An extended abstract of this article [AP19a] was accepted for presentation at FSE 2020 and publication in ToSC with DOI 10.13154/tosc.v2019.i3.152-168. This is the full version, and available as entry 2019/989 in the IACR eprint archive.

Substitution Attacks against Message Authentication[*]

Marcel Armour[1] and Bertram Poettering[2]

[1] Royal Holloway, University of London
marcel.armour.2017@rhul.ac.uk
[2] IBM Research Labs Zurich
poe@zurich.ibm.com

Abstract. This work introduces Algorithm Substitution Attacks (ASAs) on message authentication schemes. In light of revelations concerning mass surveillance, ASAs were initially introduced by Bellare, Paterson and Rogaway as a novel attack class against the confidentiality of encryption schemes. Such an attack replaces one or more of the regular scheme algorithms with a subverted version that aims to reveal information to an adversary (engaged in mass surveillance), while remaining undetected by users. While most prior work focused on subverting encryption systems, we study options to subvert symmetric message authentication protocols. In particular we provide powerful generic attacks that apply e.g. to HMAC or Carter–Wegman based schemes, inducing only a negligible implementation overhead. As subverted authentication can act as an enabler for subverted encryption (software updates can be manipulated to include replacements of encryption routines), we consider attacks of the new class highly impactful and dangerous.

Keywords: Algorithm Substitution Attacks · Authentication · Mass Surveillance

1 Introduction

Message authentication schemes are designed to provide a guarantee of integrity: that is, the assurance that a message really was sent by its purported sender. Symmetrically keyed message authentication is a well-studied problem, and there exist many reliable and provably secure solutions (the most popular likely being HMAC). These solutions rely on the assumption that the software or hardware in which they are implemented behaves as expected. However, we know that in the real world this assumption does not necessarily hold. Powerful adversaries have the means to insert unreliability into cryptography via external ("real-world") infrastructure: whether by influencing standards bodies to adopt "backdoored" parameters, inserting exploitable errors into software implementations, or compromising supply chains to interfere with hardware. The Snowden revelations showed that this is indeed the case, and that large and powerful adversaries (interested in mass surveillance) have sought to circumvent cryptography. The reader is referred to the survey by Schneier et al. [SFKR15], which provides a broad overview of subversion of cryptography, with some useful case studies detailing known subversion attempts.

The idea that an adversary may embed a backdoor or otherwise tamper with the implementation or specification of a cryptographic scheme or primitive predates the Snowden revelations, and was initiated in a line of work by Young and Yung that they named *kleptography* [YY96, YY97]. This area of study can be traced back to Simmons' work on *subliminal channels*, e.g. [Sim83], undertaken in the context of nuclear non-proliferation during the Cold War. In the original conception [YY96], kleptography considered a saboteur who designs a cryptographic algorithm whose outputs are computationally indistinguishable from the outputs of an unmodified trusted algorithm. The saboteur's algorithm should leak private key data through the output of the system, which was achieved using the same principles as Simmons' earlier subliminal channels. Post-Snowden, work in this area was reignited by Bellare, Paterson and Rogaway

[*]The research of Armour was supported by the EPSRC and the UK government as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London (EP/P009301/1). The research of Poettering was supported by the European Union's Horizon 2020 project FutureTPM (779391).

(BPR) [BPR14], who formalised study of so-called *algorithm substitution attacks* (ASAs) through the specific example of symmetric encryption schemes. In abstract terms, the adversary's goal in an ASA is to create a subverted implementation of a scheme that breaks some aspect of security (such as IND-CPA in the case of encryption) while remaining undetected by the user.

1.1 Our Work

In this work, we examine the consequences of ASAs to message authentication. In abstract terms, the adversary's goal in an ASA is to create a subverted implementation of a scheme that breaks some aspect of security while remaining undetected by the user(s). In essence, a message authentication scheme provides a message authentication code or *tag* for a given message. Furthermore, given a message and a tag, the message authentication scheme provides verification that the tag was generated from the message (that is, that the tag is genuine). The security of a message authentication scheme is determined by the difficulty of forging tags. If no adversary can forge a tag, then a message with a correct tag must have been generated by the sender. An ASA against a message authentication scheme replaces either the tagging function (the generation of message authentication codes) or the verification function (checking that tags have been honestly generated) in such a way as to be able to leak information to an adversary. We provide formal definitions for subversion attacks against message authentication schemes whose syntax allows for both sender and receiver subversion. Previous work in this area considered only subversion of the sender; our main contribution is to show that this assumption misses an important class of attack that targets the receiver. We believe that this class of Algorithm Substitution Attack, or at least the insight that the receiver rather than the sender may be targeted, can be applied to other cryptographic primitives.

Our attack is designed to leak the secret key, as this is the most devastating attack from the point of view of an adversary. Once the secret key is known, any tag can be forged. Although considering a message authentication scheme in isolation is a useful perspective from which to analyse, drawing borders in such a way hides the complexity of the real world, in particular the ways in which various (cryptographic) schemes or services are used together. Once integrity has been compromised, an adversary can often leverage this to perform any number of other attacks, for example: enabling attacks against ("encrypt-then-MAC") confidentiality; getting users to accept compromised (authenticated) software updates; injecting malicious packets into (secured) communication streams to de-anonymise users. Whilst we consider key recovery as the ultimate goal of an adversary, a more general attack could hide arbitrary information (for example the user's encryption key for their messaging application, or the internal state of its random number generator). We note that allowing arbitrary information to be exfiltrated rather than user keys is a straight-forward extension of our definitions. We consider such a general attack to be less realistic, as an algorithm that interacts with other applications (such as accessing secret keys stored on disk) would risk detection.

1.2 Preceding work

BPR [BPR14] demonstrate an attack against certain randomised encryption schemes that relies on influencing the randomness generated in the course of encryption. Their attack applies to the sub-class of schemes satisfying a property they call 'coin-injectivity'. Their result was later generalised by Bellare, Jaeger and Kane (BJK) [BJK15] whose attack applies to all randomised schemes. Furthermore, whereas the attack of BPR is stateful and so vulnerable to detection through state reset, the BJK attack is stateless.

There is a tension for 'Big Brother' between mounting a successful attack and being detected; clearly an attack that simply replaces the encryption algorithm with one that outputs the messages in plaintext would be devastating yet trivially detectable. BPR stipulate that subverted schemes should at the very least decrypt correctly (according to the unmodified specification) in order to have some measure of resistance to detection, going on to define the success probability of a mass surveillance adversary in carrying out a successful attack, as well as the advantage of a user in detecting that a surveillance attack is taking place. BJK [BJK15] later formalised the goal of key recovery as the desired outcome of an ASA from the point of view of a mass surveillance adversary. Lastly, BPR also establish a positive result that shows that under certain assumptions, it is possible for authenticated encryption schemes to provide resistance against subversion attacks.

Degabriele, Farshim and Poettering (DFP) [DFP15] critiqued the definitions and underlying assumptions of BPR. Their main insight is that the perfect decryptability —a condition mandated by BPR— is a very strong requirement and artificially limits the adversary's set of available strategies. In practice, a subversion with negligible detection probability, say 2^{-128}, should be considered undetectable.[1] As DFP note, decryption failures may happen for reasons other than a subverted encryption algorithm, and if they occur sporadically may easily go unnoticed. Thus a subverted encryption scheme that exhibits decryption failure with a very low probability is a good candidate for a practical ASA that is hard to detect. DFP demonstrate how this can be achieved with an input-triggered subversion, where the trigger is some message input that is difficult to guess, making detection practically impossible.

