Subverting Counter Mode Encryption for Hidden Communication in High-Security Infrastructures

Alexander Hartl
Joachim Fabini
alexander.hartl@tuwien.ac.at
joachim.fabini@tuwien.ac.at
Institute of Telecommunications, TU Wien
Vienna, Austria

Christoph Roschger
christoph.roschger@tgm.ac.at
TGM - Vienna Institute of Technology
Vienna, Austria

Peter Eder-Neuhauser
Marco Petrovic
Roman Tobler
peter.eder-neuhauser@wienernetze.at
marco.petrovic@wienernetze.at
roman.tobler@wienernetze.at
Wiener Netze GmbH
Vienna, Austria

Tanja Zseby
tanja.zseby@tuwien.ac.at
Institute of Telecommunications, TU Wien
Vienna, Austria

ABSTRACT

In highly security-critical network environments, it is a popular design decision to offload cryptographic tasks like encryption or signature generation to a dedicated trusted module or key server with paramount security features, we in this paper refer to with the general term Cryptographic Key Management Device (CKMD). While this network design yields several benefits, we demonstrate that the use of popular counter mode encryption modes like CTR or GCM can show substantial shortcomings in terms of security when used in conjunction with this network design. In particular, we show how the use of authenticated encryption using GCM enables the possibility of establishing a subliminal channel by exploiting the authentication information within messages. We show how decoding of hidden information can proceed in addition to decryption of overt information without raising authentication failures.

With an exemplary but typical infrastructure, we show how the subliminal channel might be exploited and discuss approaches to mitigating the threat by preventing the ability to embed hidden information. In contrast to previous work, we conclude that, when using an infrastructure involving a CKMD and GCM is deployed, the use of random, CKMD-generated Initialization Vectors (IVs) is beneficial to avoid the subliminal channel described in this paper. However, the most potent remedy is deploying a different operational mode like GCM-SIV.

ARES 2021, August 17–20, 2021, Vienna, Austria

ACM ISBN 978-1-4503-9051-4/21/08…$15.00
https://doi.org/10.1145/3465481.3470082

CCS CONCEPTS

• **Networks** → *Network architectures*; • **Security and privacy** → **Symmetric cryptography and hash functions**; **Network security**.

KEYWORDS

Information leakage, subliminal channels, counter mode encryption, GCM

ACM Reference Format:
Alexander Hartl, Joachim Fabini, Christoph Roschger, Peter Eder-Neuhauser, Marco Petrovic, Roman Tobler, and Tanja Zseby. 2021. Subverting Counter Mode Encryption for Hidden Communication in High-Security Infrastructures. In *The 16th International Conference on Availability, Reliability and Security (ARES 2021), August 17–20, 2021, Vienna, Austria.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3465481.3470082

1 INTRODUCTION

In high-security infrastructures, use of cryptographic methods is ubiquitous to ensure confidentiality, integrity and authenticity of transmitted data. The most frequent requirement is to provide both, message confidentiality and authenticity, which has led to the development of Authenticated Encryption with Associated Data (AEAD) encryption modes. An AEAD algorithm performs encryption of the data and additionally computes an authentication tag, thus providing both, confidentiality and authenticity.

As of today, the most popular method to perform authenticated encryption is to use Galois/Counter Mode (GCM), utilizing Advanced Encryption Standard (AES) as underlying block cipher.

Additional security improvements can result from restricting involved parties' direct access to secret values. Tasks like encryption, decryption and authentication can be offloaded to a dedicated device like, e.g., a separate network server or a dedicated hardware module. In this paper, we denote such devices as Cryptographic Key Management Devices (CKMDs). Secret keys, which are considered

among the most valuable pieces of information to be kept confidential, can thus be stored in a single location that is equipped with paramount security features. Architectures deploying CKMDs are often used within critical infrastructures, aiming to meet highest demands in security and privacy.

When designing a high-security infrastructure it thus seems natural to combine both, AEAD encryption and the use of CKMDs. However, as we show in this paper, the straightforward use of counter mode encryption such as GCM, CCM, CWC, EAX or the plain Counter Mode (CTR) in such a scenario opens doors for attack options that allow the circumvention of the CKMD in encryption and decryption tasks, cancelling a large part of benefits the CKMD provides.

For example, since a message's authentication tag does not carry actual information, it might be exploited by malware to carry hidden information, which can pose a major threat in a critical infrastructure, as it evades detection possibilities by network monitoring software and Intrusion Detection Systems (IDSs). When using a CKMD, it seems easy to prevent this communication possibility by denying decryption on the CKMD for messages with invalid authentication information. We show that, when using counter mode encryption like GCM, attack possibilities can emerge that allow circumvention of the CKMD's decryption interface in both decryption and authenticity verification, thus allowing the authentication tag to be used as a very attractive subliminal channel. Due to its immense practical relevance, we focus on GCM in our present work.

Hence, in this paper, we review attack possibilities that emerge from using GCM in conjunction with a CKMD. We analyze methods for exploiting the authentication tag for the establishment of a subliminal channel, which yields particularly attractive properties for an attacker with respect to hidden communication. To high-security infrastructures, this subliminal channel therefore poses a major risk. We additionally discuss approaches to remedying the risk originating from the described scenarios, concluding that the use of a different operational mode like GCM-SIV or of random, CKMD-generated Initialization Vectors (IVs) are the best approaches to avoiding attack possibilities.

2 RELATED WORK

GCM was proposed by McGrew and Viega [25] to overcome performance restrictions in processing rates of used encryption modes while providing provable security. The authors included a security proof in their original publication [25], but Iwata et al. [22] later uncovered a flaw in the security proof. Even though further security issues were discovered for GCM [13, 14, 18, 23, 28], GCM is generally being considered secure when used correctly. In particular, when showing that the original security proof is faulty, Iwata et al. additionally provided a new proof, which, however, has larger bounds. Also multi user security of GCM has been shown [8, 20].

In the present context, Joux's attack [23], termed the "forbidden attack", is noteworthy. The attack's name stems from the fact that it relies on IVs being reused, which is a scenario that is explicitly excluded in security proofs. In this case, security of GCM breaks down completely and universal forgery becomes possible. Later, Böck et al. [11] performed a scan of the Internet, showing that, at the time of their studies, a notable amount of TLS-protected webservers suffered from implementation issues which led to IV reuse.

In this paper, we are interested in exploiting GCM's properties for hidden communication. The exploitation of cryptography for hidden communication was originally introduced by Simmons [31, 32], who envisioned prisoners who communicate clandestinely using authentication data, and introduced the term "subliminal channel" for hiding information exploiting cryptographic algorithms. To date, subliminal channels have been found in various digital signature schemes [9, 10, 19, 33]. The field was extended by Young and Yung [36, 37], terming the field "kleptography" and introducing Secretly Embedded Trapdoor with Universal Protection (SETUP) attacks.

More recently, approaches in the spirit of subliminal channels and SETUP attacks have been investigated under the name Algorithm Substitution Attack (ASA) [6] and it has been investigated how randomness used for symmetric encryption can be used for leaking information [4–6]. For example, Armour and Poettering [4] showed how AEAD encryption can be used to leak information by raising spurious authentication failures, encoding information in the resulting message retransmission pattern. Schneier et al. [30] survey various methods for weakening cryptographic algorithms.

2.1 Our Contributions

With this paper, we aim to raise awareness for a false sense of security arising when combining counter mode encryption like GCM with architectural design deploying CKMDs. This finding is of particular significance, because both, GCM and the strategy of offloading cryptographic tasks to CKMDs are known to provide high security, and hence, are likely to be used in critical infrastructures where the existence of hidden communication facilities can be detrimental for privacy, security or even safety.

