Involving Humans in the Cryptographic Loop: Introduction and Threat Analysis of EEVEHAC

Julius Hekkala[1], Sara Nikula[1], Outi-Marja Latvala[1] and Kimmo Halunen[2,3]
[1]*VTT Technical Research Centre of Finland, Kaitoväylä 1, Oulu, Finland*
[2]*University of Oulu, Faculty of Information Technology and Electrical Engineering, Oulu, Finland*
[3]*National Defence University, Department of Military Technology, Helsinki, Finland*

Keywords: Human Understandable Cryptography, Visualizable Cryptography, Narrative Authentication, Secure Channel.

Abstract: Our digital lives rely on modern cryptography that is based on complicated mathematics average human users cannot follow. Previous attempts at adding the human user into the cryptographic loop include things like Human Authenticated Key Exchange and visualizable cryptography. This paper presents our proof-of-concept implementation of these ideas as a system called EEVEHAC. It utilizes human capabilities to achieve an end-to-end encrypted channel between a user and a server that is authenticated with human senses and can be used through untrusted environments. The security of this complete system is analyzed. We find that the combination of the two different systems into EEVEHAC on a theoretical level retains the security of the individual systems. We also identify the weaknesses of this implementation and discuss options for overcoming them.

1 INTRODUCTION

Modern cryptographic systems protect users and services from many threats such as loss of confidential information or unauthorized access. However, for the human end users these cryptographic protections are opaque and it is very hard for them to gain information on whether the cryptographic protocol has been correctly executed in human understandable terms.

There are many methods that could provide some human understandable elements to cryptography, e.g., visual cryptography (Naor and Shamir, 1994). But, there is more to trusting cryptography than visually decrypting some outcomes. For example, key generation and exchange are important procedures that have great impact on the security of a system, and human-compatible functions (Boldyreva et al., 2017) provide first solutions to this problem. An interested reader can read a recent review by Halunen and Latvala (2021) for more examples of cryptography and human abilities.

Despite the promising developments on individual components that try to make cryptography more human friendly, there has not been a complete system that combines all the needed aspects for an encrypted communications channel. In this paper we introduce and provide a threat analysis of EEVEHAC (End-to-End Visualizable Encrypted and Human Authenticated Channel), a system that combines Human Authenticated Key Exchange (HAKE) (Boldyreva et al., 2017) and a visualizable encryption scheme (Forte et al., 2014). Even though these concepts have been studied in former papers, no papers discussing them as an integrated system have been published.

The paper is organized as follows. Section 2 contains an overview of the whole system, and a threat model for the whole system is presented. In section 3, the HAKE utilising stories and colors and its threat analysis is presented. Section 4 covers the visualizably encrypted channel, threats related to it and its security analysis. Section 5 includes security analysis of the whole EEVEHAC system. Finally, discussion and conclusion sections close our paper.

2 OVERVIEW OF EEVEHAC

EEVEHAC utilizes HAKE by Boldyreva et al. (2017) in establishing keys between a server and a mobile device. The keys are used in visualizable encryption, which is utilized in establishing a secure channel between the human user and the server, via an untrusted terminal. This channel can be used for, e.g., performing a simple PIN authentication. Figure 1 presents a schematic overview of the EEVEHAC system and Figures 2 and 4 showcase our implementation.

Hekkala, J., Nikula, S., Latvala, O. and Halunen, K.
Involving Humans in the Cryptographic Loop: Introduction and Threat Analysis of EEVEHAC.
DOI: 10.5220/0010517806590664
In *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)*, pages 659-664
ISBN: 978-989-758-524-1

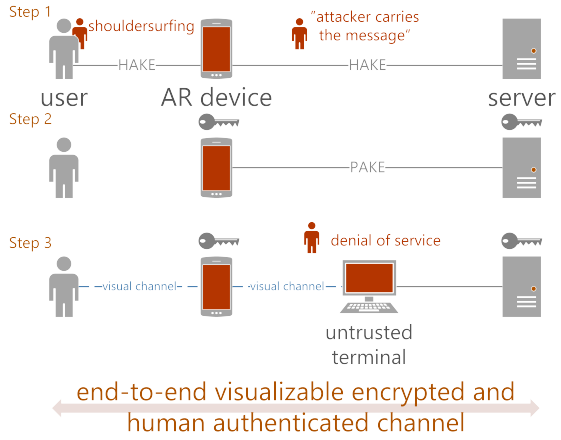


Figure 1: Overview of the EEVEHAC system. Steps 1 and 2 cover the first part (HAKE) of the system. The third step covers the second part of the system, i.e. the visualizable encryption scheme. An attacker may be physically present or online in the connection to the server.

