approaches addressing them exist like the distance bounding protocol [5].

In the following section approaches are presented that address various weaknesses of the simple scheme.

Addressing the XOR Property Weakness: One of the major weaknesses of the original scheme is that an attacker just by listening to successive authentication runs can construct sequences that subsequently allow it to track tags and to change tag identifiers respectively. The reason for this weakness lies in the fact that identifiers are used three times (newID, curID, and oldID) and therefore the XOR operation can be applied to cancel them out. To address this weakness the identifiers need to change each time they are transmitted. In order to change identifiers this approach utilizes pseudo-random sequences which are XORed with the original identifiers. The intended generators for these sequences are linear feedback shift registers (LFSRs) or even better nonlinear feedback shift registers (NLFSRs). Linear feedback shift registers, which are also the building blocks of NLFSRs, can be implemented in hardware very efficiently. The estimated gate count is 12 times the bit-length of an LFSR [1]. In the following the term pseudo-random number generator (PRNG) is used to denote either of LFSRs or NLFSRs.
The approach enhances the original idea by introducing two independent PRNG which are used to generate strings (S_T and S_A) used for changing the current identifier and the old identifier. In the original scheme the new identifier, the current identifier, and the old identifier of three successive authentication runs are equal. By changing curID to curID* = curID♦S_T and oldID to oldID* = oldID♦S_A this equality and therefore the weakness can be eliminated.

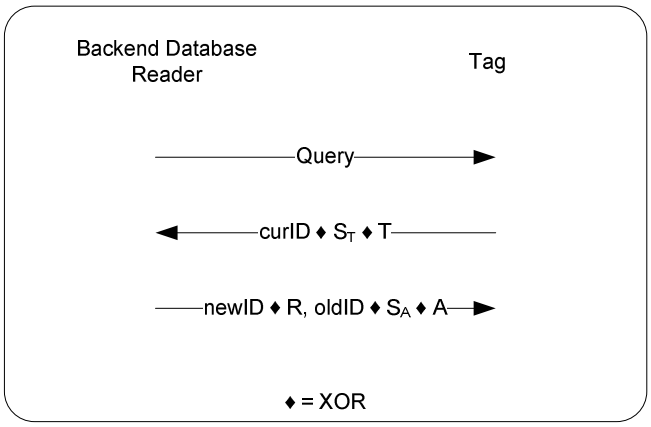


Fig 7: Strengthened XOR Authentication Scheme

Fig 7 shows the approach introducing S_T and S_A. In order to setup the strengthened scheme in addition to the original scheme two independent (different) PRNGs have to be implemented in each tag and at the backend database and two different initial seed values for the PRNGs have to be stored in both places too. During the identification phase the PRNG used for generating S_T is initialized with the stored seed value and the sequence S_T is generated (length of S_T = length of curID). Prior to sending the current identifier it is XORed with S_T and T. During the update and authentication phase the PRNG used for generating S_A is initialized with the stored seed value and the sequence S_A is generated (length of S_A = length of oldID). The old identifier is XORed with S_A and A and compared to the received authentication sequence. If both are equal the current identifier is stored as old identifier and the received new identifier is stored as current identifier. Additionally, the current statuses (seeds) of the PRNGs have to be stored in a non-volatile memory.

Since two independent PRNGs are used for generating the two strings S_T and S_A there is no correlation between these strings and no useful information can be gained from XORing

sequences including the same identifiers e.g. ($curID_2$♦S_{T2}♦T) ♦ ($oldID_2$♦S_{A2}♦A) = S_T♦S_A♦T♦A since $curID_2$ = $oldID_2$ as this was the case in the original scheme. Also XORing subsequent kindred sequences does no longer lead to any useful information (e.g. ($curID_2$♦S_{T2}♦T) ♦ ($curID_3$♦S_{T3}♦T) = $curID_2$♦$curID_3$♦S_{T2}♦S_{T3}) since the identifiers $curID_2$ and $curID_3$ are random.

What was not mentioned so far is the required size of the PRNGs. If we consider maximum length LFSRs the period length would be 2^n-1 where n is the number of bits (the number of stages) of the LFSR. It should be ensured that there is no overflow of a PRNG which would mean that the same sequences are generated again. The period length depends on the length of the identifiers to be disguised (today, 64 and 96 bits are common identifier lengths used in the EPCglobal Network) and the total number of identifications. A 16 bit maximum length LFSR would allow about 680 identifications without PRNG overflow assuming a 96 bit identifier. Doubling the number of stages to 32 bit leads to nearly 45 million identifications.

RFID tags often include CRC (cycling redundancy check) modules used for error detection. Such a CRC module basically consists of an LFSR without a dedicated output from the last stage but with an input for the sequence to be checked at the first stage. Again, considering LFSRs for random number generation, in order to attain maximum hardware efficiency, it would be possible to reuse one LFSR with reconfigurable taps for CRC as well as for both PRNGs.

Addressing the No Tag Response after Update Weakness: Another weakness is that there is no response from the tag after its identifier was updated successfully. Generally, a simple tag response like a predefined success or error code would address this problem. The reader would know if the update was successful or not and could take an appropriate action like starting another update attempt in the case of the former attempt was unsuccessful. A simple, unencrypted response sequence, however, allows attackers to reproduce it easily and deceive legitimate readers by transmitting valid success or error codes.