In this case, the CKMD seems to provide means for restricting the possibilities for cryptographic operations that can be performed. As we highlight in this paper, however, such restrictions can easily be bypassed in many cases when using counter mode encryption.

We discuss in detail a scenario where the CKMD allows both encryption and decryption of arbitrary messages and investigate whether the authentication tag of GCM can be abused for hidden communication while leaving the actual communication operational. Intuitively, a CKMD would be expected to insist on a correct authentication tag for performing decryption, leaving no space for embedding hidden information. We show that this intuition is misleading and that it is indeed possible to perform decryption despite a manipulated authentication tag. To demonstrate that our used scenarios are relevant in practice, we describe an exemplary scenario in the field of the Industrial Internet of Things (IIoT) and show that the subliminal channel can pose a major risk, while most known covert and subliminal channels can easily be prevented.

GCM is used in various protocols, e.g., in TLS [27, 29], IPsec [34], MACsec [3] and SSH [21]. Due to its immense practical relevance, we thus focus on GCM in this paper. However, while our technique for hidden communication seems to be very specific for GCM, in fact also other counter encryption modes like CCM [35], CWC [24] and EAX [7] follow an architectural design that allows applying

techniques as described in this paper for circumventing CKMDs to hide information in an authentication tag. We also provide several reasons for why, assuming a CKMD is deployed, it is likely that it is deployed in a way that allows attacks as described in this paper. Furthermore, our concept is generic in the sense that the sender of hidden information does not have to be co-located with the sender of overt communication, but instead can be located on any node on the transmission path that is able to modify the encrypted message. This property allows various scenarios for exploitation by an attacker.

To provide guidelines for protocol engineering, we finally discuss several approaches to mitigating attack possibilities. Beside the use of a different mode of operation like GCM-SIV [17], the most potent approach to avoid attack possibilities described in this paper is to generate the IVs on the CKMD, even if the IVs have to be chosen at random in this case due to limited capabilities of a CKMD. This is particularly interesting, as it has been shown previously that the use of random IVs instead of counting IVs *increases* susceptibility for hidden communication [6].

3 OFFLOADING CRYPTOGRAPHIC TASKS

In many architectures, cryptographic operations are offloaded to a specific device, in this paper referred to as CKMD. We introduce the term CKMD as an abstract term, which in practice occurs in different instantiations. In particular, a CKMD might be

- a dedicated network server, which performs, e.g., encryption, decryption or signature creation tasks for other network participants.
- a Trusted Platform Module (TPM), i.e. a dedicated device soldered on computers' mainboards to provide a secure key storage for cryptographic tasks.
- a Smart Card, functioning as portable security token.

The main reason for deploying a CKMD is to provide means to store and encapsulate cryptographic key material in a highly secure manner and restrict access to cryptographic functions using it to a well-defined, security and privacy preserving interface. In particular, often special hardware features are used to provide a secure storage area shielding keys from unauthorized access, possibly even despite physical access. A CKMD might additionally be equipped with special hardware for performing cryptographic tasks, like, e.g., a good source of randomness or hardware support for cryptographic algorithms.

Furthermore, a CKMD might be used to restrict the usage of cryptographic keys or monitor how they are used as well as the frequency of usage. For example, a CKMD might permit data to be encrypted, but prohibit the decryption of data, even though a symmetric key is used. However, as we show in the following, possibilities for restricting usage might be easily bypassed when using an operational mode like GCM.

4 GCM ENCRYPTION

Frequently it is necessary to both encrypt and authenticate transmitted messages. Encryption modes that provide both functionalities are known as AEAD. A very popular AEAD encryption mode is GCM. We summarize GCM encryption in this section.

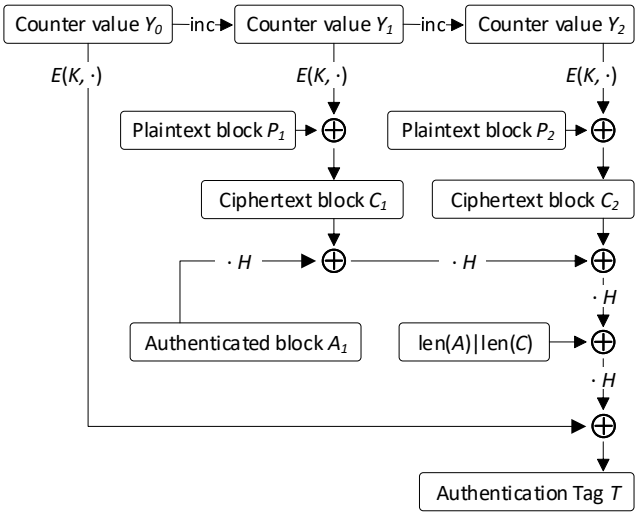

Figure 1: The GCM encryption process for two blocks of encrypted data and one block of additional authenticated data.

4.1 Encryption and Decryption

When processing 128-bit plaintext blocks P_i with $1 \leq i \leq n$ with a user-provided IV, GCM produces ciphertext blocks C_i and an authentication tag T. The encryption operation of GCM is defined by the following set of equations [25]:

$$
\begin{aligned}
H &= E(K, 0^{128}) \\
Y_0 &= \begin{cases} \mathrm{IV}|0^{31}|1 & \text{if } \mathrm{len}(\mathrm{IV}) = 96 \\ \mathrm{GHASH}(H, \{\}, \mathrm{IV}) & \text{otherwise} \end{cases} \\
Y_i &= \mathrm{inc}(Y_{i-1}) \quad \forall\, i = 1, \ldots, n \\
C_i &= P_i \oplus E(K, Y_i) \quad \forall\, i = 1, \ldots, n-1 \\
C_n^* &= P_n^* \oplus (E(K, Y_n)[:\mathrm{len}(P_n^*)]) \\
T &= (\mathrm{GHASH}(H, A, C) \oplus E(K, Y_0))[:L_T]
\end{aligned}
$$

Here, $E(\cdot)$ denotes encryption using the underlying block cipher (e.g. AES [2]), K denotes the symmetric key used for encryption, $\oplus$ denotes the XOR operation or, equivalently, addition in GF(2^{128}) and L_T denotes the desired length of the authentication tag T in bits. Furthermore, we denote by $x[:N]$ the N most significant bits of x and by $x[N:]$ the N least significant bits of x. len($\cdot$) returns its argument's length in bits and inc($\cdot$) returns its argument incremented by one in big-endian representation.

The function GHASH() is obtained from X_i defined according to the equations [25]

$$
X_i = \begin{cases}
0^{128} & \forall\, i = 0 \\
H \cdot (X_{i-1} \oplus A_i) & \forall\, i = 1, \ldots, m-1 \\
H \cdot (X_{i-1} \oplus (A_m^* | 0^{128-\mathrm{len}(A_m^*)})) & \forall\, i = m \\
H \cdot (X_{i-1} \oplus C_{i-m}) & \forall\, i = m+1, \ldots, m+n-1 \\
H \cdot (X_{i-1} \oplus (C_n^* | 0^{128-\mathrm{len}(C_n^*)})) & \forall\, i = m+n \\
H \cdot (X_{i-1} \oplus (\mathrm{len}(A)|\mathrm{len}(C))) & \forall\, i = m+n+1,
\end{cases}
$$

where GHASH$(H, A, C) = X_{m+n+1}$. Here, $\cdot$ denotes multiplication in GF(2^{128}) and A_i denote 128-bit blocks of additional authenticated data with $1 \leq i \leq m$. The encryption operation is illustrated in Figure 1. The transmitted message then consists of the ciphertext blocks C_i and the authentication tag T.

