Identification and RFID

Cryptografy and Computer Security

Nejc Kišek

January 2017

Introduction

The goal of this paper is to look at how RFID devices are replacing barcodes for identification purposes, what security concerns they present and a comparison of new cryptographic protocols that were developed to protect them.

In the first section, we look at identification numbers and barcodes, and how they provide integrity with check digit schemes.

In the second section we describe the basic concepts of RFID, types of RFID tags and some of the limitations of the technology.

In the third section we examine security of RFID devices, describe the problems we need to prevent and some of the possible attacks.

In the fourth section we take a detailed look at four different protocols that can be used to address the concerns from previous section: The first one is based on AES encryption, the second is a new protocol that relies on hash functions, third is based on Elliptic Curve cryptography and the last is a variant of Rabin cryptosystem. We evaluate security and practicality of each protocol to see how suitable it is for use in RFID devices today or in the near future.

1 Identification numbers and barcodes

Identification numbers are used in many places around us and provide a quick way to identify items: products, books, documents, accounts, people... Most common examples are the Universal Product Codes (UPC), International Standard Book Numbers (ISBN), bank account numbers, driver license or ID card numbers. They are usually a string of multiple digits, letters and symbols.

1.1 Transmission errors

When identification numbers are read, copied or transmitted, there is a chance that mistakes will occur. Transmission errors need to be prevented, especially when we are dealing with important information such as bank account numbers. Two most common types of errors are:

- *Single-digit error* – a single digit is changed for a different one,
- *Transposition-of-adjacent-digits error* – two consecutive digits change places.

A study [3, page 5] showed that those two types account for more than 90% of all errors – about 80% for single-digit and about 10% for transposition-of-adjacent-digits error. Because of that, most error detecting schemes focus on those two types of mistakes. This is usually done by adding additional digits, called *check digits*, to the identification number, which are used to validate if the number has been changed.

1.2 Examples of check digit schemes

Universal Product code (UPC) uses a check digit scheme that catches all single-digit and most transposition-of-adjacent-digits errors. UPC consists of 12 numbers: first one denotes the type of product, digits 2-11 identify the manufacturer and the item itself and the last one is a check digit. The number is valid if it satisfies the following equation:

$$a_1 \cdot 3 + a_2 \cdot 1 + a_3 \cdot 3 + a_4 \cdot 1 + ... + a_{11} \cdot 3 + a_{12} \cdot 1 \equiv 0 \pmod{10}$$

where a_i is the i-th digit in the number.

This scheme will miss transpositions of consecutive digits when their values differ for exactly 5 ($|a_i - a_{i+1}| = 5$).

International standard book number (ISBN) uses a check digit scheme that catches all single-digit and transposition-of-adjacent-digits errors. It consists of 10 numbers: one for book version, next 3 for the publisher, following 5 for the book and the last one as a check digit. The number is valid if it satisfies the following equation:

$$a_1 \cdot 1 + a_2 \cdot 2 + a_3 \cdot 3 + a_4 \cdot 4 + ... + a_9 \cdot 9 + a_{10} \cdot 10 \equiv 0 \pmod{11}$$

where a_i is the i-th digit in the number.

There is a possibility that the check digit's value will have to be 10, in order to get 0 in mod 11. When this happens, the letter X is used as the last digit.

More examples and explanations of check digit schemes can be found in a paper by Tatjana Tomaš [5].

1.3 Barcodes

Barcodes were introduced as a quicker and more reliable way to read the identification numbers with an optical reader, instead of people reading and typing the numbers manually. Each digit is encoded as a combination of lines and can be decoded with a barcode reader.

UPC or ISBN identification numbers mentioned before can usually be found on products as barcodes. In terms of security, they only use check digit schemes to detect errors, and offer no mechanisms to prevent an attacker from reading or forging the number. They can also easily be copied.

2 RFID

Radio frequency identification (RFID) is a term used for technology that identifies objects via radio waves. The system consists of transponders or tags and receivers or readers that work in a way similar to how barcodes and barcode scanners work. The main difference is that RFID tags are not simply printed on paper, but are electronic devices and can offer (limited) computational power. For security this means that we can implement much more complicated mechanisms then barcodes can offer. On the other hand, there are also new problems: since RFID tags can be read remotely, attackers can read or track them much more easily.