Hence, in order to prevent attackers from deceiving legitimate readers by sending valid tag responses it has to be ensured that valid responses can only be generated by legitimate parties, i.e. tags belonging to the system.

Although the described enhancement prevents replay attacks it does not solve the problem of relay or man-in-the-middle attacks which remain an open problem.

Addressing other weaknesses: In the presentations of the original and strengthened scheme it was not yet mentioned that it is important to take care of errors like flipped bits during transmissions. If for example a transmission error (e.g. flipped bit) occurs in the update sequence (newID ♦ R) the tag and the database would store different identifiers and the tag subsequently can no longer be identified. Hence, an error detection mechanism like CRC16 should be used to protect all sequences that are transmitted over the air interface.

In the original scheme the use of decoys was mentioned in order to add security. Decoys, however, do not prevent from the attack described in Fig and Fig. Since the strengthened scheme does not reveal any useful information for eavesdroppers decoys do not provide any additional security.

So far, it was assumed that identifiers at the backend database are updated along with the transmission of the update and authentication sequences, i.e. only one database entry is

maintained per tag storing the current sequences assigned to the tag. If, however, an update is unsuccessful the tag will reply with the old identifier next time it is queried and the database would be unable to identify it (since the identifier in the database was already updated). Therefore, it should be considered to maintain two database entries for each tag, one storing the sequences before the last update and one storing the new sequences after it. The database entry storing the old sequences can be deleted right after the tag was successfully identified with its new ID next time.

Finally, it should be considered that with using the pseudo-random sequence S_T for disguising tag identifiers it cannot be guaranteed that the resulting identification sequence curID ♦ T ♦ S_T is unique. Hence, the backend database, prior to updating a tag with a new identifier, is required to check this identifier against collisions. A similar problem arises with external tags, e.g. tags introduced by an attacker. The identifier of any external tag might collide with an identification sequence curID ♦ T ♦ S_T stored in the database leading to a positive identification of a tag not belonging to the system. Therefore, sufficiently long identifiers (e.g. 96 bits) should be considered in order to minimize the chances of collisions by having a very sparse populated database.

9. Other research, in Random One-Time Codes

There are diverse research areas in the generation of one-time codes. Quantum Cryptography [8] is one such area. This is based on the quantum states of spin, other variables, of subatomic particles like photons, or electrons. This area is currently used in public key cryptography, only, to change the keys at regular intervals during a session, where otherwise the keys are static. This random changing of keys gives a little extra security to the session. It is not yet feasible for full data encryption. The application of random noise in space, galaxies, via radio astronomy is another promising area of one-time codes applications. As is true with any one-time code, both parties must be in synchronization, using the same one-time code (or pad). Having something from nature has the advantage both parties have the same set of variables, and hence generate the random one-time code, at the same time. Each party in the communication must keep their clocks in synchronization, to note the exact time of the events, while storing the episodes or the occurrence or the data bank of random codes. Another suggested direction is [18] use of 2-factor authentication. One part is the fixed part like password, another dynamically changing such as time of day or some variable parameter to make any attack by an intruder difficult.

A D Wyner (1975), brought together information theory, codes, and cryptography, in "Wyner's wiretap channel" [20]

Alexi, Chor, Goldreich, and Schnorr (1984) [4] and [20] have developed some of the most efficient and secure systems, and are based on computationally intractable nature of factoring.

Welsh [20] further shows the relationship between random number generators, and one-way functions. Given the output of such a generator, it is computationally infeasible to find out anything about the seed in any reasonable (polynomial) amount of time.

Welsh [20] also cites Yao's "effective entropy" concept. Given a RSA ciphertext message C, sent from message M, the actual entropy, H(M | C) = 0. This is because there is no uncertainty as we know the message as well as the ciphertext.

However, to the intruder, the computationally intractable problem to factorize the public key, K, results in an uncertainty. This is termed, "effective uncertainty" or "effective entropy" (Yao, 1982) in Welsh [20].

Merkle-Hellman Knapsack Problem is a very practical [20] solution to the problem of generation of pseudo-random sequences. This is an interesting method, using combinatorial mathematics where the easy part, choice of a random number, is used in the encryption, while the more difficult part (equivalent to the factorisation problem) is used in the decryption [20]. Again, whether in the longer term, this is a possible candidate to replace the current nearest or best choice, stream ciphers for one-time codes, needs more research. Aldeman, and Shamir have shown via “attacks”, in polynomial time, certain weakness of the Merkle-Knapsack problem, which belonging to the NP class of problems, was expected to be harder to attack than even RSA [20]. There is scope perhaps refining the original problem inherently NP (exponential time) [20], so as to generate cryptographically secure one-time codes.

