Contingency Revisited: Secure Construction and Legal Implications of Verifiably Weak Integrity

Henrich Pöhls

Contingency Revisited: Secure Construction and Legal Implications of Verifiably Weak Integrity

Henrich C. Pöhls

Institute of IT-Security and Security Law, University of Passau, Germany
hp@sec.uni-passau.de

Abstract. Digital signatures are by far the most prominent mechanisms to detect violations of integrity. When signing rights are delegated, the integrity protection is gradually weaker as the delegatee's actions are not considered integrity violations. Taken to an extreme, delegating the right to undetectably change everything to everyone will achieve a property called contingency. Contingency was introduced as the "dual of integrity" in 2009 by Rost and Pfitzmann in German [26] and later translated into English in 2011 [4]. Contingency describes the exact opposite of integrity: the provable absence of integrity. Following this line of privacy research, this paper gives the first rigorous definition of contingency and presents a cryptographic protocol build upon a transparent sanitizable signature scheme. Hence, contingency is a verifiable statement that the signer explicitly desired that the integrity status of data is not verifiable. We analyze legal implications and applications of contingent information.

1 Introduction

Integrity is among the classic security protection goals stated in textbooks, together with confidentiality and availability. In 2009 the "dual of integrity" [26] termed *contingency* [4] got introduced by Rost and Pfitzmann. Here, 'dual' means that something can be proven to be in only one of two states, in this case data either has "integrity" or it is "contingent". Hence, *contingency* describes the verifiable state that the data's integrity is intended to be unknown. This work revisits this interesting privacy property, which is in line with Pfitzmann's long history of privacy research [14].

In particular, we noted that it can be securely achieved if signing rights are cryptographically delegated. We argue that a delegation will gradually weaken integrity protection, which generates integrity protection of weaker strengths. From this follows a need to re-evaluate the trust the verifier can put on information under weaker or no integrity protection. Hence, we will also look at legal implications of contingency.

Motivating Example. Contingency is introduced as a method to increase privacy. As an example Rost and Pfitzmann mention the following legal case [26]:

It is the right of a female employee to answer the employer's question if she is pregnant with "No". Even if pregnant at the time of her statement, she shall face no consequences shall the fact become known later. Hence, she has the right to revise a previously given statement without the revision being held against her. The case shall be handled as if she modified her answer to "Yes" in all records that would have shown her lying [11].

Technically this means that revisions of this answer must not be detected. When revisions go undetected, then even if the statement is still verifiable to have originated from her, one cannot hold her technically responsible and hence one cannot construct legal consequences upon the technically integrity-protected record. This loss of integrity protection does not directly impact origin authentication, which is important when analyzing possible legal consequences.

Digital Signatures to Protect Integrity. Integrity can be protected by several means, e.g., access control, message authentication codes or digital signatures. For this paper we focus on the standalone and a-posteriori verifiable integrity protection mechanism using digital signatures. Digital signatures fulfill the technical requirements of contingency protection mechanisms to capture consent and offer the requested origin authentication and a legal assessment can be based on the existing signature regulations and the digital signature's property of non-repudiation. Moreover, the delegation of the right to generate a valid digital signature on behalf of the signer is a well studied concept in cryptography. One cryptographic solution are proxy signatures, as introduced by Mambo et al. in [23], or group signatures, as introduced by Chaum and Van Heyst [10]. Lately, also sanitizable signatures are introduced by Ateniese et al. as a concept in which "the signer delegates the signing rights of parts of the message to a designated party, the sanitizer."[2].

Weak Integrity up to Contingency. A standard non-delegated digital signature offers very strong integrity protection, as every subsequent change is detected. This is technically accepted and got legally backed. For example the European Union's Directive 1999/93/EC on a Community Framework for Electronic Signatures requires "that any subsequent change of the data is detectable" [13]. Hence, unforgeable digital signature schemes combined with other organizational[1] and technical measures[2] fulfill legal requirements set out by EU signature legislation. This would allow the verifier to give information with this level of integrity protection and accountability the ultimate "Statutory Trust" [25]. In the extremest case the integrity protection becomes verifiably inexistent. Hence, one achieves contingency [26], the "dual of integrity" [4].

The property of contingency itself might at first sight look unintuitive, undesirable and not worth pursuing. As an extreme corner case, however, it is useful to

[1] e.g., public key must be bound to legal entity, as done by public key infrastructures
[2] e.g., signature must be created by a secure signature creation device, like a smartcard

discuss how much integrity protection is needed to fulfill legal and application specific requirements. According to Rost and Pfitzmann the first obvious technical solution ensuring contingency would be **not** to employ integrity protecting mechanisms [26] in the first place. However, not applying integrity protection must be done **intentionally**. It must be a conscious decision not to apply mechanisms, which allow integrity violations to be detected [26]. This intention must be verifiable to allow a precise legal analysis. Hence, a naive technical solution which just does **not** digitally sign does **not** provide contingency.

As we have mentioned accountability before, note that even in the case of contingent data the signer remains identifiable as the origin of the data by means of his public key in which the verifier trust. Quite contrary to being able to verify the data's origin, the data's integrity is verifiably not known. It is exactly this contradiction that we will further dissect in this paper with respect to the legal implications and issues of trust.