2.1 History and applications

Origins of identification via radio waves can be tracked back to the second world war, where "Identify Friend or Foe" system on British Royal Air Force planes was used to distinguish friendly and enemy planes at long distances using radio frequency signals. Later, RFID tags similar to the ones we know today were used in transport – for automatic toll collection systems on highways, for identifying railroad cars and on shipping containers.

As the size and price of tags dropped, RFID technology could be used for many more applications. Today they are commonly used to track pets and livestock, items in warehouses, in building access control cards, for bus, train or subway system tickets, for credit cards, passports...

2.2 Hardware details

Tags are roughly divided in two big groups: passive and active tags. Active tags contain a source of power (a battery) and can transmit signals on their own, but are more expensive. They are used when long distance identification is needed, in environments with a lot of radiowave noise or when we want the tag to initiate the communication with the reader (and not the other way around). They are the most expensive and are most commonly used for tracking large objects (railroad cars, shipping containers).

A subset of active tags are also "semi-passive" tags, which have onboard battery, but are turned off until the reader initiates the communication. They are used for example in automatic road toll collecting systems.

Passive tags have no source of electricity, but are wirelessly powered by the reader via induction. This means that reader and tag need to be closer and the connection needs to be uninterrupted, but tags are much simpler and cheaper. They are used whenever we need inexpensive or very small tags.

There are also different frequencies at which tags can operate:

- Low frequency (LF), 120-140 kHz – Tags typically have operating range of less than 20cm and have low data bandwidth, but can work in proximity to metals and liquids. They are usually used for animal tracking implants or applications where the tag might be submerged in water. A typical manufacturing cost is around $1.

- High frequency (HF), 3-30MHz, most commonly 13.56 MHz – They have a range of up to 1 meter are used for access control, identity cards, contactless payment systems and *Near field communication* (NFC) systems. A typical manufacturing cost for a HF tag is around 0.5$.

- Ultra High Frequency (UHF), 868-928 MHz – They are most often used for item tracking, due to their long range (up to 20 meters) and low cost ($0.05 to $0.15). Communication at ultra high frequency also allows higher data bandwidth, but cannot pass through metals or liquids.

- Microwave and Ultra-Wideband, 2.4 GHz and above – They have the highest range, but consume more power and are much more expensive (around $5 for UWB and $25 for microwave).

In this paper we will focus on the cheapest (and therefore most widely used) type of RFID tags, which are small, passive and operate in the UHF range. They usually consist of an antenna, which is used for transmitting both power and information between the reader and the tag, and an integrated circuit which provides computation and storage. Antenna and circuit are sealed in protective plastic foil, a glass capsule, or similar material.

2.3 Limitations and problems

The main limitation of RFID tags is their cost – usually we can say they are feasible for large scale use if their price is around 5 cents or less. If we want to implement a secure protocol for identification on the cheapest tags, we have to work with approximately 3000-7000 logical gates. This number is slowly increasing as manufacturing costs go down, but there are also market pressures to lower the 5 cent mark to 4 cents.

Another issue that RFID technology raises is unauthorised tracking and privacy. RFID tags on someone's personal items or clothes can be read remotely, without the person being aware of it. Even if the data on the tags is just a number that itself has no meaning, the person can effectively still be tracked. To prevent this, information needs to be encrypted and messages from the tag should seem random to an unauthorised reader.

3 RFID security

There are three main concerns that we have with security of RFID systems:

- Unauthorised access to data on the tag – only a known and authorised reader should be able to read from the tag
- Tag forgery – attacker should not be able to impersonate or copy the tag just by listening to RFID messages
- Unauthorised tracking – attacker should not be able to indirectly identify the tag by observing patterns in its messages.