Hence, similar to stream ciphers, encryption is done by performing an Exclusive Or (XOR) operation of the plaintext with the block cipher's output. Decryption can be performed by again performing an XOR operation with the ciphertext, eventually allowing computation of the authentication tag similar to the encryption operation.

AES-GCM is a very popular cipher. For example, it is used in TLS [27, 29], IPsec [34], MACsec [3] and SSH [21]. In TLS version 1.3 AES-GCM is the only cipher that has to be implemented by standard-compliant implementations [27].

Reasons for the widespread use of AES-GCM boil down to the reasons that motivated the development of GCM in the first place. In particular, due to the way GCM is engineered, encryption and decryption are parallelizable and can be efficiently implemented in hardware and software [25, 26], thus allowing vast amounts of data to be processed and deployed in high rates. The key stream used for encryption and decryption can be precomputed in advance, which constitutes a convenient feature for many applications. Furthermore, GCM is provably secure [22] and was designed to be free of patents [25].

4.2 IV Selection

As will become apparent later, it is of utmost importance to avoid encrypting distinct messages with the same IV. In general, IVs can be chosen randomly or deterministically by incrementing the IV for each processed message by one. For example, the NIST recommendation [12] allows the following methods for creating IVs:

For deterministic construction, [12] divides the IV field into two subfields, the fixed field and the invocation field. The fixed field identifies the context for the encryption, so that, e.g., for each device a unique value is chosen. For a specific value of the fixed field, the invocation field is then simply incremented for each processed message or generated using a linear feedback shift register.

For Random Bit Generator (RBG)-based construction, a random IV is generated for each processed message. If the IV is chosen at random, the uniqueness of occurrences of IV values cannot be guaranteed. Hence, [12] defines an upper limit of 2^{-32} for the probability of ever assigning the same IV to distinct messages, thus limiting the number of processed messages with the same key. For example, for an IV length of 96 bits, up to 2^{32} messages can be processed with the same key without violating the requirement.

The benefit of RBG-based construction is to avoid having to keep a state for the encryption operation and the risks of flawed state-keeping. However, for RBG-based construction only a probabilistic statement can be made for the reoccurrence of IVs. It is also noteworthy that for the same number of processed messages, the IV needs to be considerably longer than for deterministic construction. Furthermore, if both sender and receiver can be constructed stateful, IV values do not even need to be transmitted for deterministic construction, as the receiver can construct them on his own. Furthermore, the deterministic construction has been argued to reduce facilities for hidden communication [6].

For these reasons, deterministic construction becomes increasingly popular. For example, in IPsec [34] and TLS version 1.2 [29] IVs are transmitted with each message, leaving selection of suitable values to the implementation, but suggesting simply using a counter. In TLS version 1.3 [27] and SSH [21] IVs are no longer transmitted, but implicitly computed based on a counter.

5 CKMD ARCHITECTURES AND COUNTER MODE ENCRYPTION

As described in the previous Section 4.2, IV selection frequently takes place deterministically by incrementing a counter for each processed message. When deploying a CKMD, the question arises whether to implement this counting procedure on the CKMD itself, or rather on the requesting device, transmitting it to the CKMD within the encryption request.

We argue that in most practical scenarios the counter indeed is implemented on the requesting device for reasons of robustness, cost, simplicity and hardware design. On the one hand, CKMDs in many cases are low-cost devices, where state-keeping is difficult. On the other hand, even if state-keeping is possible, robustness of the communication might be at stake or would at least require additional complexity when computing deterministic IVs on the CKMD. For example, if the connection between requestor and CKMD is unreliable, the delivery of an encryption response might fail. In this case, if the state on the CKMD has been updated anyhow, a wrong IV would be the consequence when rerequesting encryption, leading to failing decryption if the IV is computed implicitly on the receiver side.

We thus argue that in practice a reasonable scenario is that the IV is chosen by the requestor and submitted to the CKMD for each message, leaving the requestor full control over which IV is used. However, if the requestor is in full control about the used IVs, it can perform several operations described in this section, which are usually implicitly considered impossible.

5.1 Decrypting Messages

A CKMD might allow the encryption of messages, but prohibit the decryption of messages. If counter mode encryption modes are used, such constraints are elusive. Since counter mode encryption modes function like stream ciphers with respect to encryption and decryption, both encryption and decryption proceed by performing an XOR with a key stream obtained from the underlying block cipher. Hence, decryption of an encrypted message C_i can be performed by instead requesting encryption with the same IV, interpreting the returned ciphertext as decrypted plaintext.

However, the CKMD might monitor the messages to be encrypted and enforce a certain message syntax and semantics. This check can be circumvented as well by requesting the encryption of an arbitrary, syntactically valid, message R_i with the same IV, resulting in the ciphertext $C_{R,i}$. Decryption of the original message can then be performed by computing the XOR $P_i = C_i \oplus R_i \oplus C_{R,i}$.

This attack strategy might also be used for performing decryption independent of the authenticity of the message. If the CKMD

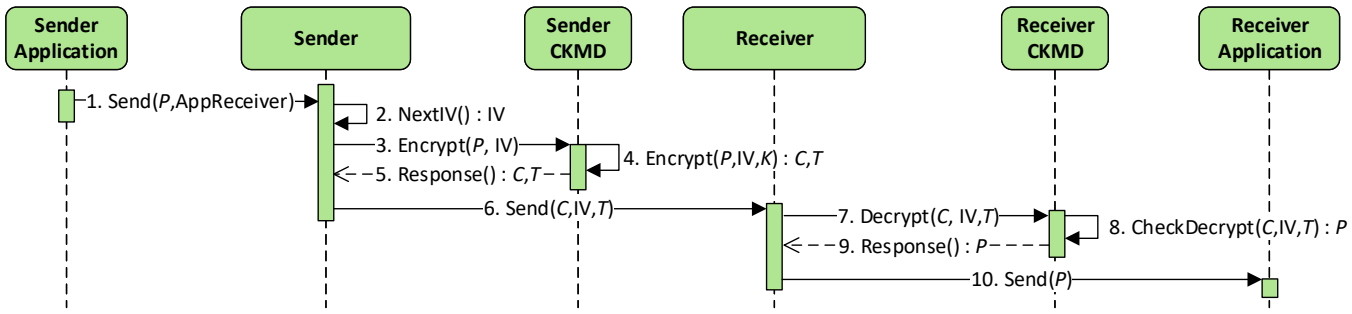

Figure 2: Normal device communication when using CKMDs.

determines that an encrypted message's authentication tag is wrong, it usually not only reports the failing authenticity check, but also denies decryption of the message. In Section 6, we will make use of this attack to decrypt messages while exploiting the authentication tag to carry hidden information.

The attack scenario described in this section is not limited to GCM, but rather constitutes a general weakness of counter mode encryption. In particular, it similarly applies to plain CTR [1] encryption, but also to the authenticated CCM [35], CWC [24] and EAX [7] encryption modes. It does not apply to the recent GCM-SIV [17] and AES-GCM-SIV [15, 16] modes, however, as the IV used for counter mode encryption is derived from the message itself for GCM-SIV and AES-GCM-SIV. Due to the practical relevance of GCM, we focus on GCM in this paper.

