QR-TAN: Secure Mobile Transaction Authentication

Guenther Starnberger, Lorenz Froihofer and Karl M. Goeschka
Vienna University of Technology
Institute of Information Systems
Argentinierstrasse 8/184-1
1040 Vienna, Austria
{*guenther.starnberger, lorenz.froihofer, karl.goeschka*}*@tuwien.ac.at*

Abstract

The security of electronic transactions depends on the security of the user's terminal. An insecure terminal may allow an attacker to create or manipulate transactions. Several techniques have been developed that help to protect transactions performed over insecure terminals. TAN codes, security tokens, and smart cards prevent an attacker who obtained the user's password from signing transactions under the user's identity. However, usually these techniques do not allow a user to assert that the content of a transaction has not been manipulated.

This paper contributes with the QR-TAN authentication technique. QR-TANs are a transaction authentication technique based on two-dimensional barcodes. Compared to other established techniques, QR-TANs show three advantages: First, QR-TANs allow the user to directly validate the content of a transaction within a trusted device. Second, validation is secure even if an attacker manages to gain full control over a user's computer. Finally, QR-TANs in combination with smart cards can also be utilized for offline transactions that do not require any server.

1. Introduction

In traditional signature mechanisms, the user who applies a signature has full control over the signature process. However, in the case of electronic signatures the user depends on a client that often cannot be trusted. Even if a secure smart card is used, the user is often not able to assert that the information displayed on the screen is actually equal to the information signed by the smart card.

This problem is present in all types of electronic transactions that require some type of signature by the user. Examples include online banking and electronic signatures for contracts. The main problem is that information on the client can be arbitrarily modified by malicious software. The article *Secure Internet Banking Authentication* [1] by Thorsten Kramp and Thomas Weigold provides a taxonomy of techniques classified by resilience against offline and online attacks. The only evaluated technique that is robust against content-manipulation attacks is *transaction-signing*. This method requires the user to execute critical operations on a trusted reader device.

In this paper we propose an authentication technique called *QR-TAN* (Quick Response - Transaction Authentication Numbers). QR-TANs use a method based on *transaction-signing* that has been adapted to fit the capabilities of commonly used Web-based applications. QR-TANs are based on two-dimensional QR barcodes. Similar to other approaches [2]–[6], QR-TANs authenticate transactions by using a trusted device. This device can be a mobile phone with a display and a camera with a modest resolution. QR-TANs use QR codes for the transmission of information.

If smart card technology is combined with QR-TANs, transactions can be conducted completely offline without any network connection. QR-TANs do not require any special hardware support on the terminal and could thus easily be used by any kind of Web application. QR-TANs improve on the properties of mobile TANs by not requiring any networking capabilities on the trusted device and by using secure encryption techniques to provide security if an attacker gains access to the trusted device.

2. Problem definition and threat model

The main problem in today's electronic transactions is that the user cannot fully trust the terminal. An attacker might be able to control the terminal and to manipulate any transaction. Therefore, the user cannot assert that the authentication is applied to the same data as the data displayed on the screen. In *Secure Input for Web Applications* [7] the three phases of an attack against a user's computer are described. In the first phase, executable malware is installed on the computer. In the second phase, the malware monitors the user's interaction with Web applications. If the malware detects a security critical operation, it modifies or captures the transmitted information in the final phase.

For most electronic transactions, it is in the interest of both parties that the transactions cannot be forged. Furthermore, it should not be possible for a party to repudiate a transaction. The term *non-repudiation* means that a party must not be able to dispute such a transaction after it has been completed. In order to guarantee *non-repudiation* in online banking applications, the bank must not only ensure the security of its own systems, but also the security on the client side. Therefore, malicious software must not be able to create transactions that have not been approved by the signing party.

We assume that an attacker has full control over the computer and may read and modify any message transferred between the user and a trusted server.

3. State of the art

In the past, different approaches have been suggested and used to address the problem of the untrusted client. Many of them have been developed in the context of online banking applications. This section discusses the state of the art of techniques that are currently used in commercial settings.