3.1 Types of attacks

Some examples of attacks that try to achieve the points mentioned above :

- Man in the middle attack – Communications between reader and tag are intercepted by the attacker. This can be a passive or an active attack – attacker can just listen to the data or also change it.
- Replay attack – attacker records the message from the tag and reuses it later, without necessarily knowing the contents.
- Unauthorised tracking attack – attacker does not get the exact content and identity of the tag, but can identify it using its past responses

- Relay attack – attacker relays messages between a tag and a reader to convince the reader that tag is nearby. For example, attacker establishes the communication channel between an RFID keycard in somebody's pocket and the receiver at a different location.
- Physical attack – instead of trying to determine the content using radio waves, content is read directly from the circuitry of the tag.

In this paper we will focus on the first three attacks, because they can be addressed by making changes to the logic behind the communication and require no changes to overall design of the RFID tags or readers.

Relay attack is difficult to prevent and needs a different approach as regular cryptographic protocols. There are several proposed distance bounding protocols, most of which try to fix this problem by precisely timing the communications between the reader and the tag.

Physical attacks can not be addressed by changing the communication protocols, but can be prevented by using tamper resistant chips or similar technologies.

4 Protocols

In this section we will take a look at some of the proposed RFID protocols for providing security against attacks from previous section. All of them are based on challenge-response mechanism, the first two use symmetric cryptography and the last two use public key cryptography.

A big focus in the described mechanisms is reducing the complexity, so that the implementations are cheaper. If the protocol fits in the previously mentioned interval of 3000-7000 logic gates, it can be implemented in a low cost RFID tag today or in the near future. More expensive tags with higher security can still be used for special cases, when the cost is not a big concern.

For each described tag, we will evaluate how well it addresses the three concerns from section 3 and how complex is its implementation.

4.1 Simple AES-based protocol

Protocol described by Feldhofer et al.[2] uses a simple challenge-response mechanism in combination with symmetric AES encryption to authorise the reader before revealing the tag's ID.

In the basic version, reader first sends a random challenge c to the tag. Tag encrypts it using AES with with a shared key and sends the result back to the reader. Reader can decrypt the message and verify the authenticity of the tag, based on the fact that it has the secret key.

The second version uses two-way authentication: reader sends a random challenge c_r to the tag. Tag generates another challenge c_t, encrypts both c_r and c_t and sends the result back. The reader can decrypt the message, verify that its challenge c_r is correct, encrypts the c_t and sends it back to the tag. Tag decrypts it and verifies, that the reader also knows the secret key.

The second version requires additional logic for generating a random challenge, as well as modification needed to perform additional steps needed (decryption and verification of the reader's response). Because of that, the paper mentioned above focused on implementing the one-way authentication. This AES implementation was different from the regularly used ones, because it optimised for simplicity instead of the speed. The resulting hardware implementation needed around 3600 logic gates, which is well within the reasonable complexity for a low cost RFID tag today.

Unauthorised access to data on the tag is prevented with AES encryption. Since AES is a well studied and widely used algorithm, we will assume it is secure.

Tag forgery can only be done if we know the secret key, otherwise we can not send a valid response to the reader. To prevent the replay attack, we need enough ($> 2^{80}$) different challenges, so that the attacker cannot create a table of all possible challenge/response pairs that could be used to impersonate a tag. This means that replay attacks are prevented if challenge is at least 80 bits long.

Tracking is not prevented in the first version of the algorithm – if attacker always send the same challenge, the response will always be the same. This could be prevented by adding a random number generator to the tag, but that would raise the same cost issues as in the secnd version of the algorithm.

In the second version, the responses include a random element c_t and appear random if not decrypted.

The first version of the algorithm is very inexpensive, but it does not prevent unauthorised tracking – it is therefore not suitable for all situations or as an universal standard. The second implementation is more secure, but might get too expensive to be useful on a large scale.

4.2 Lightweight non-deterministic protocol

Protocol described in an article by Trček and Jäppinen[6] uses a secret key that both reader and tag know and a hash function. It is non-deterministic, meaning that the response from the tag is not an exact value, but lies in a specified interval and the reader needs to check all possible values to find a match.

The protocol has three parameters set in advance: the shared secret s, size of the interval for random numbers n and a hash function H. It goes as follows:

1. Reader sends a timestamp t to the tag as a challenge.

2. Tag checks if the challenge is new, by comparing it against a list of previous challenges. If the challenge is valid, it is added to the list.