5.2 Encrypting and Authenticating Messages

Similar to the previous Section 5.1, an attacker is able to obtain an encrypted message C_i from plaintext P_i by requesting encryption of a different message of equal length instead. However, for an attacker this approach is only useful if he also is able to generate a valid authentication tag T.

From one observed ciphertext/plaintext pair an adversary can form the polynomial $c_1H^{m+n+1}+\ldots+c_{m+n}H^2+(\text{len}(A)|\text{len}(C))H+E(K,Y_0)=T$, where the coefficients c_i can be derived from the non-secret A_i and C_i. To authenticate arbitrary messages, the attacker needs to know H and $E(K,Y_0)$, which cannot be deduced from a single polynomial equation. However, as soon as a second message for the same IV is observed, the equation system becomes solvable, obtaining $E(K,Y_0)$ and a small number of candidates of H. Hence, by requesting the encryption of two messages with equal IV, it is possible to circumvent the CKMD in the creation of authenticated messages. This attack is known as the "forbidden attack" [11], as it relies on IVs being reused, which is a scenario that is explicitly excluded in security proofs for GCM.

Note, however, that the sole possibility for decryption does not enable the attacker to authenticate arbitrary messages, since he cannot obtain distinct plaintext pairs with the same IV in this case. He can, however, if encryption is allowed, easily circumvent restrictions with regards to the content or the amount of encrypted messages.

6 EXPLOITING GCM FOR SUBLIMINAL COMMUNICATION

We now describe how the use of GCM can lead to a subliminal channel for architectures where CKMDs are used. We target particularly high-security infrastructures, where deployed network protocols have been designed to minimize the possibility for covert channels and possibly even the use of digital signatures, which are prone for subliminal communication [9, 10, 19, 33], might have been avoided. At the same time, in security-critical infrastructures, the need for ensuring message integrity and authenticity is inevitable, leading to the use of AEAD encryption like GCM.

6.1 Using T for Subliminal Communication

Figure 2 shows device communication for normal, i.e. benign, operation. Here, both sending and receiving parties are equipped with CKMDs, which perform the actual encryption and decryption tasks. When the sending application transmits a message P, the Sender creates a new IV, passes both P and the IV to the CKMD for encryption in step 3 and sends the returned ciphertext C and authentication tag T to the Receiver in step 6. The Receiver passes C, T and the IV on to the CKMD in for decryption step 7, which verifies message authenticity and performs decryption. In case of successful verification of the authentication tag T, the CKMD returns the message P to the Receiver in step 9. Finally, the receiving application obtains the message P. If verification of T fails, the CKMD denies decryption and returns a decryption error.

6.1.1 A Trivial Covert Channel. We now describe how a subliminal channel can be established, exploiting the authentication tag for carrying hidden information. To this end, without going into detail about how compromisation might take place, we assume that the Sender and Receiver have been compromised and require means to communicate clandestinely thereafter. As we will demonstrate in Section 7, it is easily possible that Sender and Receiver cannot rely on covert channels of lower network protocols and, hence, can only use the message itself or its authentication tag for hidden communication. Sender and Receiver deploy CKMDs, which we do not assume compromised. Hence, Sender and Receiver do not have access to secret keys for encryption.

With this setup, we will encode hidden information into the authentication tag of encrypted messages in what follows. Hence,

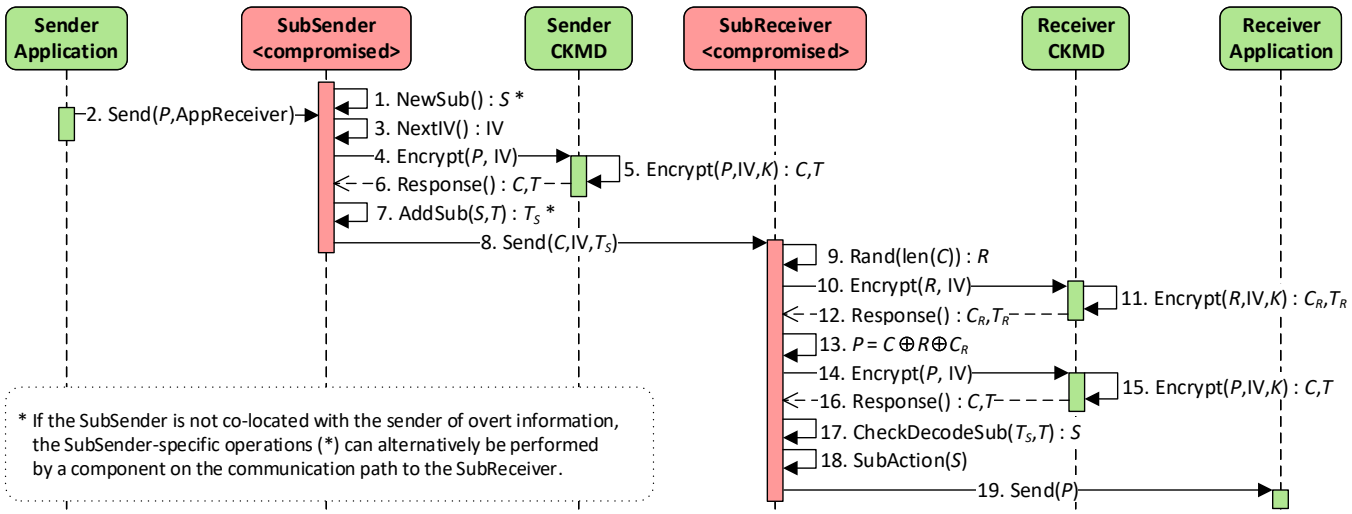

Figure 4: Subverting CKMDs in building a subliminal channel.

embedment of hidden information would cause the legitimate verification procedure to fail. While this seems unsuitable for subliminal or covert channels at first sight, since the encryption key is needed for verifying message authenticity, only Sender and Receiver can verify message authenticity. In our case, both Sender and Receiver are compromised, as they perform the hidden communication, and therefore do not report the existence of the subliminal channel.

Of course, someone in possession of the respective secret key might reveal existence of the subliminal channel. However, due to the secrecy of the appropriate key, this scenario is highly unlikely. Furthermore, assuming that random oracles are implemented using cryptographic hash functions, also traditional subliminal channels can be unveiled when holding the secret key.

To ease comprehension, we begin by presenting how a covert channel can be established when neither of the communicating parties deploy CKMDs. Figure 3 shows a scenario where hidden information is transmitted from the SubSender to the SubReceiver. In this scenario, it is possible to exploit the authentication tag for hidden information without exploiting the mechanisms described earlier in this paper.

Steps 1-4 in Figure 3 match normal operation, where SubSender obtains a message from the application and performs authenticated encryption, obtaining C and T. However, to clandestinely transmit information, SubSender encodes hidden information in step 5. The operation AddSub() accepts the authentication tag T and computes the altered authentication tag T_S, so that $T_S[L_T - L_S:] = T[L_T - L_S:] \oplus S$ and $T_S[:L_T - L_S] = T[:L_T - L_S]$, where L_S denotes the length of the hidden information S. Hence, the hidden information is encoded in the L_S least significant bits of T using an XOR operation.

The message C, T_S is then sent to the SubReceiver in step 6, which can perform decryption and computation of the expected authentication tag T in steps 7,8 as usual. However, instead of raising an authentication failure when the received tag does not match T, authenticity of the message is verified by only comparing the most significant $L_T - L_S$ bit of T. Furthermore, hidden information $S = T_S[L_T - L_S:] \oplus T[L_T - L_S:]$ can be recovered in step 9 in terms of an XOR with the L_S least significant bits of the authentication tag computed during decryption.