Contribution and Outline. This work analyzes the impact of delegation of signing rights on integrity. Obviously, it allows reducing the initially very strong integrity protection by allowing others to sign on behalf of the signer. However, we show how to achieve contingency and analyze the possible legal consequences and henceforth the trust that a verifier can assign to the information protected to such a level of integrity.

We first give informal definitions of the notions and look at the state of the art in Sect. 2. Next, in Sect. 3 we give application examples and highlight the legal impact. For the construction of a provably scheme achieving contingency, we also give a more rigorous game based definition of contingency in Sect. 4. We show that our cryptographic solution achieves contingency securely in our security model. To the best of our knowledge, this is the first secure technical solution achieving contingency. Neither, the original work by Rost and Pfitzmann [26], nor the same authors [28] nor closely related work [4] give a more precise definition or a precise technical construction. We conclude and give an outlook in Sect. 5.

2 State of the Art and Definitions

The notion of Integrity. Gollmann's computer security textbook acknowledges from the first edition in 1999 that it is "not easy to give a concise definition" [15] of integrity. Other definitions of integrity allow to differentiate between authorized and unauthorized modifications.[15] We do not want to do another hunt for an integrity definition. For this paper it is sufficient to see integrity as a state describing data which is verifiably to be "as it was supposed to be"[15] since the integrity protection was applied. We define it as follows:

Definition 1 (Integrity:). Integrity *describes the verifiable state that data "is as it is supposed to be"[15] since the integrity protection was applied. The applied*

integrity protection mechanism intrinsically defines which actions will make the verification of integrity return a negative result.

This definition, in accordance with [4, 15, 25, 26] acknowledges that data is "supposed" to undergo changes and that the data's integrity status shall remain unchanged for those changes. Above definition does not consider the protection of wrong information to be a problem of data integrity [16]. And "what is authorized"[25] must be codified in the integrity protection mechanism [25].

The notion of Contingency. Rost and Pfitzmann defined the "dual of integrity" [26] to be the verifiable state of not having integrity. It is defined to be more than just not having integrity. Duality in this case means that information can only be in the state of *integrity* or in the state of *contingency*, but information cannot be in both, as this would be a contradiction. The work by Rost and Pfitzmann from 2009, available in German, called it "Kontingenz" [26], which Bedner and Ackermann translated into "contingency" [4]. The term contingency is later also called *intervenability* by Rost and Bock in [28]. This English version of the German article [27] describes contingency as the data subject's "ability to intervene" [28]. *Data subject* is the term for "an identified or identifiable natural person" given in the European data protection law in the EU Directive 95/46/EC [12]. The ability to intervene is described as an umbrella term to allow the data subject to exercise the rights given by data protection law. [28] In detail contingency allows "information processing entities [...] to demonstrate that they actually have steering control over their systems and are not dominated by the system" [27]. In accordance with [28] we see both terms as synonyms and continue to use the original term's translation contingency.

Definition 2 (Contingency:). Contingency *describes the verifiable state that the data's integrity is intended to be unknown. We call data in that state* contingent*. Contingency is a verifiable property explicitly established by the applied protection mechanism and not an accidental state.* [3]

To put it differently: For contingency-protected data it is technically not deductible if the data itself is sincere and thus integrity would hold, or if it is insincere, and thus integrity would be violated. In this context, we use the terms *sincere* and *insincere* to circumvent using English terms which have already a pre-defined meaning in information security. In particular the translations of the German original term "echt" like "genuine", "correct" or "authentic" cause misunderstanding by having pre-occupied meanings. The opposite of integrity is often found to be defined differently, e.g. in the information flow community the opposite is confidentiality [5].

[3] For simplicity, we omit the original definition's constraint that this state might only be a temporal state, i.e., only for a given time [26].

The notion of contingency as given by Rost and Pfitzmann becomes more clearly related to a general privacy notion when studying their motivation. Other work of Pfitzmann in the area of privacy can be found in [14]. The absence of integrity is described as a method to increase privacy in [26]. Hence, the work of Bedner and Ackermann subsumes properties like *plausible deniability* under the notion of contingency in [3]. Rost and Pfitzmann also mention the special legal case where one has the right to lie, in other words, one has the right to revise a previously given statement without it being detected. From the fact that revisions go undetected follows that one cannot held responsible and hence one faces no consequences. The example given is the right of a female employee to answer the employer's question if she is pregnant with "No". Even if pregnant at the time of the statement, she shall face no consequences shall the fact become known later. Hence, it is as if she modified her answer in all records to "Yes" [11].

Following the work of Rost, Pfitzmann and Bock [26, 28] we deduct that a technical contingency solution has to fulfill all of the following requirements:
(a) The integrity of contingent information cannot be established.
(b) The absence of integrity on contingent information cannot be established.
(c) The state is not an error state, but explicit and verifiable by third parties.

Note, contingency cannot be achieved if:

- **no consent is given.** Contingency must be applied explicitly and consented. Assume one signs one half of data with a digital signature to gain integrity protection and just not signing the other half. This does not create contingent data for the unsigned half of the data.
- **it is not indistinguishable.** The previously mentioned half signed data is obviously distinguishable. The same holds true for signing the complete data to protect integrity and than change something to violate the integrity protection, i.e., invalidate the signature.

Due to the latter, technical constructions must make changes indistinguishable.