In Figure 1 we have depicted a trusted user's device, that has a camera and a screen for augmented reality (AR) purposes, a server, and an untrusted terminal in a public space. A story database and the keys for the visualizable encryption scheme are stored on the user's device and on the server. There are three actors: the user, the server and an attacker.

The objective of the user is to log into the service, the server's is to allow legitimate users to log in. An attacker's objective is to steal the user's credentials. There are two types of keys the attacker could obtain: the long term keys generated during the HAKE phase and the short term session keys used with the visual channel. The long term keys are more valuable. Threats towards the server are ruled out of scope.

We assume that the user's device is trusted: there is no malware and no one else can access it. The public device is untrusted: it may have malware, anyone can access it and actions on it are visible to onlookers. The server is neutral: there are no protocol violations. The story database needed for the first step is public, but the color mapping for each user is secret. Communication between the public device and the server is unsecured and there is only a visual channel between the public device and the user's device.

The threat model used in this paper is based on the well-known Dolev-Yao model (Dolev and Yao, 1983). This model has several parties using the same communication protocol: All messages are encrypted, and only the legitimate receiver possesses the decryption key. Anyone can read, copy and resend messages and anyone can write and send forged messages. The attacker is a legitimate user of the communication network trying to obtain the original plaintexts.

Figure 2: Inserting the UAN code in our implementation.

In our scenario, an attacker on the Internet is able to see and manipulate messages between the user's device and the server, as well as the untrusted terminal and the server. This is depicted as the "attacker carries the message" -figurine in Figure 1. An attacker present on the use location may be able to see critical information on the personal device, or be able to disrupt the visual channel by, e.g., standing in the way ("shouldersurfing" and "denial of service").

3 HAKE FOR EEVEHAC

Boldyreva et al. (2017) present the *Human Authenticated Key Exchange* protocol, a new type of interaction between a human user and a server. The user receives a challenge from a server, computes the answer in their head and responds. The messages sent need to be *human readable* or *human writable*, and the mathematical operations *human computable*.

The value of including this protocol in our system is that it involves the human user in the authentication phase at a new level. In contrast to traditional cryptographic protocols, the human user can compute the answer in their head and thus it should be clear to them why the authentication protocol succeeded, increasing trust towards the system and further interactions between the user and the server.

3.1 Our Implementation

HAKE in EEVEHAC is the Basic Generic HAKE protocol by Boldyreva et al. (2017). For the Password Authenticated Key Exchange (PAKE) part of the protocol we used the Encrypted Key Exchange (Bellovin and Merritt, 1992). The result of the human understandable function is used in PAKE to encrypt the first

messages sent between the client and the server.

Our HAKE schema is based on a story and a mapping between colors and numbers. First, the user and the server establish a long time secret as the user registers to the service. In this phase, the user gets a story and an identifying user number. Basis of the story is automatically generated by the server, but the user can modify it, balancing the strengths and weaknesses of machine and human generated stories. Machine generated stories are random and hard to guess, whereas human generated stories are easier to remember (Somayaji et al., 2013).

The user also gets a color-to-number mapping, which includes six numbers corresponding to six colors in our implementation. The colors are the same for every user, but the numbers vary between users. The user needs to memorize this secret mapping in order to use it in every authentication session in the future.

In the login phase, the user sees colored, grammatically correct sentences on the screen of their device (Figure 2). Some of the sentences come from the original story, but one word has been changed in them. The user detects the changed words and notes their colors. Next they recall the corresponding numbers and count modulo 10 of the sum of these numbers. The result is entered in the Unique Authentication Number (UAN) sequence. These steps are iterated as many times as is the length of the UAN.

3.2 Threat Model of HAKE

The goal is that the user can establish a secure channel with the server based on their knowledge of the long time secret: the user and the server exchange keys for subsequent phases, and nobody else has knowledge of these keys. The long time secret should not be revealed even if an attacker sees some challenges and responses in a key exchange session.

The threat model assumes that there is a human computable function, and a channel between the user and the server is established by applying this function to the challenges and answering them correctly. Our HAKE protocol consists of sum and modulo operations based on the user's story and color-to-number-mapping. A secure HAKE protocol should not reveal the long term secret even if a limited number of challenge-response-pairs were leaked. Furthermore, the attacker should not be able to build a connection with the server by impersonating the user, or vice versa, without knowledge of the long time secret.

In the registration phase, there is a risk of a man-in-the-middle attack. Since the registration is conducted on a computer with Internet connection (e.g. TLS) to the server, threats concerning it are the same as with any communication over the Internet.