TAN codes. TAN codes have traditionally been used by banks to prevent attackers from using a captured password to authenticate transactions. Each user receives a private list of TAN codes and each code may only be used once. An attacker cannot use a captured TAN code if it has already been used before. TAN codes are not able to assure the content of a transaction. A malicious terminal might accept transaction data and a TAN code from a user, but then relay different transaction data to a server. Furthermore, a phishing attack may cause a user to provide her TAN codes to an attacker.

Mobile TANs. Mobile TANs are a relatively new alternative to traditional TANs. When a user needs to authenticate a transaction, the bank sends a summary of the transaction in combination with a TAN code via SMS (Short Message Service). The user then verifies that the summary matches the intended transaction and enters the TAN code into the computer. Hence, the advantages of mobile TANs in comparison to traditional TANs are that the user does not need a TAN list and can use her mobile phone as secure device to approve and validate transactions.

However, mobile TANs also exhibit several disadvantages. First, they result in additional costs for the transmission of messages. Furthermore, there are also problems with the encryption used in mobile networks. The GSM protocol (Global System for Mobile communications) uses the A5/1 and A5/2 ciphers that suffer from well-known vulnerabilities. A5/2 can already be cracked in real-time with a ciphertext only attack [8]. According to a presentation held by the security researchers David Hulton and Steve Muller at the Black Hat 2008 security conference in Washington, D.C., A5/1 can be decoded in about thirty seconds with equipment that consists of 16 128GB flash hard drives and 32 FPGAs (Field Programmable Gate Arrays).

Security tokens. Security tokens are small electronic devices that can be used for two-factor authentication. Typically, they are either directly connected to a computer or a time-dependent number displayed by the device has to be entered into the computer. Security tokens suffer from the same vulnerabilities as TAN codes: The user has no possibility to verify that the data displayed by the computer is actually equal to the data validated with the security token.

Smart cards. Smart cards are credit-card sized devices able to perform cryptographic operations. They are typically used to apply digital signatures or to provide two-factor authentication to a system. A smart card usually does not provide a screen or a keyboard. Therefore, a terminal is required for communication with the user. Unfortunately, smart cards do not generally prevent attacks performed on untrusted terminals, because man-in-the-middle attacks on the terminal may modify any information transferred between the smart card and the user [9].

4. QR-TAN

In order to address the shortcomings of existing solutions, we propose QR-TANs as new transaction authentication technique based on two-dimensional barcodes. QR-TANs allow a user to validate and approve a transaction using an untrusted terminal connected over an untrusted network. We do not place any requirements on the security that must be provided by the terminal. In fact, even if an attacker would be able to fully control the terminal, this would not affect the security of QR-TANs.

This section first gives a short introduction to QR codes. Afterwards, the QR-TAN algorithm used for transaction authentication is described, followed by a discussion of the design decisions and implementation details.

4.1. QR codes

QR codes have been invented by Denso Wave Incorporated (http://www.denso-wave.com/qrcode/) in 1994. The two-dimensional structure of QR codes allows for codes with a lower resolution in any single dimension than a comparable one-dimensional code. Therefore, they can be better recognized by cameras, as the resolution of these cameras is typically the same in both directions. In contrast, one-dimensional barcodes require higher resolution cameras, as more information is stored in a single dimension.

There are different QR code *versions* that mainly differ by the amount of data that can be stored and by the size of the two-dimensional barcode. For example, version 1 contains 21x21 modules and can encode up to 25 alphanumerical