None of the above-mentioned works considers the cryptographic subversion of MAC schemes. The only prior work on that topic that we are aware of is by Al Mansoori, Baek, and Salah [ABS16] who explore how a MAC component in the EAP-PSK wireless protocol could be subverted. While this might sound precisely like what our article is offering as well, there is actually surprisingly little overlap between their results and ours: In their article, after first arguing [ABS16, §II.D] that randomised MAC schemes offer better protection against a kind of birthday attack, they restrict attention to precisely one corresponding construction (two-key CBC-MAC with a random translation of the second key, a scheme that already turned out to be broken in [KK03]) and show that the rejection-sampling based key-extraction techniques from [BPR14] are applicable in this setting as well. We emphasise that our results reach far beyond this: our subversion attacks are generic (rather than being focused on one specific MAC) and we don't require exotic technical conditions like randomised tag generation.[2] We believe that the increased generality that our techniques offer make them more suitable to backdoor EAP-PSK implementations in practice.

Related Work. We outlined the key antecedent publications on ASAs against symmetric encryption schemes above. Other works, briefly described here, consider subversion on different primitives and in different contexts. Berndt and Liskiewicz [BL17] reunite the fields of cryptography and steganography. Ateniese, Magri and Venturi [AMV15] study ASAs on signature schemes. In a series of work, Russell, Tang, Yung and Zhou [RTYZ16a, RTYZ16b, RTYZ17, RTYZ18] consider ASAs on one-way functions, trapdoor one-way functions and key generation as well as defending randomised algorithm against ASAs. Goh, Boneh, Pinkas and Golle [GBPG03] show how to add key recovery to the SSL/TLS and SSH protocols. Dodis, Ganesh, Golovnev, Juels and Ristenpart [DGG+15] provide a formal treatment of backdooring PRGs, another form of subversion. Camenisch, Drijvers and Lehmann [CDL17] consider Direct Anonymous Attestation in the presence of a subverted Trusted Platform Module. Cryptographic reverse firewalls [MS15, DMS16, MZY+18] represent an architecture to counter ASAs via trusted code in network perimeter filters. Fischlin and Mazaheri show how to construct ASA-resistant encryption and signature algorithms given initial access to a trusted base scheme [FM18]. Fischlin, Janson and Mazaheri [FJM18] show how to immunise hash functions against subversion. Bellare, Kane and Rogaway [BKR16] explore using large keys to prevent key exfiltration in the symmetric encryption setting. Bellare and Hoang [BH15] give public key encryption schemes that defend against the subversion of random number generators.

In concurrent work, we study the effects of subverting decryption in the setting of AEAD schemes (authenticated encryption with associated data) [AP19b]. Using similar techniques, we provide ASAs that result in successful key exfiltration, ultimately leading to a full breach of confidentiality.

Cryptographic vs. non-cryptographic subversion. In the literature on cryptography, the notion of an ASA (against an authentication scheme or otherwise) assumes the malicious replacement of one or more algorithms of a scheme by a backdoored version, with the goal to leak key material or at least to weaken some crucial security property. Different types of substitution attack appear in other areas of computing and communication. We discuss some examples in the following.

Program code in the domain of computer malware routinely modifies system functions to achieve its goals, where the latter comprises delivering some damaging payload, ensuring non-detection and thus survival of the malware on the host system, and in some cases even self-reproduction. Uncountable techniques towards suitably modifying a host system have been developed and reported on by academic researchers and hackers. Standard examples include redirecting interrupt handlers, changing the program entry point of an executable file, and interfering with the OS kernel by overwriting its data structures [Gol11].

[1]This is analogous to the fundamental notion in cryptography that a symmetric encryption scheme be considered secure even in the presence of adversaries with negligible advantage.

[2]We are not aware of *any* randomised MAC of practical relevance.

Malicious modifications of implemented functionality are also a recognised threat in the hardware world. It is widely understood that circuit designers who do not possess the technical means to produce their own chips but instead out-source the production process to external foundries, risk that the chips produced might actually implement a maliciously modified version of what is expected. A vast number of independent options are known for when (within the production cycle) and how (functionally) subversions could be conducted. For instance, the survey provided in [BHBN14] reports that circuit design software (CAD) could be maliciously altered, that foundries could modify circuits before production, and that after production commercial suppliers could replace legitimate chips by modified ones. Further, [BHBN14] suggests that appealing types of functionality modification include deviating from specification when particular input trigger events are recognised, and/or to leak values of vital internal registers via explicitly implemented side channels. Any such technique (or combination thereof) has an individual profile regarding the involved costs and detectability of attack. Which of the many options is the most preferable depends on the specific attack scenario and target.

We refer to the software and hardware based subversion techniques discussed above as "technology driven". This is in contrast to the techniques considered in this paper which we refer to as "semantics driven". We consider the two approaches orthogonal: Our (semantics driven) subversion proposals *can* be implemented using the techniques from e.g. [Gol11, BHBN14] (but likewise also through standard methods), and technology driven subversion proposals *can* be applied against cryptographic implementations (but likewise also against any other interesting target functionality). Our semantics driven approach in fact aims to maximise technology independence. As a consequence, the line of attacks proposed in this paper can be implemented easily in software (e.g. in libraries or drop-in code), in hardware (e.g. in ASICs and FPGAs), and in mixed forms (e.g. firmware-programmed microcontrollers). The strategy to achieve this independence is to base the attacks and corresponding notions of (in)security on nothing but the abstract functionalities of the attacked scheme (in our case: authentication systems) as they are determined by their definitions of syntax and correctness.

As the technology driven and semantics driven approaches are independent, they can in particular be combined. This promises particularly powerful subversions. For instance, consider that virtually all laptops and desktop PCs produced in the past decade are required to have an embedded trusted platform module (TPM) chip that supports software components (typically boot loaders and operating systems) with *trusted* cryptographic services which amongst others includes symmetric message authentication. In detail, software can interact with a TPM chip through standardised API function calls and have cryptographic operations applied to provided inputs, with all key material being generated and held exclusively in the TPM. As TPMs are manufactured in hardware, it seems that the (technology driven) subversion options proposed in [BHBN14] would be particularly suitable. However, as most of the attacks from [BHBN14] require physical presence of the adversary (e.g., to provide input triggers via specific supply voltage jitters or for extracting side channel information by operating physical probes in proximity of the attacked chip), only those options seem feasible where all attack conditions and events can be controlled and measured via the software interface provided by the API. This is precisely what our semantics driven attacks provide. We thus conclude by observing that dedicated cryptographic hardware like TPMs can only be trusted if *extreme* care is taken during design and production. While our article lays open the most general and clean line of attack, other attacks might exist as well.

2 Preliminaries

2.1 Notation

We refer to an element $x \in \{0,1\}^*$ as a string, and denote its length by $|x|$. The set of strings of length l is denoted $\{0,1\}^l$. For $x \in \{0,1\}^*$ we let $x[i]$ denote the i-th bit of x, with the convention that we count from 0, i.e., we have $x = x[0] \ldots x[|x|-1]$. For a bit value $b \in \{0,1\}$ we write $!\,b$ for its inverse, i.e., the value $1-b$. We use Iverson brackets $[\cdot]$ to derive bit values from Boolean conditions: For a condition C we have $[C] = 1$ if C holds; otherwise we have $[C] = 0$.

