Secure Communication Protocol for a Low-Bandwidth Audio Channel

Waldemar Berchtold, Patrick Lieb, Martin Steinebach
Fraunhofer Institute for Secure Information Technology
Rheinstraße 75, 64295 Darmstadt, Germany
Email: {berchtold, lieb, steinebach}@sit.fraunhofer.de

***Abstract*—Data transmission over an inaudible audio channel describes a low-bandwidth alternative to exchange data between devices without any additional infrastructure. However, the established communication channel can be eavesdropped and manipulated by an attacker. To prevent this, we introduce a tailored protocol with smallest possible overhead to secure the communication. The proposed protocol produces an overhead of 256 bits for the handshake message for setting up the first conversation with each partner. Further, the protocol produces $\lceil msg\_len/64 \rceil * 3 + 67$ bits overhead for each message. The overhead of 67 bits at the beginning of each message corresponds to one second transmission time with the used FSK modulation in the frequency range of 16kHz-20kHz. The additional overhead of 3-bit per 64-bit sequence poses a relation of 95% message to 5% overhead. For the implementation of the protocol, algorithms implemented in the Crypto++ library such as SHA-256, CCM and PBKDF2 have been used.**

I. Introduction

The communication over an inaudible high-frequency audio channel is a convenient way to exchange data since only microphones and speakers are required to establish an audio communication channel. In general, location-based audio communication applications do not enforce any security mechanisms. Due to the absence of a secure audio communication protocol, applications requiring security usually only send a data identifier over the audio channel. This identifier is used to establish a secure out-of-band channel to exchange all further data. However, since the data is commonly fairly small, it is reasonable to directly exchange the data over the audio channel instead. In order to assure the security of the communication over the low-bandwidth audio channel, a tailored protocol is required [1]. The majority of the mobile devices can usually only process frequencies up to 20kHz. However, the sensitivity of the human ear is very low in the frequency range of 16kHz-20kHz [2]. That is why the communication protocol operates on this frequency range.There have already been introduced some high-frequency audio communication applications and protocols such as Way2ride [3], Aerolink [4], shopKick [5], SmartGuide [6], Signal360 [7], Infosound [8], SilverPush [9], SlickLogin [10] and Chirp [11]. However, some of these techniques do not state if or how they use cryptographic mechanisms in order to secure their communication or they do not use security mechanisms at all. Only the inventors of Chirp describe that their technique allows a secure communication [11] but they do not explain how security is enforced. Since only [6], which does not use any cryptographic techniques, describes an open standard and all other protocols are proprietary, it is hard to verify wether their communication is actually secure.

Audio channels can also be used for secure pairing of devices. The authors of [12]–[15] use audible sounds when setting up an audio communication channel. However, they assume that the audio channel can only be eavesdropped but not be manipulated. Based on their assumptions, the audio channel would always be authenticated. In fact, this is not generally the case. Any possible attacker could use a directional microphone to eavesdrop the audio communication and a directional audio source to manipulate the channel. That is why this work introduces a data communication protocol that enables to securely exchange data over an audio channel. The proposed protocol uses open standard and open source cryptographic algorithms.

II. Analysis

The requirements of the resulting protocol are exactly defined: In order to establish a secure communication, the two communicating parties can exchange passwords before the start of the communication. In addition, both parties have the capability to show a six-digit alphanumeric value. At least one party has both display capabilities for a six-digit number and input capabilities. However, the computing power of both parties is also limited. The protocol is supposed to be able to run on resource-constrained devices including but not limited to embedded systems.

Currently, properties of the underlying modulation enable to transmit only 64 bits per second reliably over the audio channel. This is due to the used basic version of the frequency-shift-keying modulation (FSK) this work is based on. The used frequency range is between 16kHz and 20kHz. Thus, the bandwidth of the audio communication channel is extremely limited.