6.1.2 Circumventing CKMD Verification. We now assume that both SubSender and SubReceiver deploy non-compromised CKMDs, aiming to achieve superior security. Receiving a message with a faulty authentication tag, a CKMD is likely to deny message decryption and possibly even raise an alarm. Exploitation of the authentication tag as subliminal channel therefore does not seem feasible anymore without sacrificing decryptability of the message.

However, it is one of the specific properties of GCM encryption, which allows exploitation as subliminal channel also in this case. Figure 4 shows the steps that SubSender and SubReceiver undertake in submitting the message. Hence, in steps 4-6 similar to normal operation, the SubSender submits the message to its CKMD

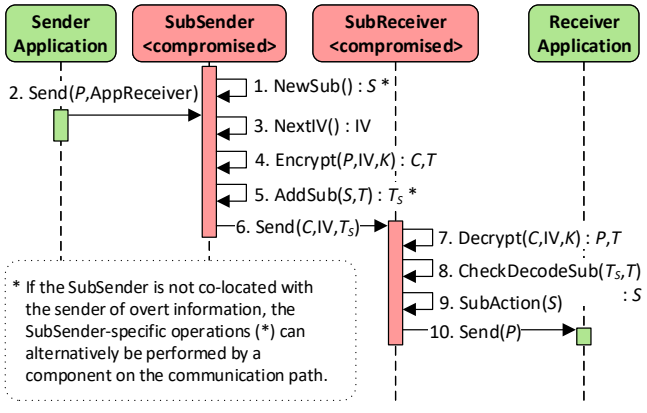

Figure 3: Exploiting the authentication tag for hidden communication.

for encryption. After receiving back the ciphertext blocks C_i and authentication tag T, however, the SubSender embeds hidden information S in step 7. Similar to the previous scenario, AddSub() encodes S into the least significant bits of T in terms of an XOR operation, obtaining T_S.

At the SubReceiver, processing of the message diverges from the previous scenario, as the encrypted message cannot simply be passed to the CKMD due to the manipulated authentication tag. As described in Section 5.1, the SubReceiver can perform decryption by issuing an encryption request. Hence, in step 9 the SubReceiver generates an arbitrary message R that has at least the same length as C and passes it to the CKMD for encryption in steps 10-12, using the same IV as in the received message. It receives back the ciphertext C_R and authentication tag T_R. Decryption of the ciphertext blocks C_i can now be performed in step 13 as $P_i = C_i \oplus R_i \oplus C_{R,i}$.

For both verifying message authenticity and decoding the hidden information, however, the original authentication tag T is needed. For this reason, the SubReceiver now passes the obtained plaintext blocks P_i to the CKMD for encryption in steps 14-16, obtaining the original authentication tag T. Similar to the previous scenario, the SubReceiver can now recover $S = T_S[L_T - L_S:] \oplus T[L_T - L_S:]$ and verify message authenticity by verifying the $L_T - L_S$ most significant bits of T_S in step 17. If verification passes, the SubReceiver processes S in step 18 and, similar to normal operation, passes P to the receiving application in step 19.

6.1.3 Proof of Concept Implementation. To verify functionality in practice, we used the Envelope (EVP) encryption interface of OpenSSL version 1.1.1k and implemented the steps depicted in Figure 4. We used `EVP_aes_256_gcm` as cipher. The 256-bit encryption key and 96-bit IV were chosen at random. Furthermore, we used a random authentication tag with length between 8 bytes and 16 bytes, a random message with length between 1 and 100 bytes and a random hidden message with length between 1 byte and $\text{len}(T)/8 - 1$ bytes. In all 10^9 runs we performed, verification of authenticity, decryption of overt message and recovery of the hidden message proceeded successfully.

6.2 Properties of the Subliminal Channel

In high-security environments, our presented subliminal channel has properties that might be particularly attractive to an adversary.

6.2.1 Deployment. The subliminal channel is agnostic to the used protocol. Hence, an adversary does not have to learn semantics of the used protocol, but only needs to compromise cryptographic algorithms. Since in normal operation preservation of the authentication tag is necessary to ensure successful authenticated decryption, an adversary can be sure that the subliminal information is retained from sender to receiver, independent of how many nodes are in between them. This is in contrast to covert channels in network headers, which might be destroyed by, e.g., devices performing Network Address Translation (NAT).

6.2.2 Concealment Properties. As discussed above, discovery of the subliminal channel is possible only if the encryption key is known, which usually is difficult to achieve in practice. Furthermore, even if the existence of the subliminal channel is known, decoding of the information is only possible when holding the encryption key.

Of course, the requests that are sent to the CKMD differ substantially from usual operations. Hence, statistical monitoring on the CKMD, or of the communication with the CKMD, might show abnormalities. However, considering the characteristics of the architecture, it is highly unlikely that the subliminal channel would raise suspicion. In particular, in most cases the CKMD lacks resources for performing statistical monitoring and performing anomaly detection on the observed data.

Also considering its communication, even if communication can be monitored, e.g., in a network environment, leaving this communication unencrypted would be detrimental for security. Hence, in a practical scenario this communication must be encrypted, e.g., using a TLS-protected transport, making communication monitoring highly challenging.

6.2.3 Bandwidth for Hidden Communication. For many practical security protocols, small chunks of information are encrypted and authenticated like, e.g., network packets. Hence, authentication tags are transmitted with a substantial frequency, offering a large bandwidth for subliminal communication. In particular, compared to subliminal channels known from digital signature schemes, the resulting number of requests to the CKMD and the available bandwidth for hidden communication in most cases are considerably higher.

6.2.4 Location of Compromised Devices. In Figure 4, we depicted a scenario where compromised parties coincide with the sender and receiver of benign communication. Indeed, the receiver of the hidden information can only coincide with the receiver of the overt information, as depicted in Figure 4, or, trivially, the receiver's CKMD. The secret key needs to be known to recover the hidden information and, furthermore, authenticity verification would fail if the receiver cannot be compromised.

On the other hand, for encoding the hidden information no secret information is needed. Hence, the sender of hidden information does not necessarily need to coincide with the sender of the overt information, but instead might be located on any intermediate node that is able to manipulate the transmitted message. We also want to note that, when being able to compromise both sender and receiver, it might in certain cases be possible to entirely circumvent CKMDs on both sender and receiver side and deploy an attacker-provided encryption, hence allowing to create other facilities for hidden communication. However, compared to a simple manipulation of the authentication tag, such approach clearly would require a more extensive compromisation of the involved devices, additional measures to avoid detection, and might overload the computational resources of low-power devices like, e.g., sensors, as encryption can no longer be offloaded to the CKMD. Furthermore, in comparison to such scenarios, dropping the requirement for compromising the sender allows the described subliminal channel to be used in various further scenarios. This property of the subliminal channel thus raises its significance substantially, as attackers can choose arbitrary locations on the communication path to unidirectionally communicate hidden information into a high-security infrastructure.