We use code-based notation for probability and security experiments. We write $\leftarrow$ for the assignment operator (that assigns a right-hand-side value to a left-hand-side variable). If S, S' are sets, we write $S \overset{\cup}{\leftarrow} S'$ shorthand for $S \leftarrow S \cup S'$. If S is a finite set, then $s \leftarrow_{\$} S$ denotes choosing s uniformly at random from S. We denote a α-biased Bernoulli trial by $B(\alpha)$, i.e., a random experiment with possible

```
Exp CC($S,\alpha$)
$S' \leftarrow \emptyset$; $l \leftarrow 0$
While $S' \subsetneq S$:
    $s \leftarrow_\$ S$
    If $B(\alpha)$:
        $S' \overset{\cup}{\leftarrow} \{s\}$
    $l \leftarrow l + 1$
Stop with $l$
```

Figure 1: Coupon collector experiment (see Lemma 1). See Section 2.1 for the meaning of "$B(\cdot)$".

outcomes 0 or 1 such that $\Pr[b \leftarrow B(\alpha) : b = 1] = \alpha$. The assignments $b \leftarrow_\$ \{0,1\}$ and $b \leftarrow B(1/2)$ are thus equivalent. We use superscript notation to indicate when an algorithm (typically an adversary) is given access to specific oracles. An experiment terminates with a "Stop with x" instruction, where value x is understood as the outcome of the experiment. We write "Win" ("Lose") as shorthand for "Stop with 1" ("Stop with 0"). We write "Require C", for a Boolean condition C, shorthand for "If not C: Lose". (We use Require clauses typically to abort a game when the adversary performs some disallowed action, e.g. one that would lead to a trivial win.) The ":=" operator creates a symbolic definition; for instance, the code line "$A := E$" does *not* assign the value of expression E to variable A but instead introduces symbol A as a new (in most cases abbreviating) name for E.

2.2 Combinatorics: Coupon Collection

In the course of this article we expose subversion attacks of various types. Although we used different design principles when constructing them, the attacks share as a common property that they exfiltrate secret key material one bit at a time. The following lemma recalls a standard coupon collector statement that will help analysing the efficiency of this approach, in particular how long it takes until all bits are extracted. For a proof of the lemma, see e.g. [Isa13, §8.4], or any introductory text to probability.

Lemma 1. *Fix a finite set S (of "coupons") and a probability $0 \le \alpha \le 1$. Experiment $\mathrm{CC}(S,\alpha)$ in Fig. 1 measures the number of iterations it takes to visit all elements of S ("collect all coupons") when picked uniformly at random and considered with probability α. The expected number of iterations is given by $O(n \log n)$, where $n = |S|$. More precisely we have*

$$\mathbb{E}[\mathrm{CC}(S,\alpha)] = \frac{|S|}{\alpha}\left(\frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{|S|}\right) = O(n \log n).$$

Note that parameter α is fully absorbed by the $O(\cdot)$ notation.

2.3 Message Authentication Schemes

Cryptographic message authentication is typically achieved with a message authentication code (MAC). Given a key k and a message m, a tag t is deterministically derived as per $t \leftarrow \mathsf{tag}(k,m)$. The (textbook) method to verify the authenticity of m given t is to recompute $t' \leftarrow \mathsf{tag}(k,m)$ and to consider m authentic iff $t' = t$. If this final tag comparison is implemented carelessly, a security issue might emerge: A natural yet naive way to perform the comparison is to check the tag bytes individually in left-to-right order until either a mismatch is spotted or the right-most bytes have successfully been found to match. Note that, if tags are not matching, such an implementation might easily give away, as timing side-channel information, the length of the matching prefix, allowing for practical forgery attacks via step-wise guessing.

This issue is understood by the authors of major cryptographic libraries, which thus contain carefully designed constant-time string comparison code. A consequence is that services for tag generation and verification are routinely split into two separate functions tag and vfy.[3] Our notion of a message authentication scheme, formalised next, follows this approach. It comprises MAC based authentication as a special case, but it also comprises the more exotic randomised MACs as considered in [ABS16].

[3]See `https://nacl.cr.yp.to/auth.html` for an example.

```
**Game** UF($\mathcal{A}$)
00 $k \leftarrow_\$ \mathcal{K}$
01 $C \leftarrow \emptyset$
02 $\mathcal{A}^{\text{Tag},\text{Vfy}}$
03 Lose

**Oracle** Tag($m$)
04 $t \leftarrow \mathsf{tag}(k, m)$
05 $C \overset{\cup}{\leftarrow} \{(m, t)\}$
06 Return $t$

**Oracle** Vfy($m, t$)
07 $v \leftarrow \mathsf{vfy}(k, m, t)$
08 If $v \wedge (m, t) \notin C$:
09     Win
10 Return $v$
```

```
**Game** subUF($\mathcal{A}$)
11 $k \leftarrow_\$ \mathcal{K}$; $i \leftarrow_\$ \mathcal{I}$; $j \leftarrow_\$ \mathcal{J}$
12 $C \leftarrow \emptyset$
13 $\mathcal{A}^{\text{Tag},\text{Vfy}}$
14 Lose

**Oracle** Tag($m$)
15 $t \leftarrow \mathsf{tag}_i(k, m)$
16 $C \overset{\cup}{\leftarrow} \{(m, t)\}$
17 Return $t$

**Oracle** Vfy($m, t$)
18 $v \leftarrow \mathsf{vfy}_j(k, m, t)$
19 If $v \wedge (m, t) \notin C$:
20     Win
21 Return $v$
```

Figure 2: Games modelling the unforgeability (UF) and subverted unforgeability (subUF) of a message authentication scheme.

Formally, a scheme Π providing message authentication consists of algorithms $\mathsf{tag}, \mathsf{vfy}$ and associated spaces $\mathcal{K}, \mathcal{M}, \mathcal{T}$. The tagging algorithm tag takes a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$, and returns a tag $t \in \mathcal{T}$. The verification algorithm vfy takes a key $k \in \mathcal{K}$, a message $m \in \mathcal{M}$, and a tag $t \in \mathcal{T}$, and returns a bit value v. If the bit is set we say the algorithm accepts; otherwise we say it rejects. A shortcut notation for this syntax is

$$\mathcal{K} \times \mathcal{M} \to \mathsf{tag} \to \mathcal{T} \qquad \text{and} \qquad \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \mathsf{vfy} \to \{0, 1\} \ .$$

For correctness we require that for all $k \in \mathcal{K}$, $m \in \mathcal{M}$, $t \in \mathcal{T}$ we have that $\mathsf{tag}(k, m) = t$ implies $\mathsf{vfy}(k, m, t) = 1$. We formalise the (strong) unforgeability of a message authentication scheme via the game UF in Fig. 2 (left). For any adversary $\mathcal{A}$ we define the advantage $\mathbf{Adv}^{\text{uf}}(\mathcal{A}) := \Pr[\text{UF}(\mathcal{A})]$ and say that scheme Π is (strongly) unforgeable if $\mathbf{Adv}^{\text{uf}}(\mathcal{A})$ is negligibly small for all realistic $\mathcal{A}$.