Based on this, the protocol requirements are derived. Evidently, the secure communication protocol has to provide confidentiality, integrity and authenticity. Moreover, it must not rely on any additional infrastructure, which means that the two parties cannot communicate with a trusted third party or other public key infrastructures over a different channel. As the current implementation of the signal modulation can only transmit up to 64 bits per second, the overhead produced by the protocol should be as small as possible. The number

ISBN 978-0-9928626-7-1

of handshake messages has to be as little as possible as well since it takes quite long until a device can switch from receiving to transmitting. Additionally, the protocol must support a communication between two parties with equal roles. Therefore, both parties must be able to initiate the communication, assuming that the initiating party has input capabilities and can display a six-digit alphanumeric value. Lastly, all used algorithms to ensure the security standard should be open-source which means they should be publicly available and royalty-free.

As explained in section I, known audio communication protocols cannot be used in this work since they either do not enforce any cryptographic algorithms or they are proprietary and not published. Common data communication protocols can also not be used. Infrastructure-based communication protocols such as WPANs, WLANs and LoRaWANs need either additional devices or their communication is based on coordinator nodes that are superior to normal nodes [16]–[18]. Other popular communication protocols are non-infrastructure-based protocols such as NFC, RFID or Bluetooth. The standard of NFC does not include any cryptographic protocols to protect against eavesdropping, data modification and Man-In-The-Middle (MITM) attacks [19]. Common RFID communication is reader-based which means that the reader has to activate the tag in order to communicate with it [20]. Bluetooth allows the secure communication between two devices with equal roles. In combination with the Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol, it improves the security by protecting against passive eavesdropping and MITM attacks during pairing [21]. Albeit, it makes use of additional authentication stages since the public keys are exchanged over an insecure channel during the execution of the ECDH protocol. Unauthenticated ECDH does not guarantee any security against MITM attacks. Thus, Bluetooth requires the exchange of at least five messages during the handshake [22]. This is still too much overhead for the secure audio channel communication protocol.

Even though the aforementioned protocols are not applicable to the application scenario, some of their techniques are still very helpful for the development of the secure audio communication protocol. The analysis of the state-of-the-art protocols has revealed that cryptographic algorithms are necessary to establish a secure communication. In this context, authenticated encryption algorithms have been analyzed. They ensure confidentiality, authenticity and integrity of the exchanged messages. Two common algorithms are Counter with CBC-MAC (CCM) and Galois/Counter Mode (GCM). Even though GCM is more efficient than CCM, the CCM in combination with AES (AES-CCM) has still been selected as authenticated encryption algorithm for our protocol as the security of GCM is significantly decreased if it is used with small tags [23]. AES-CCM does not have this security flaw and can be securely used with authentication tags as short as 32 bits (AES-CCM-32) if the restrictions given by [24] are satisfied [25], [26].

In order to setup a secure communication channel without any additional infrastructure beside a microphone and loudspeaker, pre-shared passwords can be used. However, they must not be directly used as encryption keys. Rather, a password-based key derivation function is required. The Password-Based Key Derivation Function 2 (PBKDF2) has been chosen as a key derivation function since it is well established and NIST-approved [27]. It is proven to be secure if the number of iterations of PBKDF2 is set to the highest tolerable value [28]. Lastly, the analysis has shown that it is necessary to verify the integrity of the handshake messages. That is why SHA-2 is selected as the cryptographic hash function to compute a hash over the handshake message, since it is very fast to compute and has been proven to be secure [29].

III. Proposed Protocol

The proposed protocol allows the secure communication between two parties, the party that initiates the communication to transmit a message is denoted transmitter; the other party is labeled as receiver. It is divided into three stages: password agreement, handshake and data transmission. If the password is not predefined because of the lack of input capabilities, the receiver generates a password by means of the password generator. The password is required since key exchange protocols such as ECDH need to be authenticated in order to be secure against MITM attacks. These authentication stages produce too much overhead during the handshake. That is why the password generator is implemented in order to create the root of trust. In general, security is assured only if the password generator is used. The generated password has to be typed into the receiver. Even if the password is predefined and displayed on one of the devices, it is strongly encouraged to use the password generator to generate this password in advance. Otherwise, security cannot be guaranteed. By default, the password generator randomly selects eight ASCII characters ranging from `0x20` to `0x7E`. These values consist of all printable characters of the ASCII table including all upper and lower case, special characters, digits and the space character. If stronger security is required, the parties can agree on a longer password by increasing the desired password length. As soon as both parties have stored the password, the handshake stage can take place. In the following, the handshake and data transmission will be explained on the transmitter's side. The receiver's side can then be easily derived.