If an attacker can obtain the long term key via breaking the HAKE protocol, the visual channel between the user and the server would no more be secure. Our threat model related to this case has elements similar to presented by Boldyreva et al. (2017). We consider two types of attacks: a brute force attack, where an attacker tries to complete the protocol by guessing, and a man-in-the-middle attack, where an attacker is eavesdropping on a real authentication session. In the latter case, the attacker can try to figure out the long time secret by numerically analyzing challenges and responses. In this case, an attacker can also modify the messages sent by the server, to try to derive even more information about the long time secret.

3.2.1 Security Analysis of HAKE

Currently, whoever possesses the user number can ask for challenges from the server. Because the server only modifies the clauses in the user's story by changing one word at a time, after seeing 2 or 3 authentication sessions it becomes easy to infer the original story. Thus, the security of the UAN protocol is based solely on the color-to-number mapping. Because this protocol is executed only rarely, we expect that the user can make sure the environment is safe (e.g., by doing it at home). However, we will go through a scenario where a shoulder surfer is able to see some challenge-response -pairs in a UAN session, and analyze their possibilities to break the protocol. Moreover, an active attacker can be more efficient than a passive one, since they can pick optimal challenges to get the most information out of the challenge-response-pairs.

Every UAN code consists of four digits. Thus the probability that an adversary can, if only given one pure guess, correctly guess the UAN code is $1/10000$. Considering the color-to-number mapping, there are in total 6^{10} possible mappings between six colors and ten numbers. Thus, the probability that an adversary can correctly guess the mapping without any additional information is $1/(6^{10})$.

An attacker listening to messages sent in one or more UAN sessions can derive information of the long time secret either by constructing a list of correct answers corresponding to a specific color combination, or by trying to figure out the secret color-to-number-mapping. With n different colors, it is possible to make $n(n-1)/2+n$ combinations; $n(n-1)/2$ combinations of two different colors and all n colors combined with themselves. In our implementation, there are six colors and 21 possible color combina-

tions. After seeing challenges and responses in one UAN session, an attacker would have seen approximately 4 different color combinations, and 17 remain unknown. To successfully impersonate the user, an attacker would need information on all the asked color combinations in the four challenges sent by the server. The probability that all four challenges in the next UAN session only include color combinations already seen by an attacker is $(x/(n*(n-1)/2+n))^4$, where x is the amount of different color combinations seen by an attacker and n is the amount of colors.

Another way to break the UAN protocol is to derive information about the color-to-number-mapping by comparing challenges and responses. Because the user is counting modulo 10 of the sums, every digit entered in the UAN code can be derived from two different numbers, e.g. $3 \bmod 10 = 13 \bmod 10 = 3$. For every such number pair there are five combinations of two different numbers which result to this number when summed together: e.g., $5+8=13$ and $3+0=3$. In order to make use of this information in subsequent authentication sessions, the adversary must not only know which two numbers were added together but also which color corresponded to which number. In this case every pair of numbers can be derived from two different mappings, e.g., red = 5 and green = 8 or vice versa. Because there were five possible number combinations yielding the outputted digit and two possible mappings per one combination, one answer to a challenge including two different colors can be derived from ten different color-to-number mappings. This is illustrated in Figure 3a. The case with one color seen twice is illustrated in Figure 3b.

4 VISUALIZABLE CHANNEL

The second part of EEVEHAC is an end-to-end visualizably encrypted channel. For this purpose, EEVEHAC implements EyeDecrypt (Forte et al., 2014). The original model consists of a user's personal device with the app installed, a server and an adversary. Data moving in the visualizably encrypted channel is divided into frames and blocks. A frame represents the contents shown on the screen of the untrusted device at one point of time and is divided into multiple blocks. In our implementation a frame contains six blocks (QR codes in 3x2-grid).

According to the model, an adversary can manipulate the untrusted device and also shoulder-surf. Active adversaries can manipulate the communication between the server and the untrusted device and thus deploy e.g. man-in-the-middle attacks, while passive adversaries observe the user's inputs and the screen of the untrusted device. Forte et al. (2014) define the security of EyeDecrypt in terms of how much information can leak from the system to the adversary: the visual representation of the current frame, any user input or information gained from active tampering.

Forte et al. (2014) also formally define visualizable encryption and its security proof. Because EEVEHAC includes the whole EyeDecrypt scheme, we assume the security proofs presented in the paper hold for the visualizably encrypted end-to-end channel part of EEVEHAC. EyeDecrypt requires a visual encoding scheme that fulfills certain requirements: the visual encoder needs to encode a single unit of code in a single block, the system must be able to decrypt multiple blocks at once and know and interpret the spatial arrangement of the blocks.