3. Tag generates a random number Δt from the interval $[0, n-1]$. Then it calculates $H(s||(t \oplus \Delta t))$ and sends the resulting hash to the reader.

4. Reader calculates hashes $H(s||(t \oplus 0)), H(s||(t \oplus 1)), ..., H(s||(t \oplus n-1)$ with s-es that are stored in the database as a pair (ID, s) for each tag. If it finds a value that matches tag's response, tag is recognised and authenticated.

The second step is used to prevent an attacker from tracking the tag by always sending the same challenge and remembering the responses. This memory can be a FIFO buffer as large as the cost of the tag allows.

The hash function in the protocol is implemented using DESL – a lightweight version of DES. Using block ciphers for hashing is more efficient in a hardware implementation than a dedicated hash function, which are very effective in software.

The random generator is implemented as a 8-bit shift register (LFSR). We do not want too many different random numbers, because reader will have to check all possibilities.

The whole protocol using 96-bit keys and timestamps with 4 places in the FIFO buffer can be implemented with approximately 4800 logic gates, which is within the reasonable limits for RFID tags. If the second step that checks for challenge freshness is omitted (applications where tracking is not a concern), this can further be reduced to 2400 gates.

Unauthorised access to data on the tag is prevented with a hash function based on DESL.

Tag forgery can only be done if we know the secret key s and the challenge t, otherwise reader will not find a match in the database. Since reader's challenges are timestamps we can assume they will not repeat and responses from old communications can not be reused. On the other hand this means that challenges can be predicted based on time and there is a possible scenario for a replay attack.

We have an active attacker who knows the timing of his attack in advance. He has access to the tag prior to that, can send challenges and record responses:

- Attacker sends timestamps to the tag and record the responses, to obtain multiple valid challenge/response pairs. He should choose the timestamps at and around the time when he plans to attack.

- Attacker waits until the reader sends him a challenge that he has a valid response to. He send the valid response and successfully impersonates the tag.

We can easily avoid this scenario by sending a random challenge instead on a timestamp. With 96-bit challenges, there is little chance of attacker recording the challenge/request pairs to encounter a repeat.

Tracking is prevented by randomising responses with Δt generated on the tag (which gives us 2^8 different responses) and by checking the challenge against the last 4 ones.

This can be circumvented if the attacker has access to the tag for a long time:

- He selects the challenge x and sends it to the tag. Then he needs to send 4 different challenges to clear the FIFO buffer, before sending x again to get another response. Each such try therefore requires 5 messages to be sent.
- He repeats the first step until he gets a list of all 2^8 possible responses from the tag. The expected number of tries should be asymptotically $\mathrm{O}(n \cdot \log(n))$ where n is number of possible responses (Coupon collector problem). This means around $5 \cdot 2^8 \cdot \log(2^8) = 10240$ tries. If scanning a tag takes 1 second, this would amount to less than 3 hours.
- He can then identify the tag by sending it a challenge x and seeing if the response is on his list of 2^8 possible responses.

This attack does not automatically work on all tags using this protocol, but the attacker that targets a specific one can track it. The solution for this problem is to increase the number of different possible responses, but this would also increase the time for the reader to identify the tag – if the number of different responses is so large tat even very powerful attackers could not try all of them, then the reader would also not be able to try all of them to find a match.

The protocol is lightweight and provides good secrecy and security against tag forgery if we change the timestamps for random challenges. The tracking protection is not perfect as shown in the example above, but tracking a tag is not trivial – attacker needs to have access to the tag for a fairly long amount of time before he can track it. This is certainly better than the AES protocol, but not good enough to be used in all situations.

4.3 ECC based authentication protocol

Protocol described in an article by Chen and others [1] uses elliptic curve cryptography to authenticate RFID communications. We need the following elements to implement the protocol:

- G – a group on a elliptic curve, $\mathrm{ord}(G) = q$,

- P – a primitive element of group G,
- $h()$ – a hash function that maps elements from G to a number in $\mathbb{Z}_q$,
- $H()$ – a hash function that maps a bit string to an element in G.