A. Transmitter

At first, the transmitter has to reset its local 32-bit counter. The counter indicates how many messages have already been sent. Since the handshake establishes a new key together with a new random explicit nonce, the counter has to be reset to its initial value `0x00000001`. The transmitter then generates a random 256-bit salt. This salt defines the handshake message that has to be sent to the receiver in the later process. Additionally, the salt and the password that has been set in the previous step serve as input for the key derivation function PBKDF2 in which the Keyed-Hash Message Authentication

ISBN 978-0-9928626-7-1 © EURASIP 2017

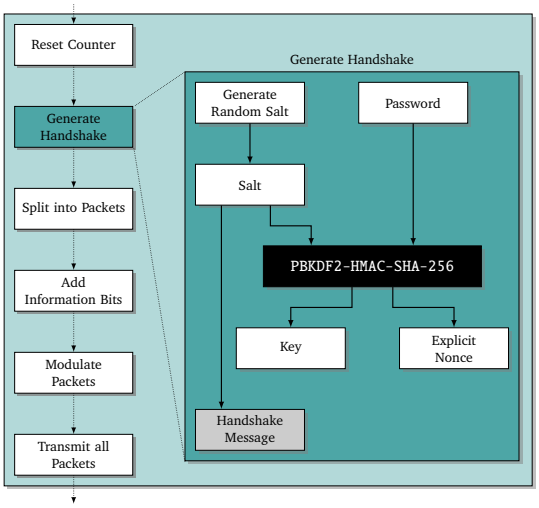

Fig. 1: Transmission of Handshake

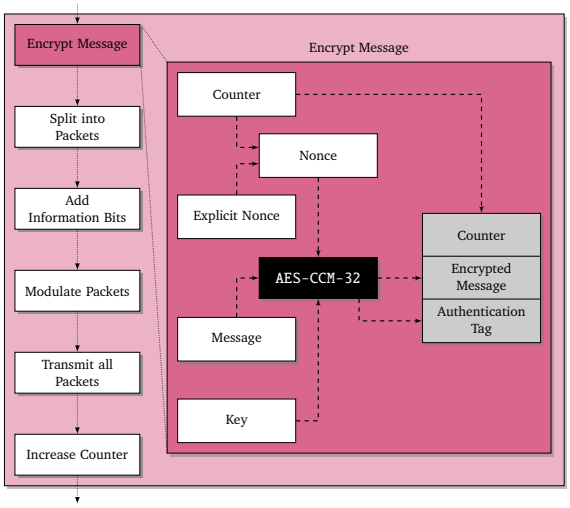

Fig. 2: Encrypted and Authenticated Transmission

Code in combination with SHA-256 (HMAC-SHA-256) is used as pseudo-random function. The function outputs a 128-bit key and a 32-bit explicit nonce. Both values have to be stored at the transmitter for later use. After the 256-bit handshake message has been generated, it must be split into four 64-bit packets. This is due to the limitations of the signal modulation implementation used in this work, since only small packets can be transmitted. Before the modulation of all packets can take place, three information bits are prepended to each packet. The first information bit is the Even Message (EM) information bit. It describes if the index of the current packet is even or not. In general, the very first packet of each message has the index zero. Thus, the EM bit of the very first packet of every message is always set. The next information bit Last Message (LM) defines if this packet is the last packet of the message. Lastly, the information bit HS announces a Handshake Message. Hence, the total size of each packet is 67 bits. Lastly, all packets are modulated and transmitted by playing audio signal on the speaker. Figure 1 shows the whole handshake stage of the transmitter.