4.1 Our Implementation

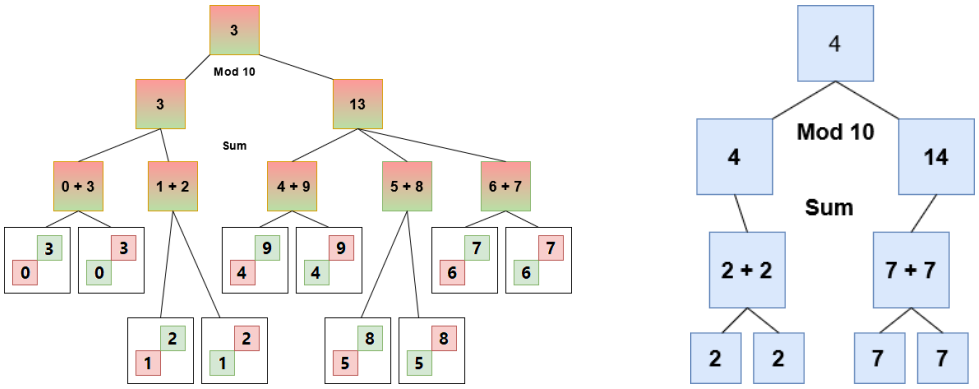
In our proof-of-concept implementation, the untrusted device and the trusted server run on the same laptop. For the user's trusted device, we used a smartphone. After completing the HAKE phase the server and the user's smartphone have matching AES (Daemen and Rijmen, 2002) and HMAC (Krawczyk et al., 1997) keys that are used to encrypt and decrypt messages as well as to authenticate the server's messages.

The server sends the encrypted messages to the untrusted device, which shows them in their respective positions in the UI grid as QR codes. The user scans them with the smartphone camera. When the application recognizes all 6 QR codes simultaneously in a single frame of the camera feed, it starts processing the QR codes. The application checks the positions of the QR codes: incorrect positions are indicated to the user with red frames on the QR codes in the camera view. Correct positions are indicated with green frames, see Figure 4. If the messages are uncorrupted, the application can decrypt them correctly and show the plaintext to the user in the camera view on top of the QR codes. If the application cannot decrypt the ciphertexts to understandable messages, the user can see that something has gone wrong.

5 SECURITY ANALYSIS

This section contains security analysis of the whole EEVEHAC system. We consider both active and passive attacks. Active attacks include things like copying or forging messages, while a passive attacker only observes the user and service provider.

An active attacker could try to impersonate the user or the server. If they manage to break the HAKE



(a) Using two different colors (b) The same color used twice

Figure 3: Example cases of deriving numbers for UAN.

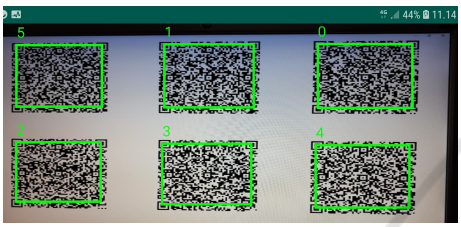


Figure 4: Scanning QR codes in our implementation.

protocol they would gain access to a new long term key that is used in encrypting the messages in the second phase. The acquired key would be different than the long term key previously used between the user and the server. A resulting mismatch between the server and user keys means that the second phase is unsuccessful, alerting the user to something being wrong and re-doing the HAKE phase. Moreover, messages sent before the impersonation cannot be decrypted with short term session keys derived from the new long term key. If the impersonation attack is successful simultaneously in both directions and the HAKE protocol is broken, all three parties would have the same long term key, and EEVEHAC is defeated.

If the attacker acquires an established long term key on the user's device, they could examine all messaging between the user and the server without alerting either party and the EEVEHAC system would not work. This could be done by planting malware on the user's device. However, this contradicts our original assumption that the user's device is trusted.

What happens when the long term keys are not compromised? Now, the attack focuses on the latter phase of EEVEHAC, which uses short term session keys and is used more often than the HAKE phase discussed above.

An attacker is assumed to be able to attach themselves between the server and untrusted device and read and edit the messages. Not sending the messages forward or replacing them with random data works as a denial of service. Unless the attacker has access to the most current AES and HMAC keys, they are not able to produce coherent messages that would fool the application in the user's smartphone.

