On the (in)security of ROS

Fabrice Benhamouda[1], Tancrède Lepoint[2], Julian Loss[3], Michele Orrù[4], and Mariana Raykova[2]

[1] Algorand Foundation, `fabrice.benhamouda@gmail.com`
[2] Google, {`tancrede,marianar`}`@google.com`
[3] University of Maryland, `lossjulian@gmail.com`
[4] UC Berkeley, `michele.orru@berkeley.edu`

Abstract. We present an algorithm solving the ROS (Random inhomogeneities in a Overdetermined Solvable system of linear equations) problem in polynomial time for $\ell > \log p$ dimensions. Our algorithm can be combined with Wagner's attack, and leads to a sub-exponential solution for any dimension ℓ with best complexity known so far.
When concurrent executions are allowed, our algorithm leads to practical attacks against unforgeability of blind signature schemes such as Schnorr and Okamoto–Schnorr blind signatures, threshold signatures such as GJKR and the original version of FROST, multisignatures such as CoSI and the two-round version of MuSig, partially blind signatures such as Abe–Okamoto, and conditional blind signatures such as ZGP17.

1 Introduction

One of the most fundamental concepts in cryptanalysis is the *birthday paradox*. Roughly, it states that among $O(\sqrt{p})$ random elements from the range $[0, p-1]$ (where p is a prime), there exist two elements a and b such that $a = b$, with high probability. In a seminal work, Wagner gave a generalization of the birthday paradox to ℓ dimensions which asks to find $x_i \in L_i, i \in [0, \ell-1]$ such that $x_0 + \cdots + x_{\ell-1} = 0 \pmod p$, where L_i are lists of random elements.

His work also showed a simple an elegant algorithm to solve the problem in subexponential time $O((\ell + 1) \cdot 2^{\lceil \log p \rceil / (1 + \lfloor \log(\ell+1) \rfloor)})$ and explained how it could be applied to perform cryptanalysis on various schemes. Among the most important applications of Wagner's technique is a subexponential solution to the ROS (Random inhomogeneities in a Overdetermined Solvable system of linear equations) problem [Sch01, FPS20], which is defined as follows. Given a prime number p and access to a random oracle $\mathrm{H}_{\mathrm{ros}}$ with a range in $\mathbb{Z}_p$, the ROS problem (in dimension ℓ) asks to find $(\ell + 1)$ affine functions $\boldsymbol{\rho}_i$, $(\ell + 1)$ bit strings $\mathsf{aux}_i \in \{0, 1\}^*$ (with $i \in [0, \ell]$), and a vector $\mathbf{c} = (c_0, \ldots, c_{\ell-1})$ such that:

$$\mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_i) = \boldsymbol{\rho}_i(\mathbf{c}) \qquad \text{for all } i \in [0, \ell].$$

This problem was originally studied by Schnorr [Sch01] in the context of blind signature schemes. Using a solver for the ROS problem, he showed that the unforgeability of the Schnorr and Okamoto-Schnorr blind signature schemes can be attacked in subexponential time whenever more than $O(\log p)$ signatures are issued concurrently. In this work, we revisit the ROS problem and its applications. We make the following contributions.

- We give the first polynomial time solution to the ROS problem for $\ell > \log p$ dimensions.
- We show how the above solution can be combined with Wagner's techniques to yield an improved subexponential algorithm for dimensions lower than $\log p$. The resulting construction offers a smooth trade-off between the work and the dimension needed to solve the ROS problem. Consequently, it outperforms the runtime of Wagner's algorithm for a broad range of dimensions.
- Finally, we describe how to apply our new attack to an extensive list of schemes. These include: blind signatures [PS00, Sch01], threshold signatures [GJKR07, KG20a], multisignatures [STV+16, MPSW18a], partially blind signatures [AO00], and conditionally blind signatures [ZGP17, GPZZ19], in a concurrent setting with $\ell > \log p$ parallel executions. While our attacks do not contradict the security arguments of those schemes (which are restricted only to sequential or bounded number of executions), it proves that these schemes are unpractical for some real-world applications (cf. Section 7).

