Signature-in-signature: the Last Line of Defence in Case of Signing Key Compromise*

Przemysław Błaśkiewicz, Mirosław Kutyłowski, and Marcin Słowik

Wrocław University of Science and Technology

Abstract. The standard approach for electronic signatures and seals is that Secure Signature Creation Devices are responsible for confidentiality of secret keys and remain under sole control of their owner. The eIDAS Regulation of EU follows this approach. However, once the private key gets compromised (e.g. by a faulty design of the signing device or powerful cryptanalysis of the public key), then the signature model becomes a deadly trap.

We address this problem by creating a second line of defence – when the signing key gets compromized we can still fish out the signatures created by a signature device. The proposed approach of *signature-in-signature* can be implemented in the standard Schnorr signature and can remain hidden until the examination of signatures after the signing key compromise.

Keywords: eIDAS, advanced electronic signature, electronic seal, SSCD, Schnorr signature, key compromise, forgery

To be used in business and administration, an electronic signature must have a legal status equivalent to that of handwritten signatures. For this purpose the lawmakers specify requirements for electronic signatures. The eIDAS regulation [6] provides such a framework for the whole EU. In particular, as all other legal frameworks, it focuses on secure signature creation devices (SSCD). An SSCD has to guarantee that the signatures corresponding to the public key of its owner can be created only by this SSCD and that the owner's consent is given prior to signature creation.

This approach leads to severe problems that are neglected in the legal framework and that are not sufficiently addressed by technical designs. One of the problems is that somebody may learn the private signing key allegedly stored only in the SSCD. There are multiple ways for this unfortunate event to happen. E.g., a powerful adversary may break the underlying scheme and reveal the key anonymously just to create a legal mess for his political or economical advantage. The keys might be compromised also by a faulty design of key generation procedure (e.g. Estonian eID case [5]), side-channel leakage (see e.g. [1]) etc. While there is an option for securing against leakage by appropriate design of cryptographic schemes (see e.g. [7]), nothing helps if the key is exposed by successful cryptanalysis. If there is no secure record, e.g. in a distributed ledger,

* ESORICS'21, conference proceedings, Springer Nature, LNCS 12973, 777-782

witnessing which signatures have been created by the attacked SSCD, or the list of signatures cannot be checked for completeness (cf. [3]), then we get into severe troubles regarding validity of signed documents.

Signature in signature. Our approach of *signature-in-signature* aims to safeguard the legitimate signatures in case of revealing the signing key. The idea is that inside a regular signature (called *outer signature*) there is a hidden *inner signature*. For the inner signature neither the public key is presented nor its existence should be detectable. The public key for the inner signature is revealed at the moment when the standard signing key is compromised.

The key procedures of such a scheme are *signature creation* (subdivided into creation of the inner and outer signatures) and *signature re-validation* executed after revealing the signing key.

Once a signing key is revealed, we expect that we can challenge the validity of all signatures generated with that SSCD. However, the legal ruling is that a qualified signature remains valid if it has been created before the qualified certificates related to the signature was revoked. In practice however, a malicious forger rarely has interest in informing the victim about his success and the victim learns about the attack when it is already too late to revoke any certificates. On the other hand, invalidating en bloc all signatures created with a compromised key is also a very bad idea – it would enable the signer to cancel his past obligations.

What we propose is a scheme where a layman user holding a SSCD can come to a court of law, where via re-validation procedure the signatures created by the SSCD will be confirmed and the other signatures will be rejected.

While some approaches in the literature aim to reach this goal with dedicated schemes (e.g. fail-stop signatures), we aim to design a solution that can be incorporated into standard schemes, so that even up to the moment of re-validation it would be unknown whether this defense has been implemented by the SSCD.

In the following we show an example design based on inner and outer Schnorr signatures. Other designs (e.g. with inner BLS signature and outer DSA signature) are also possible.

1 Example Sig-in-Sig Scheme

Pseudorandomness of the first component of the Schnorr signature. Let $Z(M, x)$ denote the set of elements s such that there is a signature (s, e) of M and the private key x. Then $s \in Z(M, x)$ if there is k such that

$$s = k - x \cdot \mathrm{Hash}(M, g^k) \bmod q \tag{1}$$

where q is the order of the group used. We ask whether it is possible to recognize if $s \in Z(M, x)$, for a known x.

In the random oracle model we may treat Hash as a random function. Then the probability that (1) holds for given s and k is $\frac{1}{q}$, as the hash needs to have

one particular value. So the probability that (1) does not hold for any k equals $(1-\frac{1}{q})^q \approx \frac{1}{e}$. The probability that there are exactly i solutions k for (1) is $\binom{q}{i}\frac{1}{q^i}(1-\frac{1}{q})^{q-i} < \frac{1}{i!}$. So we may assume that the number of solutions k for (1) is negligible with respect to q.

An algorithm $\mathcal{A}$ trying to answer whether $s \in Z(M,x)$ can make a limited number of queries T to the Hash oracle. The probability that every hash query for $\mathrm{Hash}(M, g^k)$ yields a value different from $(k-s)/x \bmod q$ is at least $(1-\frac{t}{q})^T$, where $t, T \ll q$. So the probability does not differ from 1 in a non-negligible way.