3 Notions of Subversion Attacks on Message Authentication

We consider subversions of the algorithms of message authentication schemes. Depending on whether the tagging algorithm, the verification algorithm, or even both are subverted, three attack classes could in principle be distinguished. However, as tagging and verification are typically performed by distinct, remote parties, successfully conducting attacks of the third kind would require unnoticedly replacing implementations of *two* participants, which we think is considerably less feasible than replacing only one implementation. In this article we thus focus on single-algorithm substitution attacks. If only the tagging algorithm is subverted we say the subversion is *verification-intact*, while if only the verification algorithm is subverted we say the subversion is *tagging-intact*.

In the following we give formal definitions for the two types of subversion. In each case we also formalise a corresponding notion of *undetectability* (UD). In a nutshell, a subversion is undetectable if distinguishers with black-box access to either the original scheme or to its subverted variant cannot tell the two apart. We note that in our models the distinguishers have full control over the keys to be used. This is in contrast to prior work like [BPR14, DFP15] where undetectability is defined with respect to uniform keys. As code auditors and other security researchers looking for subversion attacks can specify keys during black-box testing according to their preferred distribution, we consider uniform-key constraints a rather severe limitation of undetectability notions. Note further that our security models assume purely black-box access to the subverted functions, i.e., that distinguishing adversaries don't have access to the code or circuitry of the subversion but only to its modified functionality. Detection methods that look out for implementation artefacts like execution time and memory consumption (for software subversions), or changed circuit layouts and doping patterns (for hardware subversions), are thus *not* excluded by our definitions.

```
Game $\mathrm{UDT}^b(\mathcal{A})$
00 $i \leftarrow_\$ \mathcal{I}$
01 $\mathsf{tag}^0 := \mathsf{tag}_i$
02 $\mathsf{tag}^1 := \mathsf{tag}$
03 $b' \leftarrow \mathcal{A}^{\mathrm{Tag},\mathrm{Vfy}}$
04 Stop with $b'$

Oracle $\mathrm{Tag}(k, m)$
05 $t \leftarrow \mathsf{tag}^b(k, m)$
06 Return $t$

Oracle $\mathrm{Vfy}(k, m, t)$
07 $v \leftarrow \mathsf{vfy}(k, m, t)$
08 Return $v$
```

```
Game $\mathrm{UDV}^b(\mathcal{A})$
00 $j \leftarrow_\$ \mathcal{J}$
01 $\mathsf{vfy}^0 := \mathsf{vfy}_j$
02 $\mathsf{vfy}^1 := \mathsf{vfy}$
03 $b' \leftarrow \mathcal{A}^{\mathrm{Tag},\mathrm{Vfy}}$
04 Stop with $b'$

Oracle $\mathrm{Tag}(k, m)$
05 $t \leftarrow \mathsf{tag}(k, m)$
06 Return $t$

Oracle $\mathrm{Vfy}(k, m, t)$
07 $v \leftarrow \mathsf{vfy}^b(k, m, t)$
08 Return $v$
```

```
Game $\mathrm{UD}^b(\mathcal{A})$
00 $i \leftarrow_\$ \mathcal{I};\ j \leftarrow_\$ \mathcal{J}$
01 $(\mathsf{tag}^0, \mathsf{vfy}^0) := (\mathsf{tag}_i, \mathsf{vfy}_j)$
02 $(\mathsf{tag}^1, \mathsf{vfy}^1) := (\mathsf{tag}, \mathsf{vfy})$
03 $b' \leftarrow \mathcal{A}^{\mathrm{Tag},\mathrm{Vfy}}$
04 Stop with $b'$

Oracle $\mathrm{Tag}(k, m)$
05 $t \leftarrow \mathsf{tag}^b(k, m)$
06 Return $t$

Oracle $\mathrm{Vfy}(k, m, t)$
07 $v \leftarrow \mathsf{vfy}^b(k, m, t)$
08 Return $v$
```

Figure 3: Games UDT and UDV and UD modelling subversion undetectability. See Section 2.1 for the meaning of ":=". Note that the Vfy oracle in UDT and the Tag oracle in UDV are redundant.

A subversion should exhibit a dedicated functionality for the subverting party, but simultaneously be undetectable for all others. This seeming contradiction is resolved by parameterising the subverted algorithm with a secret subversion key, knowledge of which enables the extra functionality. (The same technique is used in most prior work, starting with [BPR14].) In what follows we denote the corresponding subversion key spaces with $\mathcal{I}$ and $\mathcal{J}$.

In this section we also specify, by introducing notions of key recoverability, how we measure the quality of a subversion from the point of view of the subverting adversary (who is assumed to know the subversion keys).

3.1 Undetectable Subversion

Subverted Tagging. A subversion of the tagging algorithm of a message authentication scheme consists of a finite index space $\mathcal{I}$ and a family $\mathcal{T}ag = \{\mathsf{tag}_i\}_{i \in \mathcal{I}}$ of algorithms

$$\mathcal{K} \times \mathcal{M} \to \mathsf{tag}_i \to \mathcal{T} \ .$$

That is, for all $i \in \mathcal{I}$ the algorithm tag_i can syntactically replace the algorithm tag. As a security property we require that also the observable behaviour of tag and tag_i be effectively identical (for uniformly chosen $i \in \mathcal{I}$). This is formalised via the games $\mathrm{UDT}^0, \mathrm{UDT}^1$ in Fig. 3 (left). For any adversary $\mathcal{A}$ we define the advantage $\mathbf{Adv}^{\mathrm{udt}}(\mathcal{A}) := |\Pr[\mathrm{UDT}^1(\mathcal{A})] - \Pr[\mathrm{UDT}^0(\mathcal{A})]|$ and say that family $\mathcal{T}ag$ undetectably subverts algorithm tag if $\mathbf{Adv}^{\mathrm{udt}}(\mathcal{A})$ is negligibly small for all realistic $\mathcal{A}$.

Subverted Verification. A subversion of the verification algorithm of a message authentication scheme consists of a finite index space $\mathcal{J}$ and a family $\mathcal{V}fy = \{\mathsf{vfy}_j\}_{j \in \mathcal{J}}$ of algorithms

$$\mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \mathsf{vfy}_j \to \{0, 1\} \ .$$

That is, for all $j \in \mathcal{J}$ the algorithm vfy_j can syntactically replace the algorithm vfy. As a security property we require that also the observable behaviour of vfy and vfy_j be effectively identical (for uniformly chosen $j \in \mathcal{J}$). This is formalised via the games $\mathrm{UDV}^0, \mathrm{UDV}^1$ in Fig. 3 (centre). For any adversary $\mathcal{A}$ we define the advantage $\mathbf{Adv}^{\mathrm{udv}}(\mathcal{A}) := |\Pr[\mathrm{UDV}^1(\mathcal{A})] - \Pr[\mathrm{UDV}^0(\mathcal{A})]|$ and say that family $\mathcal{V}fy$ undetectably subverts algorithm vfy if $\mathbf{Adv}^{\mathrm{udv}}(\mathcal{A})$ is negligibly small for all realistic $\mathcal{A}$.

While we argued above that cases where the tagging and the verification algorithms are simultaneously subverted are less realistic, for completeness we also give a joint definition of undetectability.