1.1 Technical Overview

Let $\mathsf{Pgen}(1^\lambda)$ be a parameter generation algorithm that given as input the security parameter λ in unary form, outputs a prime p of length $\lambda = \lceil \log p \rceil$. In this work, we prove the following main theorem:

Theorem 1 (ROS attack). *If $\ell > \lambda$, then there exists a (probabilistic) adversary that runs in expected polynomial time and solves the ROS problem relative to* Pgen *with dimension ℓ with probability* 1.

Let $B(\mathbf{x}) := \sum_{i=0}^{\lambda-1} 2^i \boldsymbol{\rho}_i(x_i)$ for functions ρ_i where $i \in [0, \lambda-1]$. If we can set $\rho_i(x_i)$ to be the multivariate polynomials that evaluate to 0 at the point c_i^0 and to 1 at the point c_i^1 (for $i \in [0, \ell-1]$), then we can write any value $y \in [0, p-1]$ as $y = B(c_0^{b_0}, \ldots, c_{\ell-1}^{b_{\ell-1}})$, where the b_i values are such that $y = \sum_{i=0}^{\lambda-1} 2^i b_i$. Using this idea, we first define all the functions $\rho_0, \ldots, \rho_{\ell-1}$ along with the corresponding pairs of points c_i^0, c_i^1 that are obtained as $c_i^b := \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, b)$. In a second step, we choose $\boldsymbol{\rho}_\ell(\mathbf{x}) := B(\mathbf{x})$, and query $y := \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_\ell, \mathsf{aux}_\ell)$. Now, we can write $y = \sum_{i=0}^{\lambda-1} 2^i b_i$ which determines a point $c_i^{b_i}$ from every pair. We can output the chosen points in $\mathbf{c}$ along with the vector of affine functions ρ as a solution to the ROS problem. (Note that $\boldsymbol{\rho}_\ell = B(\mathbf{x})$ is also affine.) This attack runs in expected polynomial time (since with small probability, $\mathrm{H}_{\mathrm{ros}}$ produces collisions, in which case steps need to be repeated) and works whenever $\ell > \log p$. This requirement ensures that it is always possible to write any value with ℓ terms in binary representation.

To circumvent the restriction $\ell > \log p$, we prove a second theorem:

Theorem 2 (Generalized ROS attack). *Let $L \geq 0$ be an integer and $w \geq 0$ be a real number. If $\ell \geq \max\{2^w - 1, \lceil 2^w - 1 + \lambda - (w+1) \cdot L \rceil\}$, then there exists a (probabilistic) adversary that runs in expected time $O(2^{w+L})$ and solves the ROS problem relative to* Pgen *and dimension ℓ with probability* 1.

The idea of this attack is to combine the technique from the first attack with the basic subexponential attack of Wagner. Instead of writing y entirely in binary as above, which requires ℓ dimensions, we first find a sum s of 2^w values which include y, but satisfies $|s| \in [0, \frac{p}{2^{(w+1)\cdot L}} - 1] \pmod p$. Note that s can be represented with $\lambda - (w+1) \cdot L$ many bits in binary representation. This approach requires, in total, $\lceil 2^w + \lambda - (w+1) \cdot L - 1 \rceil$ dimensions and 2^{w+L} overall work. As illustrated in Figure 4, this leads to improvements over Wagner's attack relatively quickly as the dimension ℓ of the ROS problem increases. We remark that, while in our first attack we give a concrete probability of failure, our second attack is based on the conjecture that Wagner's algorithm for $\mathbb{Z}_p$ succeeds with constant probability. While we are not aware of any formal analysis of Wagner's algorithm over $\mathbb{Z}_p$, we remark that it is considered a standard cryptanalytic tool [DEF+19]. Our attack can be seen as strictly improving over its (conjectured) performance when applied to solve the ROS problem.

1.2 Impact of the attacks

Any cryptographic construction that bases its security guarantees on the hardness of the ROS problem is affected by our attacks.

Blind signatures. An immediate consequence of our findings is the first polynomial-time attack against Schnorr blind signatures [Sch01] and Okamoto–Schnorr blind signatures [PS00] in the concurrent setting with $\ell > \log p$ parallel executions.[5] Structurally, our attack builds on the one shown by Schnorr [Sch01], who showed that a solver to the ROS problem can be turned into an attacker against one-more unforgeability of blind Schnorr and Okamoto-Schnorr signatures. As a concrete example, the attack in Section 5 breaks one-more unforgeability of blind Schnorr signatures over 256-bit elliptic curves in a few seconds (when implemented in Sage [S+20]), provided that the attacker can open 256 concurrent sessions.