We may conclude that except for a negligible number of cases, $\mathcal{A}$ will ask the oracle for k's such that (1) does not hold. Nevertheless, $\mathcal{A}$ has to make the decision whether $s \in Z(M,x)$. In this situation $\mathcal{A}$ is forced to make a random guess.

This motivates the following assumption:

Proposition 1. *Let M be a message and let x be a private signing key for Schnorr signatures. Assume that given an element $s < q$ the problem is to decide whether there is k such that $s = k - x \cdot Hash(M, g^k) \bmod q$, or s has been chosen at random. Then any feasible algorithm $\mathcal{A}$ has at most negligible advantage to provide a correct answer to this problem.*

Schnorr signature with unknown public key. When a Schnorr signature (s,e) is created, then first the element $r = g^k$ is computed. One could design the scheme so that (s,r) is the signature instead of (s,e). With standard verification procedure in mind these seem to be equivalent. However, (s,r) enables immediately derivation of the public key X corresponding to the signature: $e := \mathrm{Hash}(M,r)$, $X := (r/g^s)^{1/e}$. In case of the standard signature (s,e) in order to find X it is necessary to solve the equation

$$e = \mathrm{Hash}(M, g^s \cdot X^e) \tag{2}$$

for unknown X. Note that for each Hash oracle call $\mathrm{Hash}(M,r)$ one can derive a candidate for X which is $(r/g^s)^{1/\mathrm{Hash}(M,r)}$. We see that finding a matching public key to the signature (s,e) of M is equivalent to finding an element r such that $\mathrm{Hash}(M,r) = e$.

Proposition 2. *In the random oracle model for Hash, given (s,e), it is infeasible to derive the public key such that (s,e) is a Schnorr signature of a message M or to decide that there is no such public key. It is also infeasible to decide whether two alleged signatures (s_0,e_0) and (s_1,e_1) for messages M_0 and M_1 correspond to the same public key.*

Scheme Description In this section we show how to insert a full signature into the random component of a Schnorr signature. Due to the signature size, this implies that either this inner signature must be constructed in a different way, or that it is generated over a smaller group. In this paper we follow the second approach as it is much simpler conceptually. A smaller group usually means a

lower degree of security, however inner signatures are used only to separate the forged signatures from the signatures created by a legitimate SSCD. Moreover, the public key for inner signatures remains hidden until the re-validation phase so that attempts at breaking it are minimized.
`Setup.` Two groups for Schnorr signatures are used: the outer group $\mathcal{G}$ of prime order q and an inner group $\mathcal{P}$ of prime order ρ. The bit-length of q should be at least double of the bit-length of ρ in order to provide enough room for the inner signature. Generators g and γ are chosen for, respectively, $\mathcal{G}$ and $\mathcal{P}$. (Let us remark that the outer Schnorr signature can be replaced by any scheme, where the verifier can retrieve g^k for a number k chosen at random by the signatory).

`Outer key generation`: it is the same as in case of the regular Schnorr scheme. Let x be the private key and $X = g^x$ the public key for a device D.

`Inner key generation`: The owner of D creates a pair of keys (y, Y), where $Y = \gamma^y$ and $y < \rho$ is chosen at random.

`Device initialization:` The owner uploads y to D.

`Signature creation:` To sign a message M the following steps are executed:

1. Compute the inner Schnorr signature:
   (a) choose $\kappa < \rho$ at random,
   (b) compute $\epsilon := \mathrm{Hash}_\rho(M, \gamma^\kappa)$,
   (c) compute $\sigma := \kappa - y \cdot \epsilon \bmod \rho$, the first component of the inner Schnorr signature,
2. Compute the outer Schnorr signature:
   (a) $k := L(\sigma, \epsilon)$, where $L(\alpha, \beta)$ means encoding of numbers $\alpha, \beta < \rho$ by a single number smaller than q,
   (b) $e := \mathrm{Hash}_q(M, g^k)$,
   (c) $s := k - e \cdot x \bmod q$,
3. Output the signature (s, e) of M.

The hash functions Hash_ρ, Hash_q map into, respectively, $\mathbb{Z}_\rho$ and $\mathbb{Z}_q$.

Note 1. Any encoding function L can be used, however it should be invertible: given $L(\alpha, \beta)$ it should be possible to derive back α and β or a few candidates for them. For example, if $L(\alpha, \beta)$ can be defined as a number $z < q$ chosen at random from the numbers satisfying $z = \alpha \cdot \rho + \beta \bmod q$.

Note 2. The elements k used for the outer Schnorr signature are chosen in a particular way – they are not arbitrary random elements anymore. However, according to Proposition 2, it is infeasible to distinguish such k's from random strings if we apply the random oracle model for Hash_ρ.