In order to verify the integrity of the handshake message to prevent against MITM attacks, a checksum computation of the handshake message is required. Thus, both transmitter and receiver take the 256-bit handshake message and compute a 256-bit hash value by means of the SHA-256 algorithm. They then truncate the hash down to 30 bits and transform the truncated hash into an alphanumeric value. The alphanumeric value space consists of all digits from zero to nine and all capital letters of the English alphabet excluding `I`, `O`, `Q` and `U`: $\mathit{alpha} \in \{\texttt{0},\ldots,\texttt{9},\texttt{A},\ldots,\texttt{Z}\} \setminus \{\texttt{I},\texttt{O},\texttt{Q},\texttt{U}\}$, where `0` is the lowest value and `Z` describes the highest value: $\texttt{0} \coloneqq 0$ and $\texttt{Z} \coloneqq 32$. The set $\{\texttt{I},\texttt{O},\texttt{Q},\texttt{U}\}$ is excluded because `I` looks similar to `1`, while `O` can be confused with `0` or `Q`, and `V` does not differ much from `U`. For that reason, the alphanumeric space has 32 characters and a six-digit value can describe any number from zero to $2^{30}-1$. This technique has also been explained and evaluated in the paper of Kainda et al. [30]. After the computation has succeeded, the alphanumeric value is shown on the displays of both devices. The users of both devices now have to confirm that the alphanumeric values are the same. If one device does not have any input capabilities, the confirmation is only unilateral on the other device. At least one device has to have input capabilities in order to confirm the checksum value.

After the handshake message has been exchanged and the 128-bit key and 32-bit explicit nonce have been computed by means of the key derivation function, the actual transmission of authenticated and encrypted messages can take place. All messages are encrypted by means of the AES-CCM-32 algorithm which requires four inputs: secret key, nonce, plaintext and Additional Authenticated Data (AAD). The key generated by the key derivation function serves as the secret key, while the nonce is the concatenation of the 32-bit explicit nonce and the 32-bit counter. The counter is also the input for the AAD. Evidently, the message defines the plaintext. After the authenticated encryption has taken place, AAD, ciphertext and authentication tag are concatenated to one message. Subsequently, the whole message has to be split again into 64-bit packets and the information bits are prepended to each packet. Then, the message will be modulated and all 67-bit packets will be transmitted by playing the signal on the speakers of the transmitting device. In the last step of the transmission, the transmitter increases its counter variable. Figure 2 illustrates the flow of the data communication at the transmitter's side.

All non-handshake messages are authenticated and encrypted. In general, a message consists of three parts. The first is the 32-bit counter. It is authenticated and describes the current index of the message. Additionally, it is used to prevent replay attacks and to check if a new handshake has to take place. This is followed by the authenticated ciphertext. Its length is defined by $n * 64$ bits with $n > 0$. Possible maximum message lengths will be explained in section IV. The message is completed by the 32-bit authentication tag which is appended to the ciphertext. All messages need to

ISBN 978-0-9928626-7-1 © EURASIP 2017

have a length which is a multiple of our packet size of 64 bit and are padded otherwise. Due to the fact that counter and authentication tag add up to 64 bits and every packet needs three information bits, the total overhead is given by $64+3*(64+n*64)/64 = 67+3*n$ bits. The transmission of modulated packets works bidirectional. The reception of every packet is confirmed by the receiver with a confirmReceived message in order to tell the transmitter when it can send the next packet. There are two types of confirmReceived messages. $\text{confirmReceived}_{\text{EVEN}}$ is sent after the receiver has received a message in which the EM bit is set, whereas $\text{confirmReceived}_{\text{ODD}}$ is sent if the EM bit is not set.