Quantum key generation, based on the polarization of the transmitted photon can be shown to be a one-time code. By Heisenberg’s uncertainty principle an eavesdropper cannot listen to the message without making errors, as in the process of observation the eavesdropper would have changed the state of the photon [8]. It also follows from Heisenberg’s uncertainty principle, that in a quantum cryptography environment, an eavesdropper, even a passive listener, can be very easily detected [8]. The bit stream sent from one authentic user to another is not crucial, as long as they agree on some subset of bits as a secure key. Quantum cryptography, allows distribution of keys between two users (who have not shared any prior keying material) that is very secure. This requires the users to have a conventional channel aside from the quantum channel [8]. An intruder with unlimited computing power cannot attack the message transfer. Quantum cryptography offers one of the best sources of one-time codes.

10. Conclusions

While one-time codes are the most secure means of encryption, in practice it is very difficult to have random one-time code generators. In RFID systems, due to the publicly available wireless medium information, and shorter lifetime, one-time codes would be ideal for security. In RFID systems, one-time codes require simple xor operations for encryption and decryption. This paper has highlighted some of the issues in the deployment of one-time codes in RFID applications, and has shown that there is considerable scope for research in

the generation, synchronization, and recovery methods in the use of one-time codes, in RFID applications.

Importantly, this paper presents a lightweight, but potentially very secure authentication RFID system, based on the use of one-time codes.

References

[1] Batina, L., Lano, J., Mentens, N., Oers, S., Preneel, B. and Verbauwhede, I. (2004), Energy, performance, area versus security trade-offs for stream ciphers: in Proceedings of the Workshop on The State of the Art of Stream Ciphers; Ecrypt; Brugge; 2,004.

[2] Campbell, K.W., and Wiener, M.J. (1993), DES is not a group, Advances in Cryptology - Crypto '92, Springer-Verlag (1993), 512-520.

[3] EPCglobal (2001), EPC Radio-Frequency Identity Protocols, Class-1 Generation-2 UHF RFID, Protocol for communications at 860 MHz – 960 MHz, Version 1.0.9; January 2005; Retrieved May, 12, 2006 from http://www.epcglobalinc.org/standards_technology/ratifiedStandards.html.

[4] Goldreich, Oded (2001), Foundations of Cryptography, Cambridge University Press, Cambridge 2001, ISBN 0-521-79172-3

[5] Hancke, G. and Kuhn, M. (2005), An RFID Distance Bounding Protocol; in Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks; IEEE Communications Society; Pages 67-73; 2005.

[6] Janstcher, Manfred (2006), Use of Random and Varying Codes in Object Identification, Auto-ID Labs, School of Electrical and Electronics Engineering, University of Adelaide, Adelaide, Australia.

[7] Juels, A. (2005), RFID Security and Privacy: A Research Survey; RSA Laboratories; 2005; Retrieved May, 26, 2006 from http://www.rsasecurity.com/rsalabs/staff/bios/ajuels/publications/pdfs/rfid_survey_28_09_05.pdf

[8] Mollin, Richard (2000), Introduction to Cryptography, Chapman & Hall/CRC, Boca Raton, 2000, ISBN" 1-58488-127-5

[9] Nature-Inspired Cryptography: CEC 2004 Special Session on Evolutionary Computing and Computer Security. http://www.cs.york.ac.uk/security/LibraryPages/NatureInspired.html

[10] http://www.rsasecurity.com/rsalabs/node.asp?id=2168 - Block Cipher.

[11] http://www.rsasecurity.com/rsalabs/node.asp?id=2174 -steam ciphers

[12] http://www.rsasecurity.com/rsalabs/node.asp?id=2187 - P, NP classes

[13] http://www.rsasecurity.com/rsalabs/node.asp?id=2228 -DES

[14] http://www.rsasecurity.com/rsalabs/node.asp?id=2230 - DES is not a group

[15] http://www.rsasecurity.com/rsalabs/node.asp?id=2250 -RC4 cipher

[16] http://www.rsasecurity.com/rsalabs/node.asp?id=2306 - ANSI X9 key mgt

[17] Schneier, Bruce (1994), Applied Cryptography: Protocols, Algorithms and Source Code in C, John-Wiley and Sons, New York, 1994. ISBN: 0-471-5975602

[18] Schneier, Bruce (2005),Two-Factor Authentication: Too Little, Too Late Inside Risks 178, Communications of the ACM vol 48(4), April 2005. http://www.schneier.com/essay-083.html

[19] Stinson, Douglas R. (1995), Cryptography: Theory and Practice, CRC Press, Boca Raton, Florida, 1995, ISBN: 0-8493-8521-0.

[20] Welsh, Dominic, Codes and Cryptography, Oxford University Press, Oxford. 1990, ISBN 0-19-853287-3.

[21] http://en.wikipedia.org/wiki/One-time_pad

[22] http://en.wikipedia.org/wiki/Vernam_cipher

[23] http://en.wikipedia.org/wiki/Stream_cipher -incl: RC4, A5/1, A5/2

[24] http://en.wikipedia.org/wiki/A5/1

[25] http://en.wikipedia.org/wiki/KASUMI - A5/3 block cipher, used in 3GPP

[26] http://en.wikipedia.org/wiki/A5/1#Attacks_on_A5.2F1_as_used_in_GSM

[27] http://en.wikipedia.org/wiki/Playfair_cipher - Joseph Mauborne, OT