Subversion of Tagging and Verification. Games UD^b in Fig. 3 (right) combine games UDT^b and UDV^b into one. We define $\mathbf{Adv}^{\mathrm{ud}}(\mathcal{A}) := |\Pr[\mathrm{UD}^1(\mathcal{A})] - \Pr[\mathrm{UD}^0(\mathcal{A})]|$. By a hybrid argument, for all adversaries $\mathcal{A}$ there exist adversaries $\mathcal{A}', \mathcal{A}''$ such that $\mathbf{Adv}^{\mathrm{ud}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{udt}}(\mathcal{A}') + \mathbf{Adv}^{\mathrm{udv}}(\mathcal{A}'')$.

The above undetectability notions demand that subversions do not change the observable behaviour of tagging and verification algorithms. A consequence of this is that neither the correctness nor the security properties of the affected authentication scheme are considerably harmed (assuming random and unknown subversions keys).

Effects of Subversion on Correctness and Unforgeability. Undetectably subverting the tagging and/or verification algorithm of a message authentication scheme has only a limited negative effect on the resulting scheme's correctness: For all k, m, t and uniformly chosen i, j the probability that $\mathsf{tag}_i(k, m) = t$ does not imply $\mathsf{vfy}_j(k, m, t) = 1$ is bounded by $\mathbf{Adv}^{\mathrm{ud}}(\mathcal{A})$ for an adversary $\mathcal{A}$. Similarly, if we consider the subverted unforgeability game subUF from Fig. 2 (right) and define for any adversary $\mathcal{A}$ the advantage $\mathbf{Adv}^{\mathrm{subuf}}(\mathcal{A}) := \Pr[\mathrm{subUF}(\mathcal{A})]$, by a hybrid argument we obtain that for all adversaries $\mathcal{A}$ there exist adversaries $\mathcal{A}', \mathcal{A}''$ such that $\mathbf{Adv}^{\mathrm{subuf}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{uf}}(\mathcal{A}') + \mathbf{Adv}^{\mathrm{ud}}(\mathcal{A}'')$.

3.2 Subversion Leading to Key Recovery

We observed above that if the tagging and/or verification algorithm of an authentication scheme is undetectably subverted, with uniformly chosen indices i, j that remain unknown to all participants, then the unforgeability guarantees are preserved from the original scheme. This may be different if i, j are known to an attacking party, and indeed we assume that mass-surveillance attackers leverage such knowledge to conduct forgery attacks. An attack goal that represents a particularly attractive option to enable message forgery is key recovery (KR): Users pick symmetric keys k according to their preferred method (hidden from the adversary), and the adversary aims at recovering these keys through the subversion.

We formalise this attack goal in two versions. The KRP game in Fig. 4 (left) assumes a passive attack in which the adversary cannot manipulate messages or tags, and the KRA game in Fig. 4 (right) assumes an active attack in which the adversary can inject and test arbitrary message-tag pairs. In both cases, with the aim of closely modelling real-world settings, we (slightly) restrict the adversary's influence on the messages that are tagged by assuming a stateful message sampler MS that produces the messages used throughout the game. The syntax of this message sampler is

$$\Sigma \times A \to \mathrm{MS} \to \Sigma \times \mathcal{M} \times B \qquad\qquad (\sigma, \alpha) \mapsto \mathrm{MS}(\sigma, \alpha) = (\sigma', m, \beta)\ ,$$

where $\sigma, \sigma' \in \Sigma$ are old and updated state, input $\alpha \in A$ models the influence that the adversary may have on message generation, and output $\beta \in B$ models side-channel outputs. In Fig. 4 we write $\diamond$ for the initial state. Note that while we formalise the inputs α and the outputs β for generality (so that our models cover most real-world applications), our subversion attacks are independent of them.[4]

For any message sampler MS and adversary $\mathcal{A}$ we define the advantages $\mathbf{Adv}^{\mathrm{krp}}_{\mathrm{MS}}(\mathcal{A}) := \Pr[\mathrm{KRP}(\mathcal{A})]$ and $\mathbf{Adv}^{\mathrm{kra}}_{\mathrm{MS}}(\mathcal{A}) := \Pr[\mathrm{KRA}(\mathcal{A})]$. We say that subversion families $\mathcal{Tag}, \mathcal{Vfy}$ are key recovering for passive attackers if for all practical MS there exists a realistic adversary $\mathcal{A}$ such that $\mathbf{Adv}^{\mathrm{krp}}_{\mathrm{MS}}(\mathcal{A})$ reaches a considerable value (e.g., 0.1). The key recovery notion for active attackers is analogue.

4 Concrete Subversion Attacks via Acceptance vs. Rejection

We propose two key-recovering subversion attacks on message authentication schemes. While both attacks are tagging-intact, i.e., subvert only the verification routine, they differ in that our first attack is passive (can be mounted by an exclusively observing mass surveillance adversary) and our second attack is active (requires testing specific message-tag pairs for validity). The driving principles behind the two attacks are closely related: In both cases the verification algorithm of the attacked scheme is manipulated such that it marginally deviates from the regular accept/reject behaviour; by making these deviations depend on the authentication key, the bits of the latter are leaked one by one.

Our passive attack rejects a sparse subset of the message-tag pairs that the unmodified algorithm would accept. Our active attack does the opposite by accepting certain specially-crafted inauthentic message-tag pairs as valid. A property of the former attack is that the scheme's correctness is slightly reduced; we believe however that in settings where rejected messages are automatically retransmitted

[4]... meaning that the reader can safely ignore them.

```
Game KRP($\mathcal{A}$)
00 $C \leftarrow \emptyset$
01 $k \leftarrow_\$ \mathcal{K}$; $\sigma \leftarrow \diamond$
02 $i \leftarrow_\$ \mathcal{I}$; $j \leftarrow_\$ \mathcal{J}$
03 $k' \leftarrow \mathcal{A}^{\mathrm{Tag},\mathrm{Vfy}}(i, j)$
04 Stop with $[k' = k]$

Oracle Tag($\alpha$)
05 $(\sigma, m, \beta) \leftarrow \mathrm{MS}(\sigma, \alpha)$
06 $t \leftarrow \mathsf{tag}_i(k, m)$
07 $C \overset{\cup}{\leftarrow} \{(m, t)\}$
08 Return $(m, t, \beta)$

Oracle Vfy($m, t$)
09 Require $(m, t) \in C$
10 $v \leftarrow \mathsf{vfy}_j(k, m, t)$
11 Return $v$
```

```
Game KRA($\mathcal{A}$)
12 ~~$C \leftarrow \emptyset$~~
13 $k \leftarrow_\$ \mathcal{K}$; $\sigma \leftarrow \diamond$
14 $i \leftarrow_\$ \mathcal{I}$; $j \leftarrow_\$ \mathcal{J}$
15 $k' \leftarrow \mathcal{A}^{\mathrm{Tag},\mathrm{Vfy}}(i, j)$
16 Stop with $[k' = k]$

Oracle Tag($\alpha$)
17 $(\sigma, m, \beta) \leftarrow \mathrm{MS}(\sigma, \alpha)$
18 $t \leftarrow \mathsf{tag}_i(k, m)$
19 ~~$C \overset{\cup}{\leftarrow} \{(m, t)\}$~~
20 Return $(m, t, \beta)$

Oracle Vfy($m, t$)
21 ~~Require $(m, t) \in C$~~
22 $v \leftarrow \mathsf{vfy}_j(k, m, t)$
23 Return $v$
```

Figure 4: Games KRP and KRA modelling key recoverability for passive and active attackers, respectively.

by the sender (e.g. in low-level network encryption like IPSec), this attack is still very practical and impactful. Our active attack does not influence correctness. However, as key bits are leaked only when the verification algorithm is exposed to inauthentic tags, successful adversaries are necessarily active. The active attack furthermore has the following attractive property: The messages corresponding to the injected inauthentic tags are not arbitrary (and thus unexpected to the processing application), but identical with messages previously seen by the tagging algorithm. This allows keeping attacks 'under the radar': The adversary can suppress the (correct) tags of real messages and replace them with crafted tags; the receiver will not realise, as all accepted messages they receive will be those sent by the sender.