B. Receiver

The protocol flow of the receiver can be analogously derived from the transmitter's side. The only difference is a simple Denial-of-Service (DoS) check before a received message will be processed. If there have already been transmitted x handshake messages in the last s seconds, the receiver times out for t seconds and rejects all incoming messages. Possible values for x, s and t are discussed in section IV. Otherwise, all eight 67-bit packets will be accepted and assembled to one message. After that, a replay check is called in order to increase security. Only if this check passes as well, the receiver will read the information bits. Otherwise, the whole message will be discarded and the 'receive message' module will be aborted. If the HS information bit is set, the receiver processes the handshake by calling the PBKDF2-HMAC-SHA-256 on the handshake message and the password. Additionally, it resets its counter. If the HS bit is not set, the receiver has to decrypt the encrypted message by means of the AES-CCM-32 algorithm. As soon as the message is encrypted, the receiver increments its counter value. If the AES-CCM-32 algorithm fails, the message and its counter value get discarded.

IV. Security Evaluation and Discussion

This work assumes that any adversary is able to perform an MITM attack on the communication channel by means of high-tech devices such as a directional microphone. A secure communication protocol has to ensure four security properties: confidentiality, integrity, authenticity and availability [31]. Confidentiality and authenticity are ensured by the authenticated encryption algorithm AES-CCM-32. The authenticity of the messages also implies their integrity. Availability is hard to ensure by the protocol. It should be ensured by the design of the high-frequency audio channel. As a result, the availability of the channel is out of the scope of this work. However, the protocol offers some means to ensure the availability of the devices by preventing against DoS attacks. A handshake timeout t after x handshake messages received within the last s seconds has been introduced to prevent from unnecessary PBKDF2-HMAC-SHA-256 computations since the shared key is derived by the password-based key derivation function PBKDF2-HMAC-SHA-256. The selection of appropriate t, x and s values describes a trade-off between usability and security. Possible values for the transmission speed of 64 bits per second are $x = 4$, $s = 20$, $t = 40$ or $x = 10$, $s = 60$, $t = 120$. Both suggestions for the three parameters block DoS attacks for two thirds of the time.

Since availability is out of scope of this work, only the use of PBKDF2, CCM and SHA-2 is evaluated in this context. The analysis by Visconti et al. [28] has shown that PBKDF2 is suitable to derive secure keys if the passwords have a high entropy and the number of iterations of PBKDF2 is set to the highest tolerable value. In general, the entropy of user-selected passwords is not high enough if short passwords are allowed or there is no complex password policy that enforces more complex passwords [32]. The proposed protocol circumvents this problem by using a password generator that generates random passwords. The password entropy of the minimum eight-digit password generated by the password generator based on 94 ASCII characters can be stated with approximately $94^8 \approx 2^{52}$. However, it is also difficult to set the number of iterations to the highest tolerable value since the protocol should also be able to run on resource-constrained devices. Having more than 18,000 iterations is not recommended since older mobile devices such as a Samsung Galaxy S3 would take more than 1s to perform the PBKDF2 computation. Moreover, there will be collisions in the explicit nonce after about 2^{16} PBKDF2-HMAC-SHA-256 calls. Assuming that the precomputation attack described in [33] is feasible, the PBKDF2-HMAC-SHA-256 function should not be called more than 2^{16} times.

Furthermore, the analysis by Jonsson and Rogaway [25], [26] has revealed that AES-CCM-32 is secure if the maximum number of unauthenticated messages Max_{Err} that are allowed before the key will be retired is limited to the smallest possible value. In this context, a trade-off between security and unnecessary handshakes has to be found since there might be transmission errors in the communication channel. The lower the Max_{Err} value, the more secure the protocol but also the more the false negatives that would lead to unnecessary additional handshakes. The author of [24] states that the block cipher algorithm of AES-CCM-32 should not be called more than 2^{61} times under the same key. In order to increase the security assurance, this limit is decreased to 2^{50} times. Therefore, there must not be more than 2^{49} blocks sent over the channel encrypted with the same key. In AES-CCM-32, the maximum plaintext length and the length of the complete nonce used as input depend on each other. Since the complete nonce is 64 bits long, the maximum plaintext length must not be longer than 2^{59} bits which is 64 petabytes.