The notion of Sanitizable Signatures. We want the property of integrity to be valid even after authorized modifications, which have been explicitly endorsed by the party that applied the integrity protection. This freedom is given by called malleable signature schemes. In a nutshell, the concept of malleable signature schemes is allowing a third-party to make certain modifications of signed messages. These changes than have no impact on the validity of integrity and are not considered forgeries. To create a valid signature on a modified message the third-party does not need the signer's private key, nor is the signer involved. For malleable signatures we assume a document m split in ℓ non-overlapping blocks, i.e., $m = m[1]||\ldots||m[\ell]$. $||$ is a uniquely reversible concatenation.
We distinguish the allowed modifications of malleable signatures schemes to group them into three main categories: Redactable signatures, append only signatures and sanitizable signatures. Redactable signatures allow a third party

to modify signed data by removing blocks, as introduced by Steinfeld et al. in [30] and by Johnson et al. in [19]. Other redactable signature schemes emerged since then, e.g., [9, 24, 1, 22]. Second, Kiltz et al. introduced append-only signatures [20]. Redactable and append-only malleable signature schemes would allow a third party to modify the signed message without requiring a secret key of their own. Finally, sanitizable signature schemes (SSS), as introduced by Ateniese et al. [2] allow the third party, if authorized, to change the data arbitrarily. Hence, instead of redacting or appending blocks, a SSS allows the third party, who we call *sanitizer*, to exchange the contents of an existing blocks with an arbitrary string $\in \{0,1\}^*$. Related work can be found in [6, 8, 21]. SSS do not allow to completely remove blocks and require the sanitizer to know a private key. Hence, the operation of sanitize is not per-se public. However, it is not generally transparent. The signer can decide which blocks are *immutable* and cannot be changed. For the other blocks, called *admissible*, the sanitizer is not restricted in his changes. However, the signer could restrict the sanitizer to values as proposed by Zhang et al. [31] or by the use of Bloom-Filters as proposed by Klonoswki et al. [21]. We later show that we require the malleable scheme to offer transparency, known to imply privacy [6]. However, not even all schemes are private, e.g. for the scheme by *Kundu* and *Bertino* [22] attacks on transparency and privacy [7] and one attacking structural integrity [29] have been given.

3 Applications and Legal Implications

The general legal consequences of using a sanitizable signature are of separate concern and out of scope for this paper. We will illustrate the legal consequences using three possible applications of a reduced integrity protection.

"Right to lie". This is the one example for use of contingency given by Rost and Pfitzmann [26]. There is actually no right to lie. In most cases, e.g., to lie for financial gains, it has legal consequences. Only in exceptional cases there are explicitly no legal consequences of a lie. We follow the example from [26]: In a job interview a female applicant can give an untruthful answer to the employer's question: "Are you pregnant?". However, she shall not be held accountable for the lie, as a truthful answer "Yes, I am pregnant." could have negative consequences and hence violate the equal treatment for men and women. This equal treatment for men and women with regards to access of employment is a codified legal right in the European Economic Community (EEC) [11]. Judges have ruled in favor of lying women[4] and thereby removed any negative consequences that arose from untruthfully answers to that question.[5] Following this legal motiva-

[4] The European court (Fifth Chamber) stated this in his judgement Wiebke Busch vs. Klinikum Neustadt GmbH & Co. Betriebs-KG. (Case C-320/01) on 27.02.2003.

[5] Especially article 2 (1) and (3) of the Directive 76/207/EEC [11]are considering this. In particular article 2 (3) mentions "the protection of women, particularly as regards pregnancy" [11].

tion a technical system should allow the user to not always produce integrity protected data. For the pregnancy example Rost and Pfitzmann state that often technical systems are designed to always answer correctly, e.g., would extract the information about a known pregnancy from the woman's electronic patient record. If the system can be instructed to output contingent data there is no assurance of integrity. However, the visibility of contingency is not negative as it reflects the fact that legally an employee's answer to this question cannot be checked for integrity violations.

Underspecified Statements. Printed forms contain fields that are not to be filled out by the applicant but by administration in a second step. For example, forms state "Applicant must not fill in grey fields." [17], or the template approach recently discussed in [18]. On paper, the applicant fills only the appropriate fields, leaving the grey ones empty, and signs the form. Electronically, this quickly becomes complex: If the signer fails to sign empty grey fields the administration could theoretically add additional fields later. If the empty grey fields are fully integrity protected by classical signatures the verification of a form completed by administration fails. Only after the removal of the administration's content the applicant's signature verifies again, but this requires additional business logic in the verification preparation process.

With weak integrity selectable on message parts the applicant can sign the form and define his fields as fully integrity protected and the administration's grey fields as contingent. As a second step the administration could then use the supplied sanitizer key to fill in the fields. This would remove the need of the verification procedure to remove/exclude the administration's input. Administration shall finally sign the whole application form with an integrity preserving signature to prohibit further changes by third parties. By using a contingent signature scheme for the grey fields, the applicant explicitly acknowledges that he or she knew about the unfilled grey parts. Additionally, the applicant limits the administration to changing only contingent grey fields, while integrity protected fields must be left unchanged, to keep the applicant's signature valid.

Blanket statements. Blank cheques or the above mentioned form fields are an example of what is legally called a blanket statements. Legally, you are allowed to leave certain fields underspecified or empty, allowing them to be filled with information later. If done in a consented way, then any specific information filled in later is attributed to the original signer of a blanket form. Our construction clearly captures the signer's consent.