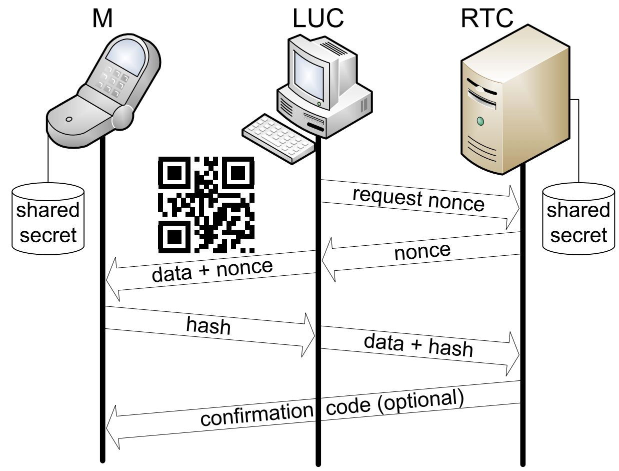

Figure 1. QR-TAN authentication

characters. Version 2 has a size of 25x25 modules and can encode up to 47 alphanumerical characters. The largest amount of data can be stored in QR codes of version 40 with a size of 177x177 modules, they can store up to 4,296 alphanumeric characters (or 2,953 binary 8-bit characters). The practical limit on the size of a QR code is the camera used to capture the code. The main influences are the resolution of the camera and if the camera is able to focus on an object. Figure 1 shows an example of a QR code over the "data+nonce" arrow that contains the string "*QR-TAN!*".

4.2. QR-TAN algorithm

The QR-TAN approach uses a challenge-response mechanism to validate individual transactions. The challenge is transmitted to the phone by the use of two-dimensional barcodes, the response is a few-letter code typed into the computer by the user. A malicious man-in-the-middle is able to learn information about individual transactions, but is neither able to create new transactions nor able to modify existing transactions.

Figure 1 shows the messages that are transmitted during the authentication process. In order to sign a message, the following steps are executed:

1) The user U, who intends to perform a transaction, generates the transaction data T on the local untrusted computer LUC. It is assumed that an attacker is able to read and modify any data stored or relayed by the LUC.
2) The LUC requests a nonce N from the remote trusted computer RTC, e.g., a bank's server. The nonce is required to prevent replay attacks and to prove the freshness of transactions. The RTC is the trusted device in charge of the transaction. In the case of offline transactions a smart card could also perform the tasks of the RTC.
3) The LUC concatenates T and N, encrypts them with the public key of the user's mobile phone M and displays the result as a QR barcode.
4) U uses the mobile phone M to extract T and N and to read T. M acts as a trusted device.
5) If the user wants to approve T, she enters her secret password on M to decrypt the device password D. The device password is a shared secret between the device and the server. It can be initially distributed to the user via letter post as a QR code.
6) M calculates $HMAC_D(T+N+approve+cnt)$ [10], converts the result to an alphanumeric format and displays the first X characters. The $approve$ value serves as an indicator that the user wants to accept the transaction. If the user prefers to explicitely reject a transaction, she can calculate the hash value of $HMAC_D(T + N + reject + cnt)$. The cnt field is usually set to zero. However, if the resulting shortened hash values for *approve* and *reject* match, cnt is increased until the two hash values differ.
7) U reads the first X characters and inputs them on LUC.
8) LUC transmits T and the hash to the RTC.
9) RTC does the calculation for the two possible hashes in step 6 itself and checks if the received hash matches one of the two hashes. Furthermore, the RTC computes the confirmation hash $HMAC_D(T+N+check)$ and transmits it back to the LUC.
10) M computes the confirmation hash using the same formula as the RTC. If the hashes displayed by M and LUC match, the user knows that the transaction has been confirmed by the RTC.

4.3. Design decisions

One of the main design decisions was that manual work required by the user and computational requirements on the trusted device should be kept to a minimum. Unlike other approaches that require the user to perform manual calculations or approaches that require a separate channel, e.g., via Bluetooth, the QR-TAN approach only requires the user to validate the transaction on her trusted device and to approve the transaction by entering a short number into her computer. The effort required by the user to learn the QR-TAN mechanism is therefore comparable to other TAN based approaches.

In order to design a technique that is also applicable to commercial applications, we placed several requirements on the resulting QR-TAN algorithm:

- QR-TANs must not require any manual computations or other complex tasks by the user.
- QR-TANs must provide the same (or a better) security than other existing established approaches.

- It should be possible to use QR-TANs offline by embedding the server side tasks within a smart card (only standard algorithms like AES/SHA-1 should be used).
- Implementation and usage should be possible with only modest expenses. Operation should be cheaper than the often used mobile TAN (mTAN) technique.