During the initialisation, reader's private key s and public key $Y = sP$ are generated. Then, for each tag i a private key d_i and an identification number $\mathrm{ID}_i = d_i P$ are generated. Each tag receives ID_i, P, Y and a timestamp t_i.

Authentication protocol:

1. Reader generates random number r and a timestamp t and broadcasts pair (r, t) to the tag.
2. Tag checks if the received timestamp is greater than its t_i, and if it is, generates a random number $k \in \mathbb{Z}_q^*$ and a random $R \in G$. It then computes $C_1 = r \cdot kP,\ C_2 = R + r \cdot kY$ and hashes $x_1 = h(R, C_1),\ x_2 = h(R, C_2)$. Then both x_1 and x_2 are divided by $\gcd(x_1, x_2)$. From that, tag generates three messages $B_1 = x_1 H(\mathrm{ID}_i),\ B_2 = x_2 H(\mathrm{ID}_i),\ B_3 = h(R, C_1, r)$. Message $(C_1, C_2, B_1, B_2, B_3)$ is sent to the reader.
3. Reader then computes $R' = C_2 - sC_1$ and validates that R' is the same as R that tag generated, by comparing $h(R', C_1, r) = B_3$. If R' is valid, it then computes x_1 and x_2 in the same way as the tag did before. Using Euclidean algorithm reader finds k_1, k_2 such that $k_1 x_1 + k_2 x_2 = 1$. He uses them to compute the hash of tag's id $H(\mathrm{ID}_i) = k_1 B_1 + k_2 B_2$ and identifies the tag.

Unauthorised access to data on the tag is prevented if attacker does not have the reader's private key. From messages $B_1 = x_1 H(\mathrm{ID})$ and $B_2 = x_1 H(\mathrm{ID})$, attacker cannot extract $H(\mathrm{ID})$ without knowing x_1,x_2. He can only get $x1$ and $x2$ if he has the random point R, which can only be extracted from messages C_1 or C_2 (the rest only contain R in a hash, which we assume is not reversible). We ca not do this without knowing the private key s, since $R = C_2 - sC_1$. Since the only known parameter that includes s is $Y = sP$, the only way to get it would require solving the EC discrete logarithm problem.

Additionally, even when the attacker knows the the ID (or $H(\mathrm{ID})$), this does not help him to break the messages – solving x_1 and x_2 from B_1 and B_2 would require solving ECDLP ($x_1 = B_1/H(\mathrm{ID})$, $x_2 = B_2/H(\mathrm{ID})$). This means that even if attacker knows the tag's ID, he cannot identify messages coming from that tag.

Tag forgery is not possible without the ID, because we need its hash to produce valid B_1 and B_2 responses. Replay attack does not work for big enough r (80-bit or more), since the challenge is included in messages C_1 and C_2.

Tracking is prevented, because all encrypted messages appear random – C_1 and C_2 are points multiplied with random number k. B_3 is a hash that contains a random point R, x_1 and x_2 appear random because of the same reason and are used in B_1 and B_2. Assuming that k and R are large enough, attacker should not be able to identify the tags.

The main problem for this protocol is its complexity – it requires 2 different hash functions, two different random number generators and operations on elliptic curves. The authors did not provide any estimates on the number of logic gates needed, but it is probably the most complex protocol covered in this paper. However, if the hardware implementation can be done efficiently enough, this could be a very good protocol for securing RFID devices.

4.4 WIPR

WIPR is a protocol based on Rabin cryptosystem and uses asymmetric keys. It was described in an article by Oren and Feldhofer[4] and some details were changed in a followup article by Wu and Stinson[8] to improve security. The protocol promises to prevent readers without the private key to identify the tag, track the tag based on previous communications or learn anything about the communications, even if public key and tag's ID are known.

Reader is provided with the private key (p, q) where p and q are large prime numbers and a tag is provided the 1024-bit public key $n = p \cdot q$. Tag also has a random number generator and a hash function H(). The protocol:

1. Reader generates a random string of 128 bits c and send it to the tag.

2. Tag generates a random 896-bit string r and a random number r'. It computes the plaintext
$$x = (c \oplus (\mathrm{ID}||o) \oplus H(r)) \;||\; r$$
where ID is a 64-bit identifier, o is a 64-bit padding of zeros, output of the function $H()$ is 128b long and the total length of x is then 1024b.