`Outer signature verification`: the standard verification procedure is applied on input public key X and message M.
`Re-validation of signatures`: If the secret key x is leaked, then it is necessary to re-validate all signatures that pass the standard verification procedure. Once a list of such signatures is created, then the owner of D may reveal the public key Y and the following steps are executed for each signature (s, e) from the list:

– the original ephemeral k is recomputed as $k := s + e \cdot x \bmod q$,
– the alleged inner signature is retrieved as $(\sigma, \epsilon) := L^{-1}(k)$,
– (s, e) is validated iff (σ, ϵ) verifies positively with Y and M.

Note 3. Before the breach becomes public, an adversary $\mathcal{A}$ holding the key x may attempt to forge signatures of D. $\mathcal{A}$ may create easily outer signatures verifiable with X. However, in order to create a signature that will not be rejected during the re-validation phase, $\mathcal{A}$ has to create valid inner signatures as well.

If $\mathcal{A}$ can break X by cryptanalysis, then it is likely that he will be able to break Y as well, as the key length here is substantially shorter. However, Y is kept secret and the only input $\mathcal{A}$ may have are some inner signatures retrieved from the signatures created by D (note that $\mathcal{A}$ can compute them as he holds the secret key x). Fortunately, according to Proposition 2 it is infeasible to derive Y from inner signatures alone.

Note that a brute force attack to create a signature and test if it contains an inner signature verifiable with the same (unknown) key Y as signatures created by D is also infeasible. According to Proposition 2, there is no effective test of this kind.

Note 4. Of course, at the time of re-validation all signatures produced allegedly by D have to be presented. This follows from the fact that at this moment the adversary learns Y and thereby may attempt to forge inner signatures as well, after its successful cryptanalysis.

Of course, there is no need to *publish* the plaintexts of all signatures at this moment. It is enough to publish their hashes (or a root of the Merkle tree built on these hashes), before the re-validation procedure starts. Later, each signature can be inspected on demand. The hashes may be simply stored in a distributed ledger with no additional information attached.

Note 5. The Sig-in-Sign schemes based on Schnorr signatures can be extended so that we get an additional security feature which we call *chaining*: during re-validation procedure the SSCD owner must provide the full history of his signatures – otherwise it will become evident that he is hiding some of them. The idea is based on fluctuations of the inner signing key. A former scheme [3] of this kind has been based on a symmetric secret shared by the SSCD and the judge. Now we do not need such a shared key.

Note 6. For an interested reader, in the Appendix we provide an overview of some implementation options with the currently available (and recommended) algebraic structures such that the resulting scheme fulfils the requirements for both the outer and inner signature.

Final remarks. Due to size constraints, this paper presents only a draft solution to the problem of securing signatures *after* their corresponding signing key has been compromised. An ongoing work is under way to present a full working solution with additional features.

At the same time, the authors wish to draw attention to the fact that it is not only the technical means that are required for such solutions to be implementable

– both procedural as well as legal steps must be taken in unison, which will yield a functional solution to digital signature application in real life.

References

1. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. J. Cryptol. **32**(1), 151–177 (2019). https://doi.org/10.1007/s00145-018-9284-1, https://doi.org/10.1007/s00145-018-9284-1
2. Hamburg, M.: Ed448-goldilocks, a new elliptic curve. IACR Cryptol. ePrint Arch. **2015**, 625 (2015), http://eprint.iacr.org/2015/625
3. Kubiak, P., Kutyłowski, M.: Supervised usage of signature creation devices. In: Lin, D., Xu, S., Yung, M. (eds.) Inscrypt 2013, Revised Selected Papers. LNCS, vol. 8567, pp. 132–149. Springer (2013)
4. Lochter, M., Merkle, J.: Elliptic curve cryptography (ECC) brainpool standard curves and curve generation. RFC **5639**, 1–27 (2010)
5. Nemec, M., Sýs, M., Svenda, P., Klinec, D., Matyas, V.: The return of coppersmith's attack: Practical factorization of widely used RSA moduli. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC CCS 2017. pp. 1631–1648. ACM (2017)
6. The European Parliament and the Council of the European Union: Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. Available at: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2014.257.01.0073.01.ENG (2014)
7. Wu, J., Tseng, Y., Huang, S., Tsai, T.: Leakage-resilient certificate-based signature resistant to side-channel attacks. IEEE Access **7**, 19041–19053 (2019). https://doi.org/10.1109/ACCESS.2019.2896773, https://doi.org/10.1109/ACCESS.2019.2896773

A Appendix

In the table below we present some choice of standard groups that may be used to implement the exapmle scheme described in Sect. 1.

Outer signature			Inner signature			ρ^2/q	C.f.
Sec. level	Curve bits	Example curve	Sec. level	Curve bits	Example curve		
160	320	**brainpoolP320t1**	80	160	**brainpoolP160t1**	1.006	[4]
192	384	**brainpoolP384t1**	96	192	**brainpoolP192t1**	1.056	[4]
224	448	**Curve448-Goldilocks**	112	224	**brainpoolP224t1**	2.841	[2,4]
256	512	**brainpoolP512t1**	128	256	**brainpoolP256t1**	0.660	[4]

Table 1. Example groups for Signature-in-Signature Schnorr based schemes