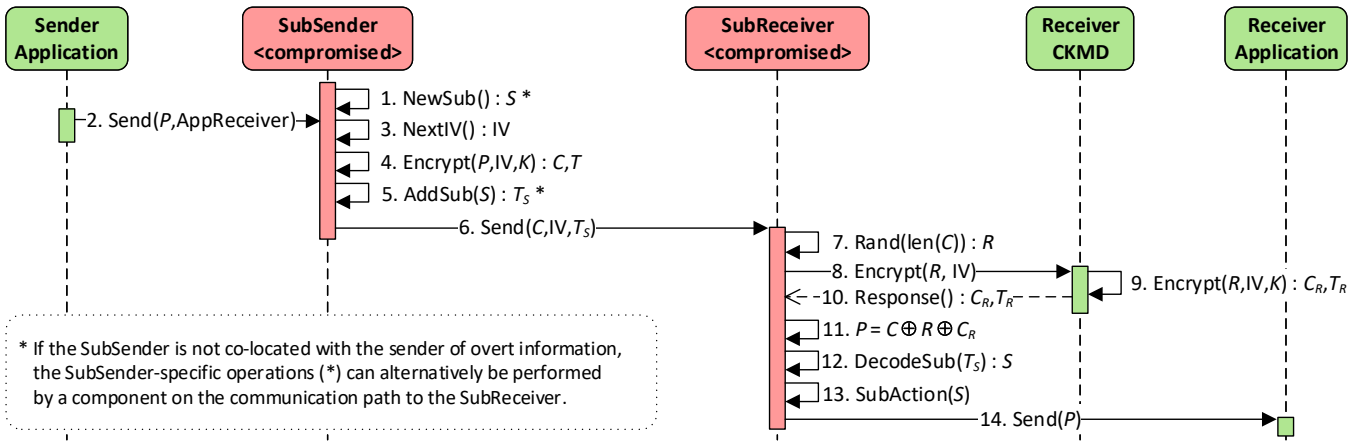

Figure 5: A simplified subliminal channel.

6.3 A Simplified Subliminal Channel

In Figure 5, we show a variant of the subliminal channel described in Section 6.1.2, which has slightly different characteristics. Steps 1-4 proceed as before. However, when skipping the second encryption request in Figure 4, the original authentication tag is no longer available for recovering S or ensuring message authenticity. Therefore, in contrast to Section 6.1.2, in Figure 5, the operation AddSub() directly encodes S into the least significant bits of T_S instead of performing an XOR, i.e. $T_S[L_T - L_S :] = S$. In step 12, this information can then trivially be recovered. However, it is no longer possible for the SubReceiver to verify message authenticity.

This variant of the subliminal channel is slightly simpler, as it avoids a second encryption request to the Receiver's CKMD. It might also be of relevance for an attacker if the communication between SubReceiver and the SubReceiver's CKMD could be monitored. In this case, it is unlikely that the observer is able to tell apart encryption requests from decryption requests, since the communication to the CKMD itself is very likely to be encrypted. However, since the processing of a message generates two requests to the CKMD in Figure 4 instead of just one, observing the request pattern might already unveil the existence of the subliminal channel.

For the attacker this simplified variant has a few downsides. Beside the fact that T is no longer available to ensure message authenticity, the hidden information S is no longer protected by the encryption key. Hence, when directly encoding transmitted information into S, the information would be openly readable by anyone who is able to intercept the message from the SubSender to the SubReceiver. Furthermore, the distribution of T would deviate from a uniform distribution, disclosing the existence of a subliminal channel on closer analysis. The attacker can counter these drawbacks by performing suitable encryption of S using an attacker-provided key.

Similar to the variant described in Section 6.1.2, we created a proof of concept to verify functionality of this simplified subliminal channel.

7 AN EXEMPLARY INFRASTRUCTURE

To illustrate the relevance of the facilities for hidden communication described in this paper, we now discuss an exemplary infrastructure, where the subliminal channel might pose a severe threat to security. Figure 6 illustrates a practical implementation of the previous generic architecture. The sensor network at the left comprises many IIoT devices like, e.g., charging stations, smart meters, or smart manufacturing devices. These IIoT sensors must authenticate and encrypt the generated data at application layer before sending it to the sensor network, presumably relying on AES-GCM due to its popularity and attractive properties.

Most IIoT devices have limited resources in terms of computational power and memory because of cost considerations. Sensor devices may benefit from using a CKMD in form of a tamper-proof on-board module over storing the cryptographic key in clear for two main reasons: Performance and security. From a performance perspective, the offloading of computationally expensive cryptographic operations to dedicated hardware on the CKMD can reduce overall system sizing and minimize costs by using specialized hardware on the CKMD. From a security perspective, the cryptographic key material must be safely protected against eavesdropping, considering that the sensor devices may be subject to continued physical access by customers or, even worse, unauthorized third parties.

The Data Concentrator in Figure 6 aggregates the data of several IIoT devices on the sensor network. It maintains a security association (for instance TLS or IPsec) to tunnel the data via the public Internet to an Edge Server connected to the corporate network's Demilitarized Zone (DMZ).

The Edge Server's role is to protect the trusted corporate Operating Technology (OT) network against outer access while serving as a security gateway for data that was generated by external sensors. On this behalf, the Edge Server terminates the Data Concentrator's security tunnel and demultiplexes the aggregated data. Subsequently, it verifies and decrypts the data originated by the IIoT devices using the CKMD Server, i.e. one of possibly several dedicated network servers in the corporate OT network for performing cryptographic tasks. Deploying dedicated servers for this task in the corporate OT network brings several benefits:

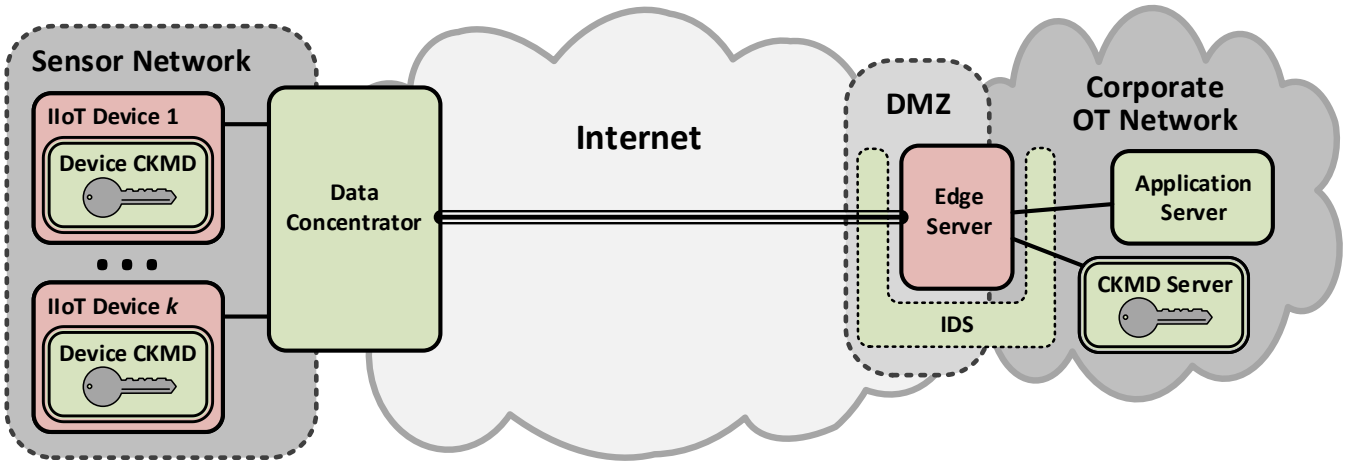

Figure 6: Exploiting the subliminal channel in an IIoT infrastructure.