In addition, an analysis of the checksum verification has been conducted. Since the checksum verification is based on a 30-bit hash computed over the handshake message, collisions are expected after about 2^{15} handshakes. In order to prevent this, the two parties have to agree on a new password after at most 2^{12} handshakes. The checksum verification follows the compare-and-confirm approach described by Kainda et al. [30]. They have shown that this approach provides the best usability; however, it is subject to security failures. There are other approaches such as copy-and-enter or barcodes that are more secure. Albeit, they decrease the usability too much

ISBN 978-0-9928626-7-1 © EURASIP 2017

or do not fulfill the requirements of the application scenario introduced in this work.

In order to limit the attack possibilities, the message limit is set to 2^{16} messages. Based on this message limit, the maximum message length has been derived as 2^{40} bits. However, this limit might be too loose for the current transmission speed of $64\frac{\text{bits}}{\text{s}}$. That is why a message length of at most 2^{15} bits is recommended. In total, 2^{28} messages can be sent over the audio communication channel before the users have to agree on a new password.

The open-source cryptographic library Crypto++ version 5.6.5 [34] was used for the implementation of the protocol to secure the validity and correctness of the cryptographic algorithms.

V. Conclusion

This work introduced a protocol for secure communication over an inaudible audio channel with smallest possible overhead for both handshake and data transmission. With the proposed protocol the low-bandwidth audio channel can be used to securely exchange data. The handshake consists only of a single 256-bit message which is used to generate the authenticated encryption key by means of the password-based key derivation function PBKDF2-HMAC-SHA-256. In order to verify the handshake message and thus prevent MITM attacks the checksum is calculated with SHA-256. The pre-shared password together with the nonce and counter steer the AES-CCM-32 encryption to ensure the confidentiality and authenticity of the data. Together with the counter and three information bits that are required for transmission, the message overhead sums up to only 67 bits plus three bits per 64 bit block.

References

[1] P. Lieb, "Secure communication over an audio channel," Master's thesis, Technische Universität Darmstadt, Germany, 11 2016.

[2] J. Lee, S. Dhar, R. Abel, R. Banakis, E. Grolley, J. Lee, S. Zecker, and J. Siegel, "Behavioral hearing thresholds between 0.125 and 20 khz using depth-compensated ear simulator calibration," *Ear and hearing*, vol. 33, no. 3, p. 315, 2012.

[3] Verifone, "Verifone way2ride," http://www.verifone.com/industries/taxi/way2ride/, [Online; accessed 28-February-2017].

[4] M. Knoll, "Mobile payment startup clinkle aims to change how we pay for things," http://trendblog.net/mobile-payment-startup-clinkle-aims-to-change-how-we-pay-for-things/, [Online; accessed 28-February-2017].

[5] S. Sachs, "Ultrasound whispers beat out gps," http://teleautomaton.com/post/1478772622/ultrasound-whispers-beat-out-gps, [Online; accessed 28-February-2017].

[6] P. Bihler, P. Imhoff, and A. B. Cremers, "Smartguide - a smartphone museum guide with ultrasound control." in *ANT/MobiWIS*, ser. Procedia Computer Science, vol. 5. Elsevier, 2011, pp. 586–592.

[7] "Signal360," http://www.signal360.com/, [Online; accessed 28-February-2017].

[8] Yamaha Corporation, "New way of transmitting information by audio frequency: Yamaha and fuji television agree to business alliance toward realizing new smartphone services that utilize infosound," https://archive.yamaha.com/en/news_release/2012/20120613a.html, [Online; accessed 28-February-2017].

[9] Hack Cave: The Root of Security, "Silverpush code : Tv ads can sent & execute ultrasonic secret commands to smartphone!" http://www.hackcave.net/2015/11/silverpush-code-tv-ads-can-sent-execute.html, [Online; accessed 28-February-2017].

[10] BBC News - Technology, "Google buys sound authentication firm slicklogin," http://www.bbc.com/news/technology-26222424, [Online; accessed 28-February-2017].

[11] Chirp, "Chirp: The internet of sound has arrived," http://www.chirp.io/, [Online; accessed 28-February-2017].