When the untrusted terminal is used in a public space, we assume that a shoulder surfer can see all the blocks in the frames and what inputs the user makes. The decrypted data, however, is only visible on the user's device. If the attacker wanted to see it they would need to be very close to the user, risking detection. If EEVEHAC was run on a futuristic AR device like smart glasses, risk of shoulder surfing would be minimal.

6 DISCUSSION

The security analysis of the HAKE part of EEVEHAC revealed that the security level of the current implementation is poor. There are at least two options for improving it: adding more colors or making the user-specific story secret.

Increasing the number of colors from 6 to 10 would increase the number of possible color combinations from 21 to 55. However, this might not be user friendly: it would be more difficult to learn the mapping, and there is a limited number of clearly distinguishable colors available. The color choices need to take into account visual impairments and the fact that different devices display color differently. The calculations involved in the HAKE phase might also disadvantage some people.

Protecting the user-specific story would increase the computational difficulty of breaking the UAN protocol: an eavesdropper seeing clauses on the screen would no longer know which colors are added together, because they would not know which words

are wrong. In the current implementation, one challenge in the UAN protocol includes four clauses on the screen, one clause includes five words, and there are six colors in use. Thus, an attacker would need to consider several possible color combinations at every response. They would be able to break the color-to-number mapping only by seeing several successful UAN protocols, which is highly unlikely, because the protocol is conducted rarely and it is assumed that the surroundings are safe.

Using a personal word database might make it easier for a human user to remember the story, if they were allowed to add some new words of their choice to be used in the story upon registering to the service. On the other hand, that could also be a risk: many users would probably pick personal or otherwise deviating words, which would be easy to spot when shoulder surfing.

Using more advanced techniques in story generation phase might be advantageous. In the current implementation, even though the clauses are grammatically correct, the result is not very coherent or meaningful. By utilizing technology such as GPT-3 (Brown et al., 2020), which is capable of producing texts resembling human-generated ones, it would be possible to make the story memorable and still maintain its randomness.

For the visual channel, we would recommend using a visual encoding scheme different from QR codes in a real use scenario. QR codes cannot be placed as densely as other visual encoding schemes, as they require ”free zones” around them. With some other visual encoding scheme, the GUI could achieve a cleaner look as well.

Performance of the application is also important. The user needs to be able to view a whole frame at once, and the application needs to be able to decipher the picture. With bigger QR codes the application recognizes the elements more easily, improving the performance. When developing an application like this for real users, usage optimization and achieving high enough framerate are extremely important. No one will use the application if the camera view lags visibly or if the application has great difficulties scanning the frame. Finally, if the system was implemented on smart glasses, the GUI and application performance would need their own consideration.

Next step for our work on EEVEHAC is usability testing. The usability of the first phase of HAKE can be compared to other results on the usability of human computable functions. The second phase has few direct comparisons, but its usability can be analyzed by how much physical effort it takes to use the devices.

7 CONCLUSIONS

In this paper, we presented the EEVEHAC (End-to-End Visualizable Encrypted and Human Authenticated Channel) system. The main purpose of EEVEHAC is to involve the human user in the cryptographic process in order to enhance understanding and trust in the digital systems that are a crucial part of our everyday lives. Our security analysis indicates that EEVEHAC can achieve as high a security level as its individual parts. Our proof-of-concept implementation is not secure enough for real world usage but implementations of EEVEHAC can easily be modified to achieve higher security.

REFERENCES

Bellovin, S. M. and Merritt, M. (1992). Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, SP ’92, page 72, USA. IEEE Computer Society.

Boldyreva, A., Chen, S., Dupont, P.-A., and Pointcheval, D. (2017). Human computing for handling strong corruptions in authenticated key exchange. In *Computer Security Foundations Symposium (CSF), 2017 IEEE 30th*, pages 159–175. IEEE.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

Daemen, J. and Rijmen, V. (2002). *The design of Rijndael*, volume 2. Springer.

Dolev, D. and Yao, A. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208.

Forte, A. G., Garay, J. A., Jim, T., and Vahlis, Y. (2014). EyeDecrypt—private interactions in plain sight. In *International Conference on Security and Cryptography for Networks*, pages 255–276. Springer.

Halunen, K. and Latvala, O.-M. (2021). Review of the use of human senses and capabilities in cryptography. *Computer Science Review*, 39:100340.

Krawczyk, H., Bellare, M., and Canetti, R. (1997). HMAC: Keyed-hashing for message authentication.

Naor, M. and Shamir, A. (1994). Visual cryptography. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 1–12. Springer.

Somayaji, A., Mould, D., and Brown, C. (2013). Towards narrative authentication: Or, against boring authentication. In *Proceedings of the 2013 New Security Paradigms Workshop*, pages 57–64.