Economically the risk could be acceptable, if you are delegating signing rights to a delegatee that you can get compensation for damages from later. However, in the extreme case of contingency the risk for the signer maybe severe, as giving out the sanitizer's secret key allows anyone to fill in the contingent part. The

application of blockwise contingency, applied only to certain parts, limits the risk to the damage that can be done filling information into those fields.

The verifier of contingent information can trust that the signer, as the identifiable originating party, has acted explicitly with consent. If the application context allows to deduct that the signer is not to be held liable for the contingent parts, i.e. the administration had to fill that in, then contingency or any visible delegation signals that the signer was well aware that contingent or delegated parts could be changed later. Both allows to provide strong technical evidence that the signer did consent to a change, which is not the case if the integrity protection is not applied or removed, as differentiated in Sect. 2. Note, contingency does not and should not hide the fact itself that data is contingent.

4 Contingency constructed from a Sanitizable Signature

Our construction is based on a unforgeable, immutable and transparent SSS. The desired security properties which Brzuska et al. have already identified and formalized the in [6]. We will restate the SSS security properties as we require them to hold for our construction of an ContingentSS, to increase readability. As already stated, a SSS allows the signer to define a sanitizer, and to define the blocks of a message that can be sanitized. This requires the message m to be split into ℓ blocks denoted as $m[i]$, combined by a uniquely reversible concatenation $||$.

Definition 3 (Sanitizable Signature Scheme). *Following the formal definitions given by Brzuska et al. in [6], a SSS consists of seven efficient algorithms:* $SSS := (KGen_{sig}, KGen_{san}, Sign, Verify, Sanitize, Proof, Judge)$ *such that:*

$\mathsf{KGen}_{\mathbf{sig}}$. *The algorithm* $KGen_{sig}$ *outputs the public and private key of the signer, i.e.* $(pk_{sig}, sk_{sig}) \leftarrow KGen_{sig}(1^{\lambda})$*, where* λ *is the security parameter.*

$\mathsf{KGen}_{\mathbf{san}}$. *The algorithm* $KGen_{san}$ *outputs the public and private key of the sanitizer, i.e.* $(pk_{san}, sk_{san}) \leftarrow KGen_{san}(1^{\lambda})$*, where* λ *is the security parameter.*

Sign. *The Sign algorithm takes as input a message* $m \in \{0,1\}^*$*, the signer's secret key* sk_{sig}*, the sanitizer's public key* pk_{san} *and a description* ADM *of the admissibly modifiable message parts. It outputs a signature (or* $\perp$*, indicating an error):* $\sigma \leftarrow Sign(m, sk_{sig}, pk_{san}, \mathrm{ADM})$.
Note, ADM *is recoverable from any signature* $\sigma \neq \perp$ *by the sanitizer holding the secret key that corresponds to* pk_{san}*.*

Verify. *The algorithm Verify outputs a bit* $d \in \{0,1\}$ *indicating whether the signature* σ*, w.r.t. pk, protecting m is valid (d=1) or invalid (d=0). In particular:* $d \leftarrow Verify(m, \sigma, pk_{sig}, pk_{san})$

Sanitize. *The algorithm Sanitize takes the document m with instructions for the changes* MOD*, the signature* σ*, the signer's public key pk, and the sanitizer's*

secret key. The algorithm applies the changes to m as instructed by MOD*. It outputs a sanitized message m' with a derived signature σ'.*

$$(m', \sigma') \leftarrow \mathsf{Sanitize}(m, \text{MOD}, \sigma, pk_{sig}, sk_{san}).$$

MOD *in our notation could contain more than one modification, which is considered to be the same as consecutive application of each single modification. Also* Sanitize *in general does not need to change the message m itself, i.e., if* MOD $= \emptyset$ *then $m' = m$. The same holds true for the signature, while the algorithm* Sanitize *of an* SSS *generally could produce a new σ' as an output it is not required to, i.e., $\sigma' = \sigma \vee \sigma' \neq \sigma$.*

Proof. *The* Proof *algorithm takes as input the secret signing key sk_{sig}, a message m and a signature σ as well a set of (polynomially many) additional message-signature pairs $(m_i, \sigma_i)_{i=1,2,\ldots,q}$ and the public key pk_{san}. It outputs a string $\pi \in \{0,1\}^*$:* $\pi \leftarrow \mathsf{Proof}(sk_{sig}, m, \sigma, (m_1, \sigma_1), \ldots, (m_q, \sigma_q), pk_{san})$

Judge. *Algorithm* Judge *takes as input a message m and a valid signature σ, the public keys of both parties and a proof π. It outputs a decision $d \in \{\mathtt{Sig}, \mathtt{San}\}$ indicating whether the message-signature pair has been created by the signer or the sanitizer:* $d \leftarrow \mathsf{Judge}(m, \sigma, pk_{sig}, pk_{san}, \pi)$

The correctness of a sanitizable signature scheme is closely related to the correctness for digital signatures. Hence, genuinely signed messages are accepted by the verify algorithm. As shall genuinely sanitized messages be accepted. Finally, for proofs genuinely created by the signer the judge algorithm will correctly decide in favor of the signer.