While our attacks are very effective in practice, we note that most prior academic formulations of ASAs, like [BPR14, DFP15] (in the domain of symmetric encryption) or [ABS16] (in the domain of message authentication), do not consider subversion on the receiver side. By consequence, by the notions proposed and used in these articles, our attacks are *trivially* absolutely undetectable. Our notions from Section 3, which cover also the receiving end, allow for a more realistic assessment of the situation. According to our notions, our passive attack could be detected with at least a noticeable probability, while our active attack remains undetectable.

In this section we consistently refer to the algorithms and spaces of a targeted message authentication scheme with $\mathsf{tag}, \mathsf{vfy}, \mathcal{K}, \mathcal{M}, \mathcal{T}$. We further assume that the key space has the form $\mathcal{K} = \{0,1\}^\kappa$ for some $\kappa \in \mathbb{N}$.

4.1 Passive Attack

We define our passive subversion of the verification algorithm of an authentication scheme in Fig. 5 (left). It is parameterised by a probability $0 \le \gamma \le 1$, a large index space $\mathcal{J}$, a PRF $(F_j)_{j \in \mathcal{J}}$, and a family $(G_j)_{j \in \mathcal{J}}$ of random constants. (That we use the same index space $\mathcal{J}$ for two separate primitives is purely for notational convenience; our analyses will actually assume that (F_j) and (G_j) are independent.[5]) For the PRF we require that it be a family $F_j \colon \mathcal{T} \to [0 \mathinner{..} \kappa - 1]$ for all j (that is: a pseudo-random mapping from the tag space to the set of bit positions of a key), and for the constants we require that $G_j \in \{0,1\}^\kappa$ for all j (that is: a pseudo-random element of the key space).

We provide details on our attack. Note that the specification in Fig. 5 assumes that the vfy_j algorithm can maintain state between invocations (but see the discussion below). The idea is that state variable k' is initialised as the random string G_j and then, during vfy_j invocations, bits of this string that are different from the corresponding bit of the target key k (line 04) are updated (line 06) until eventually the condition $k' = k$ (implicit) is reached. The Bernoulli trial (line 02) controls the rate with which such updates happen, and the PRF (line 03) controls which bit position ι is affected in each iteration.

[5]At the expense of introducing more symbols we could also have formally separated the index spaces of (F) and (G). We believe that our concise notation adds significantly to readability.

```
Proc $\mathsf{vfy}_j(k,m,t)$
Initial state: $k' \leftarrow G_j$
00 $v \leftarrow \mathsf{vfy}(k,m,t)$
01 If $v = 0$: Return $v$
02 If $B(\gamma)$:
03    $\iota \leftarrow F_j(t)$
04    If $k'[\iota] \neq k[\iota]$:
05       $v \leftarrow 0$
06       $k'[\iota] \leftarrow\, !\, G_j[\iota]$
07 Return $v$
```

```
Proc $\mathcal{A}(i,j)$
08 $k' \leftarrow G_j$
09 While $k'$ incorrect:
10    Pick any $\alpha$ for MS
11    $(m,t,\beta) \leftarrow \mathrm{Tag}(\alpha)$
12    $v' \leftarrow \mathrm{Vfy}(m,t)$
13    If $v' = 0$:
14       $\iota \leftarrow F_j(t)$
15       $k'[\iota] \leftarrow\, !\, G_j[\iota]$
16 Return $k'$
```

Figure 5: Passive tagging-intact subversion. See Section 2.1 for the meaning of "$B(\cdot)$" and "!". **Left:** Verification subversion as of Section 3.1. **Right:** Key recovering adversary for game KRP as of Section 3.2.

By PRF security, these bit positions can be assumed uniformly distributed (though knowledge of the subversion index j allows tracing which tag is mapped to which position). Any bit flip is communicated to the adversary by artificially rejecting (line 05) the message-tag pair, although it is actually valid.

We specify a corresponding KRP adversary in Fig. 5 (right). It starts with the same random string G_j as the subversion and traces the bit updates of vfy_j until eventually the full key k is reconstructed. We assume that $\mathcal{A}$ can tell whether the key has been recovered (line 09), e.g. testing the key by verifying one or many recorded authentic message-tag pairs with it.

Note that our adversary does not need to know the messages m emerging throughout the experiment. While a corresponding variable does appear in lines 11,12, this is only to formally express a fully passive attack. The core of the attack, in lines 13–15, is independent of m. This considerably adds to the practicality of our attack: While messages are not always secret information, *in practice* they might be hard to obtain. Conducting mass-surveillance attacks will definitely become easier if the attacks depend exclusively on the knowledge of tags (like in our case, line 14).

While we present our subversion attack as stateful, it also works if the vfy_j algorithm forgets its state between any two invocations (i.e., resets k' to G_j upon each invocation of vfy_j). With respect to the detectability and key recovery notions from Section 3, the attack's performance is the same whether the subversion is stateful or not. The reason for presenting the stateful version here is that the latter offers better correctness *after* a subversion is detected. (This case is practically less relevant and not covered by our formal models.)

We establish the following statements about the key recoverability and undetectability of our passive subversion attack.

Theorem 1. *Let $\mathcal{A}$ be defined as in Fig. 5 (right) and vfy_j as defined in Fig. 5 (left). If F_j behaves like a random function and constants G_j are uniformly distributed, then for any message sampler* MS*, the key recovery advantage $\mathbf{Adv}^{\mathrm{krp}}_{\mathrm{MS}}(\mathcal{A})$ is expected to reach value 1 once the adversary $\mathcal{A}$ has processed $O(\kappa \log \kappa)$ messages.*

Proof. We model algorithm $\mathcal{A}(i,j)$ by the experiment $\mathrm{CC}(S,\alpha)$ from Fig. 1, with $S = [0 \,..\, \kappa - 1]$ and $\alpha = \gamma/2$. The (pseudo-)randomness of F_j ensures that elements of $s \in S$, here representing the possible values of the index ι (line 03), are picked uniformly at random. The probability $\alpha = \gamma/2$ arises through success of the CC experiment being equivalent to the condition $v' = 0$ in line 12 (Fig. 5) and that for any tag t the probability $\Pr[k'[\iota] \neq k[\iota]]$ (line 04) for $\iota \leftarrow F_j(t)$ is 1/2; success is conditional on $B(\gamma)$. We now apply Lemma 1, which gives the expected number of messages to be sent as $O(\kappa \log \kappa)$. □

Theorem 2. *Let $\mathcal{A}$ be an adversary playing the* UDV *game (as in Fig. 3, centre), such that $\mathcal{A}$ makes at most q queries to the verification oracle* Vfy*. The undetectability advantage of the subversion vfy_j, as defined in Fig. 5 (left), is bounded by $\mathbf{Adv}^{\mathrm{udv}}(\mathcal{A}) \leq 1 - (1-\gamma)^q$.*