When comparing the transport mechanism, there is an important asymmetry. While the path to the mobile phone could easily convey several tens of bytes to several hundreds of bytes, the path from the phone to the computer can only transport a few bytes. The reason is that the user manually needs to enter the information displayed on the phone onto the keyboard. Therefore, this restricts how the QR-TAN technique is designed and which cryptographic techniques can be used. For example, if the output is encrypted with public key cryptography or a block cipher like AES, there is a minimum length for the cipher text. In most situations, however, expecting that a user manually types information of this minimum length into the computer is not an acceptable solution.

4.4. Discussion

This section discusses details affecting the security of QR-TANs as well as selected implementation details.

Encryption. Most of the data are not signed or encrypted as these security measures would not provide any additional security. The nonce can only be used by M. An attacker who knows the nonce cannot generate valid hashes or replay any transactions. If an attacker manipulates the nonce, this would cause M to generate an invalid hash. The same is true for the transmitted hashes as knowledge of these hashes does not provide any viable information to an attacker.

A nonce is required to prove the freshness of the data and to prevent replay attacks. If it is not possible to obtain a nonce from the RTC, the LUC could instead include a timestamp in the data. The user then verifies the timestamp and only signs the data if the timestamp is correct. When the data are transmitted to the RTC, the LUC also includes the timestamp that is part of the hash. The RTC can then check if the timestamp agrees with the hash and if the timestamp is near the current time. By rejecting identical transaction data with a timestamp that is identical to another timestamp, during the validity period of both timestamps, the RTC can prevent replay attacks.

The purpose of the public key algorithm used to encrypt messages to the mobile phone is to prevent some types of attacks. For example, an attacker who is physically located near the user, but who is not able to decipher the individual characters on the screen might still be able to decode the displayed QR barcode.

For the symmetric encryption of the data, any encryption scheme could be used. For the asymmetric encryption, a public key algorithm that yields a small ciphertext would be advantageous. Therefore, if possible Elliptic Curve Cryptography (ECC) should be used instead of RSA. When compared to RSA, the ciphertexts generated by ECC are smaller for a given plaintext and for a given level of security.

The password used to decrypt the shared secret for the message signatures must not be related to the decryption of the QR codes read by the mobile device, e.g., by securing the private key with this password. Otherwise, if an attacker would be able to obtain the device, to extract the private key, and obtain a picture of a QR code destined for the device, the attacker could brute-force attack the user's password and verify the result by checking if the decrypted QR code contains reasonable data. This password could then also be used by the attacker to sign transactions. If the user's password is only used for signatures, the attacker has no prior indication if the password is correct. Therefore, the RTC could block the account if there are more than a given number of failed attempts.

Resilience against attacks. For the generation of the hash displayed on the mobile phone and transmitted by the user to the LUC, the binary data needs to be recoded to a format that can easily be handled by the user. As a trivial solution, this data can be converted to a format that contains only case-sensitive alphanumerical characters. This would yield 62 different possibilities per position. If two of these possibilities are removed in order to prevent user's from mistaking *l* with *1* (one) or *O* with *0* (zero), this leaves 60 possibilities per position. With a hash length of four characters the total number of combinations is $12,960,000$. The chance of guessing the correct code is therefore roughly comparable to the chance of winning the jackpot in a 6 from 49 lottery. A hash length of six characters would already yield $46,656,000,000$ different combinations. As the number of accepted trials by the RTC can be set to a low value, this effectively prevents brute force attacks that could be used to guess the correct hash.

Hash options. The decision if a user wants to *approve* or *reject* a transaction is encoded in the hash as this prevents the LUC from learning the information. Depending on the application, additional values may be used. If it is allowable that the LUC is able to learn the choice of the user, a user could also omit the input of any hash value. Consequently, this would lead to a timeout for the nonce at the server.

The confirmation hash generated with the *check* option allows the user to verify that a transaction has been received by the RTC. It can be displayed as a shortened hash or by using hash visualization techniques [11]. The confirmation hash helps the user to discover that the LUC does not forward transactions accordingly. However, if the user does not receive any confirmation hash by the LUC she cannot detect if the message containing the transaction, or the message containing the confirmation was lost. This is a more general problem related to the Two Army Problem [12] that cannot be fully solved.