4.1 Security Properties and Security Model of SSS

Unforgeability. No one should be able to compute a valid signature verifying under pk_{sig} on m without having access to the corresponding secret key sk_{sig}, even when allowed to request signatures on different m_j. This is analogous to the unforgeability requirement for standard signature schemes. A scheme SSS is unforgeable, iff for any efficient (PPT) adversary $\mathcal{A}$, the probability that the game depicted in Fig. 1, returns 1 is negligible (as a function of λ).

Immutability. The sanitizer should not be able to change blocks that are not admissible ($\in$ ADM) and still be able to compute a valid signature verifying under pk on m' without having access to the corresponding secret key sk. This extends the unforgeability notion as it limits the actions of the sanitizer, knowing sk_{san}, to the blocks explicitly marked as admissible by the signer. A scheme SSS is immutable, iff for any efficient (PPT) adversary $\mathcal{A}$, the probability that the game depicted in Fig. 2, returns 1 is negligible (as a function of λ).

Experiment $\mathsf{Unforgeability}^{\mathsf{SSS}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk_{\text{sig}}, \dots), \mathsf{Sanitize}(\dots, sk_{\text{san}}), \mathsf{Proof}(sk_{\text{sig}}, \dots)}(pk_{\text{sig}}, pk_{\text{san}})$
  let $i = 1, 2, \dots, q$ index the queries and answers to the signing oracle, e.g., m_i, σ_i, $\text{ADM}_i, pk_{\text{san}i}$.
  let $j = 1, 2, \dots, r$ index the queries and answers to the sanitizing oracle, e.g., (m'_j, σ'_j).
return 1 iff
  $\mathsf{Verify}(pk, m^*, \sigma^*) = 1$ and
  $\forall i : 1 \le i \le q$, $(pk_{\text{san}}, m^*) \neq (pk_{\text{san}i}, m_i)$ and
  $\forall j : 1 \le j \le r$, $(pk_{\text{sig}}, m^*) \neq (pk_{\text{sig}j}, m'_j)$

Fig. 1. Unforgeability for SSS

Experiment $\mathsf{Immutability}^{\mathsf{SSS}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$d \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_{\text{sig}}, \cdot), \mathsf{Proof}(sk_{\text{sig}}, \dots, pk_{\text{sig}}, \cdot)}(pk_{\text{sig}})$
  let $i = 1, 2, \dots, q$ index the queries and answers to the signing oracle, e.g., m_i, σ_i, $\text{ADM}_i, pk_{\text{san}i}$.
return 1 iff
  $\mathsf{Verify}(pk, m^*, \sigma^*) = 1$ and
  $\forall i : 1 \le i \le q$, we have:
    $pk_{\text{san}} \neq pk_{\text{san}i}$, or
    $m^*[j_i] \neq m_i[j_i]$ for a $j_i \notin \text{ADM}_i$

Fig. 2. Immutability for SSS

Privacy. Privacy is similar to the standard indistinguishability notion for encryption schemes. Adversaries should not be able derive additional information besides what can be derived from the received message-signature pair. A SSS is private, iff for any efficient (PPT) adversary $\mathcal{A}$, the probability that the game depicted in Fig. 3, returns 1 is negligibly close to $\frac{1}{2}$ (as a function of λ).

Transparency. The verifier should not be able to decide whether a signature has been created directly by the signer using Sign, or through application of the modification algorithm Sanitize to a previously signed message. From a signed document one cannot tell whether it is a freshly signed version, or a potentially sanitized version. A SSS is transparent, iff for any efficient (PPT) adversary $\mathcal{A}$, the probability that the game depicted in Fig. 4, returns 1 is negligibly close to $\frac{1}{2}$ (as a function of λ).

Experiment $\mathsf{Privacy}_{\mathsf{SSS}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$b \xleftarrow{\$} \{0, 1\}$
$d \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_{\text{sig}}, \cdot), \mathsf{LoRSanit}(\dots, sk_{\text{sig}}, sk_{\text{san}}, b)}(pk_{\text{san}})$
  where oracle LoRSanit for input $m_0, m_1, \text{MOD}_0, \text{MOD}_1, \text{ADM}$:
    if $\mathsf{Sanitize}(m_0, \text{MOD}_0, \sigma, pk_{\text{sig}}, sk_{\text{san}}) \neq \mathsf{Sanitize}(m_1, \text{MOD}_0, \sigma, pk_{\text{sig}}, sk_{\text{san}})$, abort and return $\perp$
    let $(m_b, \sigma) \leftarrow \mathsf{Sign}(m_b, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM})$
    let $(m', \sigma') \leftarrow \mathsf{Sanitize}(m_b, \text{MOD}_b, \sigma, pk_{\text{sig}}, sk_{\text{san}})$
  oracle LoRSanit returns (m', σ').
return 1 iff $b = d$