(1) *Security.* The Edge Server's network interface connected to the corporate DMZ results in an increased attack surface. Storing unique keys for each IIoT device on such an exposed system is neither scalable nor acceptable from a security perspective. It is, therefore, recommended to store the key material on a CKMD, constituting a separate, shielded component.
(2) *Physical key access.* The CKMD Server offers a restricting interface that physically prohibits access to the cryptographic key material. Keys are encapsulated within the CKMD and well-protected against extraction attempts. Moreover, decryption operations are restricted to legitimate operations. In particular, decryption requests are rejected whenever authenticity verification fails. This restriction is an additional hurdle in abusing the CKMD.
(3) *Performance considerations.* Decryption and verification of data entries from hundreds or thousands of IIoT devices at potentially high rate can result in performance bottlenecks. The outsourcing of these operations to the CKMD Server(s) relieves the Edge Server from this effort and supports load balancing if needed.

Following successful decryption, the Edge Server forwards the decrypted IIoT data to the destination Application Server for further processing.

Due to the sensitivity of transmitted data, communication between Edge Server and both CKMD Server and Application Server is end-to-end protected, for instance using TLS transport-layer encryption. Furthermore, an IDS monitors the network traffic in the corporate OT network and in the DMZ, identifying anomalous behavior and raising alarms on suspicious traffic. However, we assume that because of security reasons the IDS neither has access to key material nor can it intercept TLS connections within the corporate OT network.

7.1 Relevance of Subliminal Communication

One main observation concerning Figure 6 is that the presented architecture can effectively protect against network-layer or transport-layer covert channels or subliminal channels. Data concentration and potentially distinct protocols in the sensor network and over the Internet – for instance IPv6 and UDP addressing in the sensor network vs. IPv4 and TLS over TCP in the Internet – can effectively block all attempts for covert communication. Assuming a compromise of IIoT sensors and the Edge Server, the proposed GCM subliminal channel is an ideal candidate for hidden communication between these two. The monitoring IDS in Figure 6 cannot unveil this subliminal channel unless the observed encrypted traffic pattern differs from the benign case.

A second observation concerns the assumption of CKMDs supporting the IV to be a user-provided parameter for encryption: IIoT device manufacturers face tremendous pressure to decrease the costs per unit due to the high production volume. The production volume of CKMDs can be increased (and the cost per module can be reduced) by implementing stateless operation and flexible options to configure cryptographic input parameters. On the IIoT device itself, on the other hand, state-keeping usually is necessary also for accomplishing other tasks. Hence, computation of IVs on the IIoT device does not add cost. Therefore, we argue that our assumption of stateless CKMD operation, allowing applications to provide the IV for GCM encryption, is absolutely legitimate.

As a final note, we remark that a straightforward solution seems to be the implementation of an end-to-end TLS secured connection between IIoT devices and Application Server. However, complex security architectures like the one shown in Figure 6 are a prerequisite to operate the huge base of existing IIoT devices. The available capacity of typical sensor networks, as well as the huge count and limited computational performance of IIoT devices, are commonly prohibitive factors in implementing end-to-end TLS.

8 MITIGATIONS

To avoid the attack scenarios highlighted in this paper, a natural solution is to use a different mode of operation instead. For example, the use of the recent GCM-SIV scheme can be recommended. GCM-SIV retains most attractive properties and, in particular, the performance benefits of GCM, but implements means to alleviate issues arising from IV reuse by choosing the IV for counter mode

encryption depending on the message itself. AES-GCM-SIV [15], which has been standardized in [16], enhances the construction further to obtain improved security bounds. Compared to GCM, GCM-SIV and AES-GCM-SIV have the only drawback of no longer operating in an online fashion, i.e. requiring two passes through the data for encryption instead of just one.

Alternatively, an operational mode that does not deploy counter mode encryption can be used instead like, for example, the traditional choice of Cipher Block Chaining (CBC) [1] in conjunction with a Hash-based Message Authentication Code (HMAC). While the reuse of IVs downgrades security also for modes like GCM-SIV or CBC, security breaches are not as detrimental as for GCM. In particular, it is no longer possible to achieve decryption by issuing an encryption request, ruling out the possibilities for subliminal channels described in this paper.

However, AES-GCM is a very popular cipher. Modes like CBC might be difficult to deploy due to their downsides with respect to encryption speed. If compatibility with existing implementations is required, the use of GCM might be the only option in practical scenarios. Thus, the question arises how to improve security of architectures described in this paper while retaining GCM encryption.

As described above, a universal method for avoiding attack possibilities is letting the CKMD generate an IV for each encrypted message. Since we assume the CKMD to be stateless in most cases, however, IV generation might have to take place randomly, allowing the random IV to be used as a subliminal channel originating from the CKMD. The use of random IVs would, furthermore, incur the drawbacks described in Section 4.2. Since random IV generation requires transmitting the IV values with each message, it might not only require a modification of existing implementations, but also a protocol adaptation.

A further simple method to avoid most attacks described in Section 5 is to ensure that forward and reverse direction use different keys or IVs. Hence, from a given configured key, subkeys can be derived for each direction, respectively. This approach allows to prevent all attacks that are based on performing decryption by issuing an encryption request or vice versa. It does not prevent the "forbidden attack", however, which allows performing authenticating encryption of an arbitrary amount of messages by requesting just two encryptions with the same IV.

If the same encryption key has to be used in both directions for reasons of, e.g., compatibility with existing implementations, a further possibility is to preconfigure certain parts of the IVs on the CKMD. It is sufficient if the CKMDs of the communicating parties require one bit of the IV to be set to respective different values for both directions for the subliminal channel, as described in this paper, to stop working. In fact, NIST recommendation [12] already specifies that a part of the IV identifies the device and, hence, is constant for all invocations on one CKMD. Therefore, this recommendation would only have to be enforced by the CKMD. However, just as using different encryption keys for communication directions, the approach similarly fails to protect against message forgery from two authenticated messages sharing one IV.

9 CONCLUSIONS

Despite being understood as reasonable approaches for achieving security when being used on their own, combining GCM and CKMDs risks to yield a false sense of security. We highlighted attack possibilities that can be used to circumvent restrictions that might be applied by a CKMD in performing encryption or decryption tasks, and showed in particular how these methods can be used to establish a subliminal channel utilizing the authentication tag. Since the CKMD would deny decryption, the establishment of this subliminal channel would not be possible otherwise. For an attacker, the subliminal channel has very attractive properties and allows communicating substantial amounts of hidden information in infrastructures that aim to meet highest demands in terms of security and privacy. Various scenarios allow clandestinely transmitting information as described in this paper. While we focused on GCM, the technique is also applicable to encryption modes that are constructed in a similar way.

We went into detail for an exemplary infrastructure in the field of IIoT and additionally highlighted several approaches for how the threats discussed in this paper might be mitigated. The most potent remedy is to choose a different operational mode like GCM-SIV for encryption. Alternatively, IVs can be chosen directly on the CKMD to avoid attack possibilities, which, however, is likely to enable a subliminal channel originating from the CKMD.

Since the described threats target specifically high-security infrastructures, future development of operational modes and future architectural design of high-security infrastructures should consider possibilities for hidden communication using approaches described in this paper.

ACKNOWLEDGMENTS

This work was supported by the project MALware cOmmunication in cRitical Infrastructures (MALORI), funded by the Austrian security research program KIRAS of the Federal Ministry for Agriculture, Regions and Tourism (BMLRT) under grant no. 873511.