[5] Okamoto–Schnorr signatures are proven secure only for ℓ parallel executions s.t. $Q^\ell/p \ll 1$, where Q is the number of queries to $\mathrm{H}_{\mathrm{ros}}$. Our attack does not contradict their analysis as our attack requires $\ell > \log_2 p > \log_Q p$.

Game $\mathrm{ROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda)$	Oracle $\mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}, \mathsf{aux})$
$p \leftarrow \mathsf{Pgen}(1^\lambda)$	**if** $\mathsf{T}_{\mathrm{ros}}[\boldsymbol{\rho}, \mathsf{aux}] = \perp$ **then**
$\mathsf{T}_{\mathrm{ros}} \coloneqq [\,]$	$\quad \mathsf{T}_{\mathrm{ros}}[\boldsymbol{\rho}, \mathsf{aux}] \leftarrow\$ \mathbb{Z}_p$
$\big((\boldsymbol{\rho}_i, \mathsf{aux}_i)_{i\in[0,\ell]}, (c_j)_{j\in[0,\ell-1]}\big) \leftarrow \mathsf{A}^{\mathrm{H}_{\mathrm{ros}}}(p)$	**return** $\mathsf{T}_{\mathrm{ros}}[\boldsymbol{\rho}, \mathsf{aux}]$
return $\big(\ \forall i \neq j \in [0,\ell], \quad (\boldsymbol{\rho}_i, \mathsf{aux}_i) \neq (\boldsymbol{\rho}_j, \mathsf{aux}_j)$	
$\qquad \wedge\ \forall i \in [0,\ell], \quad \sum_{j=0}^{\ell-1} c_j\rho_{i,j} + \rho_{i,\ell} = \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_i)\big)$	

Fig. 1. The $\mathrm{ROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda)$ game. Above, $\rho_{i,j}$ is the j-th coefficient of the polynomial $\boldsymbol{\rho}_i$, i.e., $\boldsymbol{\rho}_i(\mathbf{x}) = \sum_{j=0}^{\ell-1} \rho_{i,j}x_i + \rho_{i,\ell}$.

Other affected constructions. Our attack can be adapted to an extensive list of schemes which include threshold signatures [GJKR07, KG20a], multisignatures [STV+16, MPSW18a], partially blind signatures [AO00], conditionally blind signatures [ZGP17, GPZZ19], blind anonymous group signatures [CFLW04], blind identity-based signcryption [YW05], and blind signature schemes from bilinear pairings [CHYC05]. We note that some of the previous works claim security only for non-concurrent executions or with a bounded number of executions; therefore, our attacks do not contradict their security claims but render these schemes unsuitable for a broad range of real-world use cases.

Scope of our attacks and countermeasures. Our attacks do not extend to the modified-ROS [FPS20] and the generalized-ROS [HKLN20] problems. The concrete hardness of both problems remains an intriguing open question.

2 Preliminaries

In this work, we assume that logarithm is always base 2. Let again $\mathsf{Pgen}(1^\lambda)$ be a parameter generation algorithm that given as input the security parameter λ in unary outputs a prime p of length $\lambda = \lceil \log p \rceil$. The ROS problem for ℓ dimensions, displayed in Figure 1, is *hard* if no adversary can solve the ROS problem in time polynomial in the security parameter λ. i.e.:

$$\mathsf{Adv}^{\mathrm{ros}}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda) \coloneqq \Pr\left[\mathrm{ROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,.$$

Alternative formulations of ROS. Fuchsbauer et al. [FPS20, Fig. 7] present a variant of $\mathrm{ROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda)$ with linear instead of affine functions $\boldsymbol{\rho}_i$ (i.e., where $\rho_{i,\ell} = 0$). Hauck et al. [HKL19, Fig. 3] allow only for linear functions, and do not allow for auxiliary information aux within $\mathrm{H}_{\mathrm{ros}}$ (i.e., where $\mathsf{aux}_i = \perp$).[6] These formulations are all equivalent.