Fig. 3. Privacy for SSS

Experiment $\mathsf{Transparency}^{\mathsf{SSS}}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}) \leftarrow \mathsf{KGen}_{\text{sig}}(1^\lambda)$
$(pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{KGen}_{\text{san}}(1^\lambda)$
$b \xleftarrow{\$} \{0, 1\}$
$d \leftarrow \mathcal{A}^{\mathsf{Sign}(sk, \cdot), \mathsf{Sanitize}(\dots, sk_{\text{san}}), \mathsf{Proof}(sk_{\text{sig}}, \dots), \mathsf{SanitOrSign}(\dots, sk_{\text{sig}}, sk_{\text{san}}, b)}(pk_{\text{sig}}, pk_{\text{san}})$
  where oracle SignOrRedact for input $m, \text{MOD}, \text{ADM}$:
    computes: $\sigma \leftarrow \mathsf{Sign}(m, sk_{\text{sig}}, pk_{\text{san}}, \text{ADM})$
    modifies it: $(m', \sigma') \leftarrow \mathsf{Sanitize}(m, \text{MOD}, \sigma, pk_{\text{sig}}, sk_{\text{san}})$
    if $b = 1$
      using the above changed m'
      compute: $\sigma' \leftarrow \mathsf{Sign}(m', sk_{\text{sig}}, pk_{\text{san}}, \text{ADM})$
  oracle SignOrRedact returns (m', σ').
return 1 iff $b = d$

Fig. 4. Transparency for SSS

Transparency $\implies$ ***Privacy***. There exists no scheme which is transparent, but not private. Formal proof is given in [6].

Privacy $\not\Rightarrow$ ***Transparency***. There exists a scheme which is unforgeable and private, but not transparent. Formal proof is given in [6]

Unforgeability is Independent. There exists a scheme which is transparent and private, but not unforgeable and vice versa. Formal proof is given in [6].

4.2 Construction of a Contingent Signature Scheme using a SSS

Let us first give a quick sketch of our construction. We build upon a transparent, and hence private, SSS. The signer creates both key pairs and holds sk_{sig}, pk_{sig}, as well as sk_{san}, pk_{san}. We attach the sanitizer secret sk_{san} and distribute it alongside the message. Using the sanitizable signature scheme we sign **and distribute** an extended $k = sk_{\text{san}}||pk_{\text{san}}||pk_{\text{sig}}||m[1]||m[2]||\ldots||m[\ell-1]||m[\ell]$. Every part of the original m of length ℓ is marked admissible such that m is sanitizable for holders of sk_{san}. Only the keys sk_{san}, pk_{san} and pk_{sig}[6] in the front are immutable. Hence, every recipient including the signer, could possibly modify m. This means that the signer explicitly and verifiably denied the integrity protection for the message m. m is contingent, because the transparent SSS prohibits third parties receiving k from detecting if k is original or sanitized.

Definition 4 (Contingency Signature Scheme ContingentSS).

We build upon a secure and transparent SSS $:= (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Sanitize}, \mathsf{Proof}, \mathsf{Judge})$.

CKeyGen. *Returns both key-pairs running* KGen_{sig} *and* KGen_{san}.

CProtect. *Signing the message* m *using the algorithm* CProtect *works on input of the message* m *and the keys* sk_{sig} *and* pk_{san} *as follows:*

To ensure public sanitization of m *only the three keys are immutable:*
$$\text{ADM} = (4, 5, 6, \ldots, \ell, \ell+1, \ell+2)$$
Build an extended message, with the sanitizable m_1 *to* m_ℓ
$$k = sk_{san}||pk_{san}||pk_{sig}||m_1||m_2||\ldots||m_\ell$$
return: $(k, \sigma) \leftarrow \mathsf{Sign}(k, sk_{sig}, pk_{san}, \text{ADM})$
The extended message and signature will be distributed as a pair (k, σ).

CVerify. *Verifying a signature on input* k, σ *and* pk_{sig} *works as follows:*

Extract the public key pk_{san} *from* k.
Verify the signature σ *on* k: $d \leftarrow \mathsf{Verify}(k, \sigma, pk_{sig}, pk_{san})$
iff $d = 1$ *return* 1 *for a valid signature,*
else return 0 *for an invalid signature.*

[6] pk_{san} could be derivable from sk_{san}, but for readability we distribute both.

CChange. *Changing contingent data on input of k and σ by applying the modification defined by* MOD *works as follows:*

Extract sk_{san}, pk_{sig} from k
Calculate a MOD^k *from the supplied* MOD *such that it modifies the message m inside k.*
Generate a new signature σ' and modify k to k':
$(k', \sigma') \leftarrow \mathsf{Sanitize}(k, \text{MOD}^k, \sigma, pk_{sig}, sk_{san})$
If CVerify *on $(k', \sigma') = 0$ abort and return $\perp$.*
Return (k', σ').

Note, ADM can be reconstructed from σ and sk_{san} can be reconstructed from k simply by reversing the concatenation. In general we do not need to previously generate a key pair $(sk_{\text{san}}, pk_{\text{san}})$, CProtect could generate it for each run on the fly. W.l.o.g., we keep it as a parameter to visualize the connection to SSS.

4.3 Security of our Construction

Our above construction ContingentSS is secure iff the underlying SSS is unforgeable, immutable, private, transparent and accountable. To instantiate our scheme one can facilitate the sanitizable signature scheme based on chameleon hashes proposed by Brzuska et al. in [6], it provably offers all the required security properties. We will now shortly sketch the proofs of security with respect to contingency and integrity.

Integrity of Keys. Integrity protection is only required for the keys. They should not be removable from the contingent message.