Proof. Any detection adversary playing Game UDV against the subverted vfy_j must, in order to win, trigger $v = 0$ with a correct pair (m,t). More precisely, adversary $\mathcal{A}$ must find a pair (m,t) such that $\mathsf{vfy}(k,m,t) = 1$ but $\mathsf{vfy}_j(k,m,t) = 0$. Figure 6 (left) shows the (obviously) best adversarial strategy. Even if the adversary can submit (m,t) such that $\iota \leftarrow F_j(t)$ is assigned in line 03 (Fig. 5), it still

```
Proc $\mathcal{A}$
00 Pick any $k \in \mathcal{K}$
01 Pick any $m \in \mathcal{M}$
02 $t \leftarrow \mathsf{tag}(k, m)$
03 Repeat $q$ times:
04     $v \leftarrow \mathrm{Vfy}(k, m, t)$
05     If $v = 0$:
06         Return 0
07 Return 1
```

```
Proc $\mathcal{A}$
00 Pick any $k \in \mathcal{K}$
01 Pick any $m \in \mathcal{M}$
02 $t \leftarrow \mathsf{tag}(k, m)$
03 $S \leftarrow \{t\}$
04 Repeat $q$ times:
05     $t' \leftarrow_{\$} \mathcal{T} \setminus S$
06     $v \leftarrow \mathrm{Vfy}(k, m, t')$
07     $S \overset{\cup}{\leftarrow} \{t'\}$
08     If $v = 1$:
09         Return 0
10 Return 1
```

Figure 6: Detection adversaries for Game UDV. **Left:** For the passive attack from Fig. 5. **Right:** For the active attack from Fig. 7.

requires that $B(\gamma)$ succeeds; thus $\Pr[v = 0] \leq \gamma$ in line 05 (Fig. 6). Clearly, adversary $\mathcal{A}$ (Fig. 6) always returns 1 when interacting with the unsubverted verification algorithm. Thus, $\mathbf{Adv}^{\mathrm{udv}}(\mathcal{A}) = |\Pr[\mathrm{UDV}^1(\mathcal{A})] - \Pr[\mathrm{UDV}^0(\mathcal{A})]| \leq 1 - (1 - \gamma)^q$. □

4.2 Active Attack

We define our active subversion of the verification algorithm of an authentication scheme in Fig. 7 (left). It is parameterised by a large index space $\mathcal{J}$, a PRF $(F_j)_{j \in \mathcal{J}}$, a PRP $(P_j)_{j \in \mathcal{J}}$, and a family $(G_j)_{j \in \mathcal{J}}$ of random constants. (As in Section 4.1, our analyses will assume that (F_j) and (P_j) and (G_j) are independent.) For the PRF we require that it be a family $F_j \colon \mathcal{T} \to [0 \mathinner{..} \kappa - 1]$ for all j (that is: a pseudo-random mapping from the tag space to the set of bit positions of a key), for the PRP we require that it be a family of permutations $P_j \colon \mathcal{T} \to \mathcal{T}$ for all j, (that is: a pseudo-random bijection on the tag space), and for the constants we require that $G_j \in \{0, 1\}^\kappa$ for all j (that is: a pseudo-random element of key space).

```
Proc $\mathsf{vfy}_j(k, m, t)$
00 $v \leftarrow \mathsf{vfy}(k, m, t)$
01 If $v = 1$: Return $v$
02 $t' \leftarrow P_j(t)$
03 $v' \leftarrow \mathsf{vfy}(k, m, t')$
04 If $v' = 0$: Return 0
05 $\iota \leftarrow F_j(t)$
06 If $k[\iota] \neq G_j[\iota]$:
07     Return 1
08 Return 0
```

```
Proc $\mathcal{A}(i, j)$
09 $k' \leftarrow G_j$
10 While $k'$ incorrect:
11     Pick any $\alpha$ for MS
12     $(m, t, \beta) \leftarrow \mathrm{Tag}(\alpha)$
13     $t' \leftarrow P_j^{-1}(t)$
14     $v' \leftarrow \mathrm{Vfy}(m, t')$
15     If $v' = 1$:
16         $\iota \leftarrow F_j(t)$
17         $k'[\iota] \leftarrow\, !\, G_j[\iota]$
18 Return $k'$
```

Figure 7: Active tagging-intact subversion. See Section 2.1 for the meaning of "!". **Left:** Verification subversion as of Section 3.1. **Right:** Key recovering adversary for game KRA as of Section 3.2.

The idea of our attack is as follows. Lines 00,01 of vfy_j ensure that valid message-tag pairs are always accepted (no limitation on correctness). If however a tag t is identified as not valid, then a secret further check is performed: The original tag t is mapped to an unrelated tag using the random permutation (line 02), and the result t' is checked for validity for m (line 03). For standard (invalid) tags t also this second verification should fail, and in this case algorithm vfy_j rejects as expected (line 04). The normally not attainable case that the second verification succeeds is used to leak key bits. The mechanism for this (lines 05–08) is as in our passive attack from Section 4.1, namely by communicating via accept/reject decisions the positions where the bits of a pseudo-random value G_j and the to-be-leaked key k differ.

We construct a corresponding key recovering adversary $\mathcal{A}$ in Fig. 7 (right). Exploiting that it has access to subversion index j, it takes authentic tags t (line 12) and maps them to auxiliary tags t' (line 13) such that when verifying the latter (line 14) the first verification in vfy_j (line 00) will fail but the second verification (line 03) will succeed. The information thus leaked by the verification routine is used to reconstruct target key k in the obvious way (lines 15–17).

We establish the following statements about the key recoverability and undetectability of our active subversion attack.

Theorem 3. *Let $\mathcal{A}$ be defined as in Fig. 7 (right) and vfy_j as defined in Fig. 7 (left). If F_j and P_j behave like random functions, constants G_j are uniformly distributed, and the MAC scheme is unforgeable, then for any message sampler* MS*, the key recovery advantage $\mathbf{Adv}^{\mathrm{kra}}_{\mathrm{MS}}(\mathcal{A})$ is expected to reach value* 1 *once the adversary $\mathcal{A}$ has processed $O(\kappa \log \kappa)$ messages.*

Proof. By the unforgeability of the MAC scheme, each invocation of algorithm vfy_j in an execution of attack $\mathcal{A}(i,j)$ can be assumed to have $v = 0$ in line 01 (Fig. 7), and thus line 02 is reached. By the (pseudo-)randomness of P_j, in line 02 the condition $t' \neq t$ holds with probability $\beta = 1 - 1/|\mathcal{T}|$. We model algorithm $\mathcal{A}(i,j)$ by the experiment $\mathrm{CC}(S,\alpha)$ from Fig. 1, with $S = [0 \mathinner{..} \kappa - 1]$ and $\alpha = \beta/2$. The (pseudo-)randomness of F_j ensures that elements of $s \in S$, here representing the possible values of the index ι (line 05), are picked uniformly at random. The probability $\alpha = \beta/2$ arises through success of the CC experiment being equivalent to the condition $v' = 1$ in line 15. This occurs precisely when vfy_j returns 1 in line 07, which is conditional on vfy_j reaching past line 02. The probability that $v' = 1$ in line 15 is 1/2 as this is the probability that for any tag t and $\iota \leftarrow F_j(t)$, $k[\iota] \neq G_j[\iota]$ (line 06). We now apply Lemma 1, which gives the expected number of messages to be sent as $O(\kappa \log \kappa)$. □