5. Attacks and Security

In *Two-Factor Authentication: Too Little, Too Late* [13] Bruce Schneier argues that even modern two-factor authentication techniques will not solve today's phishing problems as they do not solve man-in-the-middle attacks and trojan attacks. This reasoning is valid if the user has no possibility to assert that the content of a transaction is correct. However, in the case of QR-TANs a user directly validates each transaction on a secure device. Therefore, the user can check if the transaction displayed on the secure device matches the transaction that has been input in the computer.

An attacker is only able to calculate the hash code, if she is able to obtain the key shared between the user's phone and the RTC. However, the key is protected by the user's personal password and optionally also stored within a secure element (e.g., on a smart card chip).

To successfully issue a malicious transaction, an attacker would need to (i) know the password required to login to the bank's website, (ii) steal the mobile phone of the user and bypass access controls (e.g., the password of a screensaver), and (iii) know the personal password of the user required to approve transactions on the phone.

If the mobile phone is considered to be insecure, for example because an attacker might be able to execute software on the phone, the attacker would still need to gain control over the phone *and* over the terminal. If the cryptographic algorithm as well as the required user credentials are stored within the phone's smart card, access to the phone is required at the time of the transaction.

For high security applications it might therefore make sense to use a device similar to security tokens for the authentication of transactions. This devices could feature a simple camera used to scan QR codes and a display to show TAN codes. This device does not need any connection to the Internet or other types of networks. Therefore, the attacker would require physical access to the device in order to install malicious software.

6. Related work

A number of other papers propose methods that can be used to implement secure transaction authentication on untrusted terminals. There are two different types of approaches: Some of these methods use external devices that are used to validate transactions. They therefore shift the security burden from an untrusted device to an external trusted device. Other approaches require the user to manually validate that a transaction is correct. This section describes related work that solves similar problems as QR-TANs.

The Untrusted Computer Problem and Camera-Based Authentication [5]. This approach by Clarke et al. describes how a trusted mobile device can be used to validate transactions that are conducted over an untrusted device. The threat model and the assumptions on the environment are similar to our assumptions made for QR-TANs. In this approach the mobile device acts as a monitoring device that can detect if the information displayed on the untrusted computer's screen is correct. If the information has been tampered with by an attacker, the monitoring device warns the user about this fact.

The main difference to QR-TANs is the way how the information is transferred from the untrusted computer to the mobile device. The paper presents two different options to read data from a screen. *Pixel mapping* establishes a mapping between the camera's pixels and the screen's pixels. It requires a calibration phase and the camera must not be moved relative to the screen. The second option uses *Optical Character Recognition* (OCR) and does not require calibration. It also allows the camera to be moved. However, validation takes longer and valid output is rejected if there is a slight distortion in the image.

Compared to QR-TANs, this approach has higher requirements on the resolution of the camera and on the computational power of the device. As a consequence, an implementation may not be feasible on today's mobile phones.

Using Camera Phones for Human-Verifiable Authentication [3]. This paper by J. M. McCune et al. describes how to use the visual channel provided by two-dimensional barcodes for authentication and identification of devices. In comparison to QR-TANs, this approach uses visual information only as a restricted communication channel, for example to exchange cryptographic keys. The actual communication is done over another channel.

Using a Personal Device to Strengthen Password Authentication from an Untrusted Computer [4]. In this paper by Mannan and Oorschot, the MP-Auth authentication mechanism is introduced. MP-Auth allows the user to securely login to a website and to approve transactions using a cell phone. The user's long-term secret password is entered on the trusted mobile device, while the untrusted device only gains access to a temporary secret.

Unlike QR-TANs, the MP-Auth mechanism requires bidirectional data transfer between a cell phone and a terminal, such as Bluetooth or wired connections. Furthermore, software installation on the untrusted terminal is required.

7. Future outlook and conclusion

An advantage of QR-TANs over existing authentication techniques is that they do not require any additional software installation on the terminal. While a software implementation of the QR-TAN authentication technique is required on the trusted device, this implementation does not need to be tailored to any particular service provider. Therefore, such software might be pre-installed on mobile phones or dedicated secure devices.