First, any adversary A for an ROS with affine functions as per Figure 1 can be reduced to an adversary B for ROS with linear functions as per [FPS20]: B runs A and for every query of the form $((\rho_{i,0}, \dots, \rho_{i,\ell}), \mathsf{aux}_i)$ to the oracle $\mathrm{H}_{\mathrm{ros}}$ (made by A), it returns $\mathrm{H}_{\mathrm{ros}}((\rho_{i,0}, \dots, \rho_{i,\ell-1}), (\rho_{i,\ell} \| \mathsf{aux}_i)) - \rho_{i,\ell}$. Finally, B modifies accordingly the solution output by A by concatenating $\rho_{i,\ell}$ to the corresponding aux_i.

Second, any adversary A for an ROS with linear functions can be reduced to an adversary B for ROS with linear functions and without auxiliary information as per [HKL19]. We assume without loss of generality that A never makes twice the same query. Then B runs A and for every query of the form $((\rho_{i,0}, \dots, \rho_{i,\ell-1}, 0), \mathsf{aux}_i)$ to the oracle (made by A), it picks a random scalar $r \in \mathbb{Z}_p^*$ and returns $\mathrm{H}_{\mathrm{ros}}((r \cdot \rho_{i,0}, \dots, r \cdot \rho_{i,\ell-1}), \perp) \cdot r^{-1} \bmod p$. When A outputs a solution $(\boldsymbol{\rho}_i, \mathsf{aux}_i)_{i\in[0,\ell]}, (c_j)_{j\in[0,\ell-1]}$, B outputs $(r \cdot \boldsymbol{\rho}_i)_{i\in[0,\ell]}, (c_j)_{j\in[0,\ell-1]}$. The simulation of the oracle $\mathrm{H}_{\mathrm{ros}}$ is perfect unless there is a collision in the scalar r, which happens with negligible probability in λ.

[6] Our attacks only apply to the case where the scalar set $\mathcal{S}$ is a finite field.

3 Attack

In this section, we prove Theorem 1. We abuse notation and $\boldsymbol{\rho}_i$ denotes both the vector $\boldsymbol{\rho}_i = (\rho_{i,0}, \ldots, \rho_{i,\ell}) \in \mathbb{Z}_p^{\ell+1}$ and the corresponding affine function $\boldsymbol{\rho}_i(\mathbf{x}) = \sum_{j=0}^{\ell-1} \rho_{i,j} \cdot x_j + \rho_{i,\ell}$ (where $\mathbf{x} = (x_0, \ldots, x_\ell)$).

Proof (of Theorem 1). We construct an adversary for $\mathrm{ROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda)$, where $\ell > \log p$. Recall that to simplify the description of the attack, we use a polynomial formulation of ROS, i.e., we represent vectors $\boldsymbol{\rho}_i = (\rho_{i,0}, \ldots, \rho_{i,\ell})$ as linear multivariate polynomials in $\mathbb{Z}_p[x_0, \ldots, x_{\ell-1}]$:

$$\boldsymbol{\rho}_i(x_0, \ldots, x_{\ell-1}) = \rho_{i,0}x_0 + \cdots + \rho_{i,\ell-1}x_{\ell-1} + \rho_{i,\ell} \ . \tag{1}$$

The goal for the adversary A is to output $(\boldsymbol{\rho}_i, \mathsf{aux}_i)_{i \in [0,\ell]}$ and $\mathbf{c} = (c_0, \ldots, c_{\ell-1})$ such that:

$$\boldsymbol{\rho}_i(\mathbf{c}) = \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_i) \qquad \text{for all } i \in [0, \ell].$$

Define:

$$\boldsymbol{\rho}_i \coloneqq x_i \qquad \text{for } i = 0, \ldots, \ell - 1,$$

and find two strings aux_i^0 and aux_i^1 such that $c_i^b \coloneqq \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_b)$ are different for $b = 0$ and $b = 1$.[7] Then, let:

$$x_i' \coloneqq \frac{x_i - c_i^0}{c_i^1 - c_i^0}$$

for all $i = 0, \ldots, \ell - 1$. We remark that, if $x_i = c_i^b$, then $x_i' = b$ (for $b = 0, 1$). Define $\boldsymbol{\rho}_\ell \coloneqq \sum_{i=0}^{\ell-1} 2^i x_i'$, and query $y \coloneqq \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_\ell, \perp)$. Finally, write y in binary as:

$$y = \sum_{i=0}^{\ell-1} 2^i b_i \pmod p.$$

(As $2^\ell > p$, it is possible to write y this way, and this implicitly defines the b_i's.) The adversary A outputs: $(\boldsymbol{\rho}_0, \mathsf{aux}_0^{b_0}), \ldots, (\boldsymbol{\rho}_{\ell-1}, \mathsf{aux}_{\ell-1}^{b_{\ell-1}}), (\boldsymbol{\rho}_\ell, \perp)$ and $\mathbf{c} \coloneqq (c_0^{b_0}, \ldots, c_{\ell-1}^{b_{\ell-1}})$. We have indeed that, for $i \in [0, \ell - 1]$, $\boldsymbol{\rho}_i(\mathbf{c}) = c_i^{b_i} = \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_i^{b_i})$ and:

$$\boldsymbol{\rho}_\ell(\mathbf{c}) = \sum_{i=0}^{\ell-1} 2^i x_i'(\mathbf{c}) = \sum_{i=0}^{\ell-1} 2^i b_i = y = \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_\ell, \perp) \ .$$

□

Remark 1. In [FPS20, Sec. 5], Fuchsbauer, Plouviez, and Seurin proposed a variant of ROS, called modified ROS. The attack above does not apply to modified ROS.

4 Generalized attack

We present a combination of Wagner's subexponential k-list attack and the polynomial time attack from Section 3. This combined attack yields a subexponentially efficient algorithm against ROS which requires fewer dimensions than the attack in the previous section (i.e., less than $\lambda = \lceil \log p \rceil$). However, for some practical cases, the attack significantly outperforms Wagner's attack in terms of work, for the same number of dimensions. At a very high level, our attack works as follows. We set $k_1 = 2^w - 1$, $k_2 = \max(0, \lceil \lambda - (w+1) \cdot L \rceil)$, and the dimension $\ell = k_1 + k_2$, for some integer w and some real number $L > 0$.

First, we use a generalization of Wagner's algorithm to find a "small" sum $s = y_{k_2}^* + \cdots + y_\ell^*$ of k_1 values $y_i^* \coloneqq -\mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_i)$, where the polynomials $\boldsymbol{\rho}_i(\mathbf{x})$ are chosen to make the second step of the attack

[7] This step is the reason why the algorithm is expected polynomial time instead of polynomial time. Note that since $\mathsf{aux} \in \{0, 1\}^*$, there will always be two values $\mathsf{aux}_i^0, \mathsf{aux}_i^1 \in \{0, 1\}^*$ so that $c_i^0 \neq c_i^1$.

work.[8] As we describe below, we can obtain that $|s| < 2^{k_2-1}$ using $O(2^{w+L})$ hash queries and space $O(w2^L)$. Then, we use the technique from the previous section in order to represent the sum s as a binary sum of at most k_2 terms. Finally, we subtract the $k_1 - 1$ terms $y^*_{k_2}, ..., y^*_{k_2+k_1-1} = y^*_{\ell-1}$ to extract the term y^*_ℓ. This solves the ROS problem. The attack runs in overall time $O(2^{w+L})$, space $O(w2^L)$, and requires $\ell = \max(2^w - 1, \lceil 2^w - 1 + \lambda - (w+1) \cdot L \rceil)$ dimensions.

We remark that the attack is a generalization of both Wagner's attack and our polynomial-time attack from Section 3. Wagner's attack corresponds to the case where $L = \lambda/(w+1)$ and $\ell = 2^w - 1$. Our polynomial-time attack corresponds to the case $w = 0$, $L = 0$, $\ell = \lambda$.

Examples. For a prime p of $\lambda = 256$ bits, a concrete example yields $w = 5, L = 15$, i.e., $\ell = 32 + 256 - 6 \cdot 15 - 1 = 197$ dimensions and time roughly 2^{20} and space roughly $5 \cdot 2^{15}$ (elements of $\mathbb{Z}_p$). On the other hand, Wagner's algorithm for 197 dimensions requires time roughly $2^{\lfloor \log 197 \rfloor} \cdot 2^{\frac{256}{\lfloor \log 197 \rfloor + 1}} = 2^7 \cdot 2^{32} = 2^{39}$ and space roughly $\lfloor \log 197 \rfloor \cdot 2^{\frac{256}{\lfloor \log 197 \rfloor + 1}} = 7 \cdot 2^{32}$.