Theorem 4. *Let $\mathcal{A}$ be an adversary playing the* UDV *game (as in Fig. 3, centre), such that $\mathcal{A}$ makes at most q queries to the verification oracle* Vfy*. Suppose that $\mathbf{Adv}^{\mathrm{uf}}(\mathcal{A}') \leq \varepsilon$ for all UF adversaries $\mathcal{A}'$ (as in Fig. 2, left). If P_j behaves like a random function then the undetectability advantage of the subversion vfy_j, as defined in Fig. 7 (left), is given by $\mathbf{Adv}^{\mathrm{udv}}(\mathcal{A}) \leq 1 - (1-\varepsilon)^q$.*

Proof. Any detection adversary playing Game UDV against the subverted vfy_j must, in order to win, trigger $v = 1$ with a bogus (m,t). That is, a message-tag pair (m,t) with $\mathsf{vfy}(k,m,t) = 0$ but $\mathsf{vfy}_j(k,m,t) = 1$. This will occur if $t = P_j(t')$, where $\mathsf{vfy}(k,m,t') = 1$. As j is chosen uniformly randomly from $\mathcal{J}$ and P is a (pseudo-)random function, the only strategy is to sample values of t' and test whether $\mathrm{Vfy}(k,m,t') = 1$. Algorithm $\mathcal{A}$ in Fig. 6 (right) shows this strategy. When $\mathcal{A}$ interacts with the unsubverted verification algorithm, by the unforgeability of the MAC scheme we can assume that $\mathrm{Vfy}(k,m,t')$ never returns 1. Thus $\Pr[\mathrm{UDV}^1(\mathcal{A})] = 1$. When interacting with the subverted verification algorithm, $\mathcal{A}$ returns 0 by either triggering line 01 or line 07 of vfy_j. We assume that line 01 is not triggered, by the unforgeability of the MAC scheme. Triggering line 07 happens with probability $\leq \varepsilon$. Thus we have $\mathbf{Adv}^{\mathrm{udv}}(\mathcal{A}) = |\Pr[\mathrm{UDV}^1(\mathcal{A})] - \Pr[\mathrm{UDV}^0(\mathcal{A})]| \leq 1 - (1-\varepsilon)^q$. □

References

[ABS16] Fatema Al Mansoori, Joonsang Baek, and Khaled Salah. Subverting MAC: How authentication in mobile environment can be undermined. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 870–874, April 2016.

[AMV15] Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signature schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 364–375. ACM Press, October 2015.

[AP19a] Marcel Armour and Bertram Poettering. Substitution attacks against message authentication. *IACR Trans. Symmetric Cryptol.*, 2019(3):152–168, 9 2019. https://tosc.iacr.org/index.php/ToSC/article/view/8361.

[AP19b] Marcel Armour and Bertram Poettering. Subverting decryption in AEAD. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, LNCS, Springer, 2019 (to appear). https://eprint.iacr.org/2019/987.

[BH15] Mihir Bellare and Viet Tung Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 627–656. Springer, Heidelberg, April 2015.

[BHBN14] Swarup Bhunia, Michael S Hsiao, Mainak Banga, and Seetharam Narasimhan. Hardware trojan attacks: threat analysis and countermeasures. *Proceedings of the IEEE*, 102(8):1229–1247, 2014.

[BJK15] Mihir Bellare, Joseph Jaeger, and Daniel Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1431–1440. ACM Press, October 2015.

[BKR16] Mihir Bellare, Daniel Kane, and Phillip Rogaway. Big-key symmetric encryption: Resisting key exfiltration. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 373–402. Springer, Heidelberg, August 2016.

[BL17] Sebastian Berndt and Maciej Liskiewicz. Algorithm substitution attacks from a steganographic perspective. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1649–1660. ACM Press, October / November 2017.

[BPR14] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 1–19. Springer, Heidelberg, August 2014.

[CDL17] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation with subverted TPMs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 427–461. Springer, Heidelberg, August 2017.

[DFP15] Jean Paul Degabriele, Pooya Farshim, and Bertram Poettering. A more cautious approach to security against mass surveillance. In Gregor Leander, editor, *Fast Software Encryption – FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 579–598. Springer, Heidelberg, March 2015.

[DGG$^+$15] Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 101–126. Springer, Heidelberg, April 2015.

[DMS16] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 341–372. Springer, Heidelberg, August 2016.

[FJM18] Marc Fischlin, Christian Janson, and Sogol Mazaheri. Backdoored hash functions: immunizing HMAC and HKDF. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 105–118. IEEE, 2018.

[FM18] Marc Fischlin and Sogol Mazaheri. Self-guarding cryptographic protocols against algorithm substitution attacks. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 76–90. IEEE, 2018.

[GBPG03] Eu-Jin Goh, Dan Boneh, Benny Pinkas, and Philippe Golle. The design and implementation of protocol-based hidden key recovery. In Colin Boyd and Wenbo Mao, editors, *ISC 2003: 6th International Conference on Information Security*, volume 2851 of *Lecture Notes in Computer Science*, pages 165–179. Springer, Heidelberg, October 2003.

[Gol11] Dieter Gollmann. *Computer Security (3. ed.)*. Wiley, 2011.

[Isa13] Richard Isaac. *The pleasures of probability*. Springer Science & Business Media, 2013.

[KK03] Lars R. Knudsen and Tadayoshi Kohno. Analysis of RMAC. In Thomas Johansson, editor, *Fast Software Encryption – FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 182–191. Springer, Heidelberg, February 2003.

[MS15] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 657–686. Springer, Heidelberg, April 2015.

[MZY$^+$18] Hui Ma, Rui Zhang, Guomin Yang, Zishuai Song, Shuzhou Sun, and Yuting Xiao. Concessive online/offline attribute based encryption with cryptographic reverse firewalls - secure and efficient fine-grained access control on corrupted machines. In Javier López, Jianying Zhou, and Miguel Soriano, editors, *ESORICS 2018: 23rd European Symposium on Research in Computer Security, Part II*, volume 11099 of *Lecture Notes in Computer Science*, pages 507–526. Springer, Heidelberg, September 2018.

[RTYZ16a] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 34–64. Springer, Heidelberg, December 2016.

[RTYZ16b] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Destroying steganography via amalgamation: Kleptographically CPA secure public key encryption. Cryptology ePrint Archive, Report 2016/530, 2016. http://eprint.iacr.org/2016/530.

[RTYZ17] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Generic semantic security against a kleptographic adversary. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 907–922. ACM Press, October / November 2017.

[RTYZ18] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Correcting subverted random oracles. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 241–271. Springer, Heidelberg, August 2018.

[SFKR15] Bruce Schneier, Matthew Fredrikson, Tadayoshi Kohno, and Thomas Ristenpart. Surreptitiously weakening cryptographic systems. Cryptology ePrint Archive, Report 2015/097, 2015. http://eprint.iacr.org/2015/097.

[Sim83] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum, editor, *Advances in Cryptology – CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1983.

[YY96] Adam Young and Moti Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 89–103. Springer, Heidelberg, August 1996.

[YY97] Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 62–74. Springer, Heidelberg, May 1997.