3. Tag calculates $y = x^2 + r' \cdot n$ and send the result to the reader. This is equivalent to the $y = x^2 \bmod n$ in the Rabin encryption in terms of security, but is easier to implement.

4. Reader calculates x from the equation $x^2 \bmod n = y \bmod n$ using the Chinese remainder theorem, which gives 4 possible solutions. By checking which solution contains the challenge c, reader selects the correct x and parses it to get tag's ID.

The pseudo random numbers on the tag can be generated using a stream cypher such as Grain. Latest versions of Grain are believed to be secure and use about 1300 logic gates.

Calculating the y in step 3. is simplified using a residue number system (RNS) approach, which calculates multiple smaller values that the reader can combine into the solution. This is done to avoid operations on very large integers.

The hash function suitable for this protocol is H-PRESENT-128, which requires around 2330 logic gates.

Including the memory, the whole protocol requires about 7000 logic gates, which is barely within the limits for a low cost RFID tags today.

Unauthorised access to data on the tag is prevented with a variant of Rabin encryption – extracting x from $y = x^2 + r' \cdot n$ is as hard as trying to extract x from $y = x^2 \text{mod } n$, which is a known hard problem (factorisation of large integers).

Impersonating a tag by replaying old messages is not feasible, because the random 128-bit challenge c should always be unique. So impersonating the tag is only possible if we have its ID.

Tracking is prevented by including the random r (and $H(r)$) in the message. That way messages will appear random unless we can decrypt them to reveal r. Since r is large (896b), creating a list of all responses is impossible.

The padding scheme used to create the plaintext x as described above (called WIPR-SAEP) has been shown by Wu and Stinson[8] to be as secure as in Rabin-SAEP variant of Rabin cryptosystem and there are no known attacks to break it.

5 Conclusions

We described the principles behind identification numbers and RFID technology and some of the security concerns that they present. We examined four different protocols that could provide security to RFID devices and looked at how well they work.

The first two algorithms had a few problems related to unauthorised tracking but are much cheaper to implement than the last two. They can be used in situations where we only need confidentiality, but have less need for privacy.

Asymmetric protocols based on discrete logarithm or factorisation problems are becoming more realistic as the price of manufacturing drops. The second algorithm is quite complicated, but if we can find a way to implement it efficiently it seems like a very good solution.

The most promising protocol described seems to be WIPR. An important benefit of the WIPR protocol is that it is based on well known and analysed Rabin

cryptosystem and factorisation problem and is therefore less likely to have weaknesses that have not been found already, whereas a completely new algorithm might contain hidden problems. The price is still an issue today, but manufacturing costs are dropping and it could soon be feasible to implement it on a large scale.

References

[1] Y. Chen, J.-S. Chou, C.-F. Lin, and C.-L. Wu. A novel RFID authentication protocol based on elliptic curve cryptosystem. *IACR Cryptology ePrint Archive*, 2011:381, 2011.

[2] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 357–370. Springer, 2004.

[3] J. Kirtland. *Identification numbers and check digit schemes*. MAA, 2001.

[4] Y. Oren and M. Feldhofer. WIPR–public-key identification on two grains of sand. In *Workshop on RFID security*, pages 15–27, 2008.

[5] T. Tomaš. Identifikacijska Števila. Univerza v Ljubljani, Fakulteta za Matematiko in Fiziko, 2003/04.

[6] D. Trček and P. Jäppinen. RFID security. In *RFID and Sensor Networks*, pages 147–167. Informa UK Limited, nov 2009.

[7] S. A. Weis. Rfid (radio frequency identification): Principles and applications. *System*, 2(3), 2007.

[8] J. Wu and D. R. Stinson. How to improve security and reduce hardware demands of the WIPR RFID protocol. In *2009 IEEE International Conference on RFID*, pages 192–199. IEEE, 2009.