For a 512 bit modulus, a concrete example yields $w = 6, L = 46$, i.e., $\ell = 64 + 512 - 7 \cdot 46 - 1 = 253$ dimensions and time roughly 2^{53} and space roughly $6 \cdot 2^{46}$. Wagner's algorithm for 254 dimensions requires time roughly $2^{\lfloor \log 254 \rfloor} \cdot 2^{\frac{512}{\lfloor \log 255 \rfloor + 1}} = 2^7 \cdot 2^{64} = 2^{71}$ and space roughly $\lfloor \log 254 \rfloor \cdot 2^{\frac{512}{\lfloor \log 255 \rfloor + 1}} = 7 \cdot 2^{64}$.[9]

4.1 Generalized k-List Algorithm

In this section, we write elements $\mathbb{Z}_p$ as signed integers in $[-\frac{p-1}{2}, \frac{p-1}{2}]$. Let w and L be two positive integers. We define the following integer intervals:

$$I_i \coloneqq \left[- \left\lfloor \frac{p-1}{2^{(w-i)\cdot L+1}} \right\rfloor, \left\lfloor \frac{p-1}{2^{(w-i)\cdot L+1}} \right\rfloor \right] .$$

Remark that $\mathbb{Z}_p = I_w$.

We now describe the k-list algorithm, which is the core of the Wagner's algorithm. We generalize it to match our needs and to output elements that sum to something in I_{-1} rather than to exactly 0. (This essentially corresponds to executing Wagner's attack as usual, but stopping earlier.) The algorithm is defined relative to random oracle $\mathrm{H}_{\mathrm{ros}}$. It takes as input $(w, L, \boldsymbol{\rho}_1, \dots, \boldsymbol{\rho}_k)$ and outputs $(\mathsf{aux}^*_1, \dots, \mathsf{aux}^*_k)$ with $k = 2^w$ such that:

$$s \coloneqq y^*_1 + \dots + y^*_k \in I_{-1} \qquad \text{where } y_i \coloneqq \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}^*_i) .$$

The high-level idea of the algorithm is to use $2^{w+1} - 1$ lists of about 2^L values organized as a tree, as depicted in Fig. 2, and to ensure that lists $\mathfrak{L}^w_i$ at level i contains elements from the set I_i.

- **Setup/Leaves:** $\mathsf{k\text{-}List}$ fills the lists $\mathfrak{L}^w_i$ in the leaves with 2^L points of the form $\mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}) \in \mathbb{Z}_p = I_w$, for $\mathsf{aux} \in [1, 2^L]$.
- **Collisions/Join:** The algorithm now proceeds to find collisions in levels from w to 1. At level i, process the 2^{i-1} pairs of lists $(\mathfrak{L}^i_1, \mathfrak{L}^i_2), \dots, (\mathfrak{L}_{2^i-1}, \mathfrak{L}_{2^i})$ into 2^{i-1} lists $\mathfrak{L}^{i-1}_1, \dots, \mathfrak{L}^{i-1}_{2^{w-1}}$ as follows:

  $$\mathfrak{L}^{i-1}_j \coloneqq \{a + b \quad : \quad a \in \mathfrak{L}^i_{2j-1},\ b \in \mathfrak{L}^i_{2j},\ a + b \in I_i\} .$$

  (Remember that $a, b \in \mathbb{Z}_p$ and $a + b$ is computed modulo p.) Moreover, we implicitly assume that the algorithm stores back pointers to a and b s.t. they can efficiently be recovered at a later point.
- **Output:** Let $\mathfrak{L}^0 = \mathfrak{L}^0_1$ denote the (only) list created at level 1. The algorithm finds an element $s \in \mathfrak{L}^0$ such that $s \in I_{-1}$. If no such element exists, it returns $\perp$. Otherwise, it recovers $k = 2^w$ strings $\mathsf{aux}^*_1, \dots, \mathsf{aux}^*_k$ such that $y^*_i = \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}^*_i) \in \mathfrak{L}^w_i$ and $s = y^*_1 + \dots + y^*_k$. It returns $(\mathsf{aux}^*_1, \dots, \mathsf{aux}^*_k)$.