REFERENCES

[1] 1980. FIPS 81, DES Modes of Operation. *Federal Information Processing Standards Publication 81* (1980), 17.
[2] 2001. FIPS 197, Advanced Encryption Standard (AES). *Federal Information Processing Standards Publication 197* (2001), 51.
[3] 2018. IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security. *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)* (2018), 1–239. https://doi.org/10.1109/IEEESTD.2018.8585421
[4] Marcel Armour and Bertram Poettering. 2019. *Subverting Decryption in AEAD.* Technical Report 987. http://eprint.iacr.org/2019/987
[5] Mihir Bellare, Joseph Jaeger, and Daniel Kane. 2015. *Mass-surveillance without the State: Strongly Undetectable Algorithm-Substitution Attacks.* Technical Report 808. http://eprint.iacr.org/2015/808
[6] Mihir Bellare, Kenneth Paterson, and Phillip Rogaway. 2014. *Security of Symmetric Encryption against Mass Surveillance.* Technical Report 438. http://eprint.iacr.org/2014/438
[7] M. Bellare, P. Rogaway, and D. Wagner. 2003. *EAX: A Conventional Authenticated-Encryption Mode.* Technical Report 069. https://eprint.iacr.org/2003/069
[8] Mihir Bellare and Björn Tackmann. 2016. The Multi-user Security of Authenticated Encryption: AES-GCM in TLS 1.3. In *Advances in Cryptology – CRYPTO 2016.* Springer, Berlin, Heidelberg, 247–276. https://doi.org/10.1007/978-3-662-53018-4_10
[9] Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. 2007. A Subliminal-Free Variant of ECDSA. In *Information Hiding.* Springer, Berlin, Heidelberg, 375–387. https://doi.org/10.1007/978-3-540-74124-4_25
[10] Jens-Matthias Bohli and Rainer Steinwandt. 2005. On Subliminal Channels in Deterministic Signature Schemes. In *Information Security and Cryptology – ICISC*

2004. Springer, Berlin, Heidelberg, 182–194. https://doi.org/10.1007/11496618_14

[11] Hanno Böck, Aaron Zauner, Sean Devlin, Juraj Somorovsky, and Philipp Jovanovic. 2016. *Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS*. Technical Report 475. http://eprint.iacr.org/2016/475

[12] Morris Dworkin. 2007. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. Technical Report NIST Special Publication (SP) 800-38D. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-38D

[13] Niels Ferguson. 2005. Authentication weaknesses in GCM. *Comments submitted to NIST Modes of Operation Process* (2005), 1–19.

[14] Shay Gueron and Vlad Krasnov. 2014. The Fragility of AES-GCM Authentication Algorithm. In *2014 11th International Conference on Information Technology: New Generations*. 333–337. https://doi.org/10.1109/ITNG.2014.31

[15] Shay Gueron, Adam Langley, and Yehuda Lindell. 2017. *AES-GCM-SIV: Specification and Analysis*. Technical Report 168. http://eprint.iacr.org/2017/168

[16] S. Gueron, A. Langley, and Y. Lindell. 2019. *AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption*. RFC 8452. RFC Editor. http://www.rfc-editor.org/rfc/rfc8452.txt

[17] Shay Gueron and Yehuda Lindell. 2015. *GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte*. Technical Report 102. http://eprint.iacr.org/2015/102

[18] Helena Handschuh and Bart Preneel. 2008. Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In *Advances in Cryptology – CRYPTO 2008*. Springer, Berlin, Heidelberg, 144–161. https://doi.org/10.1007/978-3-540-85174-5_9

[19] Alexander Hartl, Robert Annessi, and Tanja Zseby. 2017. A Subliminal Channel in EdDSA: Information Leakage with High-Speed Signatures. In *Proceedings of the 2017 International Workshop on Managing Insider Security Threats (MIST '17)*. Association for Computing Machinery, New York, NY, USA, 67–78. https://doi.org/10.1145/3139923.3139925

[20] Viet Tung Hoang, Stefano Tessaro, and Aishwarya Thiruvengadam. 2018. The Multi-user Security of GCM, Revisited: Tight Bounds for Nonce Randomization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Association for Computing Machinery, New York, NY, USA, 1429–1440. https://doi.org/10.1145/3243734.3243816

[21] K. Igoe and J. Solinas. 2009. *AES Galois Counter Mode for the Secure Shell Transport Layer Protocol*. RFC 5647. RFC Editor. http://www.rfc-editor.org/rfc/rfc5647.txt

[22] Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. 2012. Breaking and Repairing GCM Security Proofs. In *Advances in Cryptology – CRYPTO 2012*. Springer, Berlin, Heidelberg, 31–49. https://doi.org/10.1007/978-3-642-32009-5_3

[23] Antoine Joux. 2006. Authentication failures in NIST version of GCM. *Comments submitted to NIST Modes of Operation Process* (2006), 3.

[24] Tadayoshi Kohno, John Viega, and Doug Whiting. 2003. *CWC: A high-performance conventional authenticated encryption mode*. Technical Report 106. https://eprint.iacr.org/2003/106

[25] David McGrew and John Viega. 2004. The Galois/Counter Mode of Operation (GCM). *Submission to NIST Modes of Operation Process* (2004), 44.

[26] David A. McGrew and John Viega. 2004. *The Security and Performance of the Galois/Counter Mode of Operation (Full Version)*. Technical Report 193. http://eprint.iacr.org/2004/193

[27] E. Rescorla. 2018. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. RFC Editor. http://www.rfc-editor.org/rfc/rfc8446.txt

[28] Markku-Juhani O. Saarinen. 2011. *Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes*. Technical Report 202. https://eprint.iacr.org/2011/202

[29] J. Salowey, A. Choudhury, and D. McGrew. 2008. *AES Galois Counter Mode (GCM) Cipher Suites for TLS*. RFC 5288. RFC Editor. http://www.rfc-editor.org/rfc/rfc5288.txt

[30] Bruce Schneier, Matthew Fredrikson, Tadayoshi Kohno, and Thomas Ristenpart. 2015. *Surreptitiously Weakening Cryptographic Systems*. Technical Report 097. http://eprint.iacr.org/2015/097

[31] Gustavus J. Simmons. 1984. The Prisoners' Problem and the Subliminal Channel. In *Advances in Cryptology – CRYPTO'83*. Springer, Boston, MA, 51–67. https://doi.org/10.1007/978-1-4684-4730-9_5

[32] Gustavus J. Simmons. 1984. The Subliminal Channel and Digital Signatures. In *Advances in Cryptology*. Springer Berlin Heidelberg, 364–378. https://doi.org/10.1007/3-540-39757-4_25

[33] Gustavus J. Simmons. 1994. Subliminal Communication is Easy Using the DSA. In *Advances in Cryptology — EUROCRYPT '93*. Springer, Berlin, Heidelberg, 218–232. https://doi.org/10.1007/3-540-48285-7_18

[34] J. Viega and D. McGrew. 2005. *The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)*. RFC 4106. RFC Editor. http://www.rfc-editor.org/rfc/rfc4106.txt

[35] Doug Whiting, Russ Housley, and Niels Ferguson. 2002. *AES Encryption & Authentication Using CTR Mode & CBC-MAC*. Technical Report IEEE 802.11-02/001r0. https://web.cs.ucdavis.edu/~rogaway/ocb/whf02.pdf

[36] Adam Young and Moti Yung. 1996. The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone?. In *Advances in Cryptology — CRYPTO '96*. Springer, Berlin, Heidelberg, 89–103. https://doi.org/10.1007/3-540-68697-5_8

[37] Adam Young and Moti Yung. 1997. Kleptography: Using Cryptography Against Cryptography. In *Advances in Cryptology — EUROCRYPT '97*. Springer, Berlin, Heidelberg, 62–74. https://doi.org/10.1007/3-540-69053-0_6