Proof. The attacker has access to a Signing-Oracle which will generate contingent signatures on supplied messages m_j, denoted as j pairs (k_j, σ_j). To break integrity the attacker needs to change one key of the extended message k^*, i.e., one of the first 3 blocks of k^* must be different from previously queried pairs. The following two properties must hold:
(a) $k_j^*[x] \neq k_j[x]$ for any j and $x = 1, 2, 3$
(b) $1 \leftarrow \mathsf{CVerify}(k^*, \sigma^*, pk_{\text{sig}})$

Hence, the attacker has two options, either a direct forgery or modify a genuinely signed k_j using the underlying Sanitize. Assume the underlying SSS to be unforgeable, then the first option is not possible without breaking immutability of the SSS. k includes only m as admissible parts, hence the attacker cannot modify the keys either. Thus, an attacker breaking our scheme's integrity protection also breaks the immutability or unforgeability of the underlying SSS. □

Contingency of the Message. The informal definition of contingency from Def. 2 has no indication of computational limitations of the party trying to make

the distinction. As with the SSS properties we will define the unconditional contingency property for practical purposes with computationally bounded parties: Contingency protection requires that an attacker cannot distinguish between the original or a subsequently modified message. Any subsequent change to a contingent message would still result in a contingent message. The attacker has only access to a Signing-Oracle, which will generate a contingency protection for the supplied message.

Definition 5 (Contingency:). *We say that data is contingent, iff for any efficient (PPT) adversary $\mathcal{A}$, the probability that the game depicted in Fig. 5, returns* 1 *is negligibly close to* $\frac{1}{2}$ *(as a function of* λ*).*

Proof. Noticeably, the experiment for contingency in Fig. 5 resembles that of transparency from Fig. 4. Assume the underlying SSS to be private, transparent and immutable, then the sanitizer is able to change any part of the message m, which is included in k. From the attacker winning the contingency experiment from Fig. 5 an attacker on the transparency experiment obviously follows. Thus, an attacker breaking our scheme's contingency protection must also break the transparency of the underlying SSS. □

Experiment $\mathsf{Contingency}_{\mathcal{A}}(\lambda)$
$(pk_{\text{sig}}, sk_{\text{sig}}, pk_{\text{san}}, sk_{\text{san}}) \leftarrow \mathsf{CKeyGen}(1^{\lambda})$
$b \xleftarrow{\$} \{0,1\}$
$d \leftarrow \mathcal{A}^{\mathsf{CSign}(sk,\cdot),\mathsf{TouchedOrUntouched}(\ldots,sk_{\text{sig}},b)}(pk_{\text{sig}}, pk_{\text{san}})$
where oracle TouchedOrUntouched for input m and MOD:
computes: $(k, \sigma) \leftarrow \mathsf{CProtect}(m, sk_{\text{sig}}, pk_{\text{san}})$
modifies it: $(k', \sigma') \leftarrow \mathsf{CChange}(k, \text{MOD}, \sigma, pk_{\text{sig}}, sk_{\text{san}})$
if $b = 1$
using the above changed k'
compute: $(k', \sigma') \leftarrow \mathsf{CProtect}(m', sk_{\text{sig}}, pk_{\text{san}})$
return (k', σ').
return 1 iff $b = d$

Fig. 5. Contingency

4.4 Accountability and Integrity Mixed With Contingency

The algorithm Judge from the underlying SSS only works if the signer is actively participating using Proof that needs sk_{sig}. If the signer is not participating it is usually assumed that the signer is the origin. Contingency protection is a conscious decision by the signer, he did not allow integrity violations to be detected [26]. However, if he would participate in the interactive protocol of accountability from the SSS he could indeed remove contingency by giving out correct proofs using Proof. The above construction could easily be adapted to

cater for messages that contain blocks which are either contingent or integrity protected. This is achieved by excluding integrity protected blocks of m from ADM. An attacker cannot change immutable blocks if the underlying SSS is secure, if changed it would result in an invalid signature and hence every change is detected as required for integrity protection.

5 Conclusion and Future Work

In this paper we have described how weaker integrity protections could be cryptographically constructed and used. We have instantiated the not obvious, but interesting data protection concept of contingency, the verifiable absence of integrity, as introduced first by Rost and Pfitzmann [26]. To the best of our knowledge we give the first provably secure construction. Our construction also nicely allows mixing integrity and contingency protection under one signature.

One could obviously argue that a verifier should not trust a message, which is not integrity protected. However, if the sender is left with the option to apply integrity protection or not, i.e. sign the message or not, than the receipt of an unsigned message allows at least two interpretations: either the signer did not sign, or a third-party removed the signature. The latter can be technically achieved without being detected. Applying contingency protection is hence like answering the question "*Do you really want no integrity protection?*" with a clearly audible "*No integrity, thanks.*" Receiving contingent data allows the receiver to verifiably identify the signer's consent to the absence of integrity. This evidence can also be presented and verified by third parties.

We have just shown a first building block, still to be integrated with appropriate application level mechanisms. We hope that this first rigorous notion will foster the discussion and lead to its inclusion in future IT systems.

Bibliography

[1] J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, A. Shelat, and B. Waters. Computing on authenticated data. In *TCC*, pages 1–20, 2012.

[2] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable Signatures. In *ESORICS*, pages 159–177, 2005.

[3] M. Bedner and T. Ackermann. Schutzziele der IT-Sicherheit. *Datenschutz und Datensicherheit (DuD)*, 34(5):323–328, 2010.

[4] Mark Bedner and Tobias Ackermann. Schutzziele der IT-Sicherheit. *Datenschutz und Datensicherheit - DuD*, 34(5):323–328, 2010. ISSN 1614-0702. doi: 10.1007/s11623-010-0096-1.