[12] C. Soriente, G. Tsudik, and E. Uzun, "Hapadep: Human-assisted pure audio device pairing," in *Proceedings of the 11th International Conference on Information Security*, ser. ISC '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 385–400.

[13] W. R. Claycomb and D. Shin, "Secure device pairing using audio," in *43rd Annual 2009 International Carnahan Conference on Security Technology*, Oct 2009, pp. 77–84.

[14] M. T. Goodrich, M. Sirivianos, J. Solis, C. Soriente, G. Tsudik, and E. Uzun, "Using audio in secure device pairing," *International Journal of Security and Networks*, vol. 4, no. 1-2, pp. 57–68, 2009.

[15] Y. S. Kim, S. H. Kim, and S. H. Jin, "Srs-based automatic secure device pairing on audio channels," in *2010 International Conference for Internet Technology and Secured Transactions*, Nov 2010, pp. 1–6.

[16] J. Adams, "An introduction to ieee std 802.15.4," in *2006 IEEE Aerospace Conference*. IEEE, 2006.

[17] C. Yang and G. Gu, "Security in wireless local area networks," in *Wireless Network Security: Theories and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 39–58.

[18] LoRa Alliance, "Lorawan specification," *LoRa Alliance*, 2015.

[19] G. V. Damme and K. Wouters, "Practical experiences with nfc security on mobile phones," in *Proceedings of the RFIDSec'09 on RFID Security*, 2009.

[20] C. C. Tan and J. Wu, "Security in rfid networks and communications," in *Wireless Network Security: Theories and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 247–267.

[21] J. Padgette, K. Scarfone, and L. Chen, *Guide to Bluetooth Security: Recommendations of the National Institute of Standards and Technology (Special Publication 800-121 Revision 1)*. USA: CreateSpace Independent Publishing Platform, 2012.

[22] Bluetooth Special Interest Group, "Simple pairing whitepaper," http://mclean-linsky.net/joel/cv/Simple%20Pairing_WP_V10r00.pdf, 2006.

[23] N. Ferguson, "Authentication weaknesses in gcm," http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf, 2005, [Online; accessed 28-February-2017].

[24] M. Dworkin, *SP 800-38C. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*. Gaithersburg, MD, United States: National Institute of Standards & Technology, 2004.

[25] J. Jonsson, *On the Security of CTR + CBC-MAC*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 76–93.

[26] P. Rogaway, "Evaluation of some blockcipher modes of operation," *Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan*, 2011.

[27] M. Turan, E. Barker, W. Burr, and L. Chen, *SP 800-132. Recommendation for Password-Based Key Derivation: Part 1: Storage Applications*. Gaithersburg, MD, United States: National Institute of Standards & Technology, 2010.

[28] A. Visconti, S. Bossi, H. Ragab, and A. Calò, *On the Weaknesses of PBKDF2*. Springer International Publishing, 2015, pp. 119–126.

[29] R. Sobti and G. Geetha, "Cryptographic hash functions: a review," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 2, pp. 461–479, 2012.

[30] R. Kainda, I. Flechais, and A. W. Roscoe, "Usability and security of out-of-band channels in secure device pairing protocols," in *Proceedings of the 5th Symposium on Usable Privacy and Security*, ser. SOUPS '09. New York, NY, USA: ACM, 2009, pp. 11:1–11:12.

[31] R. J. Sutton, *Secure Communications*. John Wiley & Sons, Ltd, 2002.

[32] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing metrics for password creation policies by attacking large sets of revealed passwords," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 162–175.

[33] D. Whiting, R. Housley, and N. Ferguson, "Counter with cbc-mac (ccm)," Internet Requests for Comments, RFC Editor, RFC 3610, September 2003, http://www.rfc-editor.org/rfc/rfc3610.txt.

[34] W. Dai, "Crypto++ library, 5.6.5 release," https://www.cryptopp.com/release565.html, [Online; accessed 28-February-2017].

ISBN 978-0-9928626-7-1 © EURASIP 2017