As we expect the following technology changes in the future, it is likely that QR-TANs could be used for a wide range of different applications.

- New cameras with higher resolutions will be available in mobile phones.
- Processing power of mobile phones will increase. Therefore, more complex barcode algorithms can be used.
- Current QR codes are black/white only. More advanced barcodes that also use different colors (e.g., Microsoft's *High Capacity Color Barcode*) are able to store more information.

In summary, QR-TANs are able to substantially increase the security of electronic transactions. A user can use her own mobile phone to approve transactions that have been created on untrusted terminals. QR-TANs use a visual communication channel and do not require any direct communication link between the mobile trusted device and the untrusted terminal. Therefore, they do not require the installation or configuration of additional software on the untrusted terminal. This is a main advantage over other techniques that use secure devices, as these techniques often require a bidirectional link between the trusted and the untrusted device.

When compared to the established mobile TAN authentication technique, QR-TANs allow for less costs at the service provider while at the same time providing a higher level of security. Unlike other proposed techniques, QR-TANs only require modest communication and computation capabilities at the trusted device. Therefore, we are convinced that the usage of QR-TANs in security critical environments like Internet banking is reasonable and that QR-TANs provide a viable alternative to today's established authentication techniques.

Acknowledgments

This work has been partially funded by the Austrian Federal Ministry of Transport, Innovation and Technology under the FIT-IT project TRADE (Trustworthy Adaptive Quality Balancing through Temporal Decoupling, contract 816143, http://www.dedisys.org/trade/).

References

[1] A. Hiltgen, T. Kramp, and T. Weigold, "Secure internet banking authentication," *IEEE Security and Privacy*, vol. 4, no. 2, pp. 21–29, 2006.

[2] D. Balfanz and E. W. Felten, "Hand-held computers can be better smart cards," in *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 1999, pp. 2–2.

[3] J. M. McCune, A. Perrig, and M. K. Reiter, "Seeing-is-believing: Using camera phones for human-verifiable authentication," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2005, pp. 110–124.

[4] M. Mannan and P. C. van Oorschot, "Using a personal device to strengthen password authentication from an untrusted computer," in *Financial Cryptography*, ser. Lecture Notes in Computer Science, S. Dietrich and R. Dhamija, Eds., vol. 4886. Springer, 2007, pp. 88–103.

[5] D. E. Clarke, B. Gassend, T. Kotwal, M. Burnside, M. van Dijk, S. Devadas, and R. L. Rivest, "The untrusted computer problem and camera-based authentication," in *Pervasive*, ser. Lecture Notes in Computer Science, F. Mattern and M. Naghshineh, Eds., vol. 2414. Springer, 2002, pp. 114–124.

[6] A. Oprea, D. Balfanz, G. Durfee, and D. K. Smetters, "Securing a remote terminal application with a mobile trusted device," in *ACSAC*. IEEE Computer Society, 2004, pp. 438–447.

[7] M. Szydlowski, C. Kruegel, and E. Kirda, "Secure input for web applications," in *ACSAC*. IEEE Computer Society, 2007, pp. 375–384.

[8] E. Barkan, E. Biham, and N. Keller, "Instant ciphertext-only cryptanalysis of GSM encrypted communication," in *CRYPTO*, ser. Lecture Notes in Computer Science, D. Boneh, Ed., vol. 2729. Springer, 2003, pp. 600–616.

[9] B. Schneier and A. Shostack, "Breaking up is hard to do: modeling security threats for smart cards," in *WOST'99: Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*. Berkeley, CA, USA: USENIX Association, 1999, pp. 19–19.

[10] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104 (Informational), Feb. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2104.txt

[11] A. Perrig and D. Song, "Hash visualization: a new technique to improve real-world security," in *In International Workshop on Cryptographic Techniques and E-Commerce*, 1999, pp. 131–138.

[12] E. A. Akkoyunlu, K. Ekanandham, and R. V. Huber, "Some constraints and tradeoffs in the design of network communications," in *SOSP*, 1975, pp. 67–74.

[13] B. Schneier, "Two-factor authentication: too little, too late," *Commun. ACM*, vol. 48, no. 4, p. 136, 2005.