[5] A. Birgisson, A. Russo, and A. Sabelfeld. Unifying facets of information integrity. In *Information Systems Security*, volume 6503 of *LNCS*, pages 48–65. Springer, 2011.

[6] C. Brzuska, M. Fischlin, T. Freudenreich, A. Lehmann, M. Page, J. Schelbert, D. Schröder, and F. Volk. Security of Sanitizable Signatures Revisited. In *Proc. of PKC 2009*, pages 317–336. Springer, 2009.

[7] C. Brzuska, H. Busch, O. Dagdelen, M. Fischlin, M. Franz, S. Katzenbeisser, M. Manulis, C. Onete, A. Peter, B. Poettering, and D. Schröder. Redactable Signatures for Tree-Structured Data: Definitions and Constructions. In *Proc. of ACNS'10*, ACNS'10, pages 87–104. Springer, 2010.

[8] S. Canard and A. Jambert. On extended sanitizable signature schemes. In *CT-RSA*, pages 179–194, 2010.

[9] Ee-Chien Chang, Chee Liang Lim, and Jia Xu. Short Redactable Signatures Using Random Trees. In *Proc. of CT-RSA*, CT-RSA '09, pages 133–147. Springer, 2009.

[10] D. Chaum and E. Van Heyst. Group signatures. In *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*, pages 257–265. Springer-Verlag, 1991.

[11] EEC. Directive 76/207/eec. `curia.europa.eu/juris/document/document.jsf?text=&docid=48084&pageIndex=0&doclang=EN&mode=doc&dir=&occ=first&part=1&cid=143317`, February 1976.

[12] EU. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, Official Journal of 23 November 1995, L 281, page 31 - 50, Nov. 1995.

[13] EU. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. *Official Journal of the European Communities*, L 12:12–20, 2000.

[14] H. Federrath, M. Hansen, and M. Waidner. Andreas pfitzmann 1958-2010: Pioneer of technical privacy protection in the information society. In *Privacy and Identity Management for Life*, volume 352 of *IFIP Advances in Information and Communication Technology*, pages 349–352. Springer, 2011.

[15] D. Gollmann. *Computer Security*. John Wiley & Sons, 3 edition, 2011. ISBN 0470741153.

[16] D. Gollmann. Veracity, plausibility, and reputation. In *WISTP*, pages 20–28. Springer, 2012.

[17] Shiphol Group. Shiphol pass application form. `www.schiphol.nl/web/file?uuid=1f52accf-920f-4ec9-833b-68f71e26e30e`, Jul. 2009.

[18] C. Hanser and D. Slamanig. Blank digital signatures. Cryptology ePrint Archive, Report 2013/130, 2013. `http://eprint.iacr.org/`.

[19] R. Johnson, D. Molnar, D. Song, and D.Wagner. Homomorphic signature schemes. In *Proceedings of the RSA Security Conference - Cryptographers Track*, pages 244–262. Springer, Feb. 2002.

[20] E. Kiltz, A. Mityagin, S. Panjwani, and B. Raghavan. Append-only signatures. In *Automata, Languages and Programming*, volume 3580 of *LNCS*, pages 101–101. Springer, 2005.

[21] Marek Klonowski and Anna Lauks. Extended Sanitizable Signatures. In *ICISC*, pages 343–355, 2006.

[22] A. Kundu and E. Bertino. Structural Signatures for Tree Data Structures. In *Proc. of PVLDB 2008*, New Zealand, 2008. ACM.

[23] M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures: Delegation of the power to sign messages. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 79(9):1338–1354, 1996.

[24] K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, S. Tezuka, and H. Imai. Digitally Signed Document Sanitizing Scheme with Disclosure Condition Control. *IEICE Transactions*, 88-A(1):239–246, 2005.

[25] H. C. Pöhls and F. Höhne. The Role of Data Integrity in EU Digital Signature Legislation - Achieving Statutory Trust for Sanitizable Signature Schemes. In *Proc. of STM'11*. Springer LNCS, 2011.

[26] M. Rost and A. Pfitzmann. Datenschutz-Schutzziele — revisited. *Datenschutz und Datensicherheit (DuD)*, 33(6):353–358, 2009.

[27] Martin Rost and Kirsten Bock. Privacy by design und die neuen schutzziele. *Datenschutz und Datensicherheit - DuD*, 35(1):30–35, 2011. ISSN 1614-0702. English Version can be found in [28].

[28] Martin Rost and Kirsten Bock. Privacy by design and the new protection goals. `www.european-privacy-seal.eu/results/articles/BockRost-PbD-DPG-en.pdf/view`, 2011.

[29] K. Samelin, H. C. Pöhls, A. Bilzhause, J. Posegga, and H. de Meer. Redactable signatures for independent removal of structure and content. In *ISPEC*, volume 7232 of *LNCS*, pages 17–33. Springer, 2012.

[30] R. Steinfeld, L. Bull, and Y. Zheng. Content extraction signatures. In *Proc. of ICISC'01*, volume 2288, pages 163–205. Springer, 2002.

[31] Fangguo Zhang, Reihaneh Safavi-naini, and Willy Susilo. ID-Based Chameleon Hashes from Bilinear Pairings. In *IACR Cryptology ePrint Archive*, number 208, 2003.