We formally write the algorithm $\mathsf{k\text{-}List}$ in Figure 3.

[8] In the actual attack, part of the second step is executed before to allow to choose these polynomials properly.

[9] Indeed, when considering the exact values of the constants in the asymptotics, the actual complexity of Wagner's attack is $2^{\lfloor \log(\ell+1) \rfloor} \cdot 2^{\frac{p}{\lfloor \ell+1 \rfloor + 1}}$.

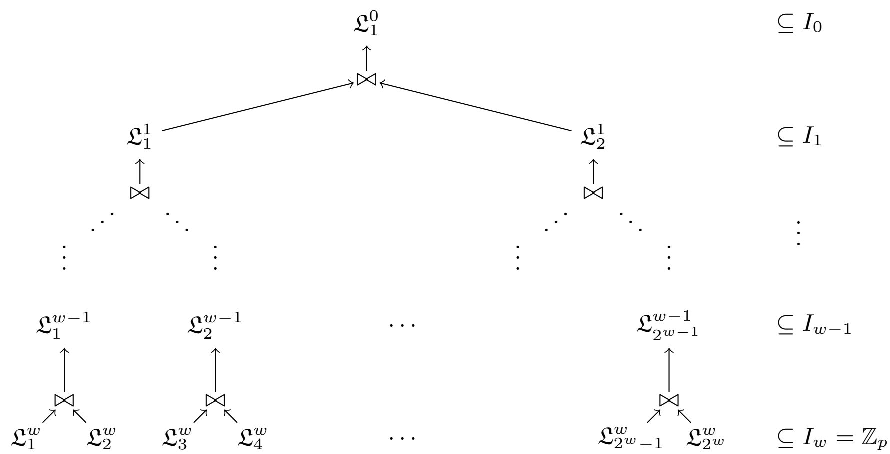

Fig. 2. Tree of lists for the k-list algorithm ($\bowtie$ represents the join operation in the algorithm; the sets in the right handside are the sets to which the elements of the lists of a given level belong).

Correctness. We do not prove correctness of $\mathsf{k\text{-}List}$ in this work, since our algorithm's correctness is implied by the correctness of Wagner's original algorithm. More precisely, our algorithm performs identical steps as Wagner's, but stops upon finding a sum of values with a suitably small absolute value, i.e., one that falls into I_0. On the other hand, Wagner's algorithm keeps continuing with more levels until it finds values who sum to 0. However, we remark that we are not aware of a formal analysis of Wagner's algorithm for values in $\mathbb{Z}_p$. The work of Minder and Sinclair [MS09] analyses the case of finding a weighted sum of *vectors* of $\mathbb{Z}_p$ values that sum to zero in each component, but uses a different technique from the one presented in Wagner's paper (and used here). Our attack can be seen as working under the assumption that Wagner's algorithm works correctly, i.e., has constant failure probability (see below). We can repeat the attack until it succeeds, which makes the resulting algorithm expected polynomial time. Formally analyzing the failure probability of Wagner's algorithm over $\mathbb{Z}_p$ remains an important open problem.

Complexity. Overall, the algorithm runs in time $O(2^{w+L})$ and is conjectured to succeed with constant probability. (As described [Wag02], this running time is made possible using an optimized join operation such as Hash Join or Merge Join). The algorithm uses space $O(2^{w+L})$, but by evaluating the collisions/joins in postfix order (in the tree), this can be reduced to $O(w2^L)$.

4.2 Combined Attack

We now prove Theorem 2.

Proof (of Theorem 2). Recall that $k_1 = 2^w - 1$ and $k_2 = \max(0, \lceil \lambda - (w+1) \cdot L \rceil)$. Set $\ell = k_1 + k_2$. For all $i \in [0, \ell - 1]$, define:

$$\boldsymbol{\rho}_i \coloneqq x_i \,,$$

and find two strings aux_i^0 and aux_i^1 with different hash values $c_i^0 = \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_i^0)$ and $c_i^1 = \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_i^1)$. Then, let:

$$x_i' \coloneqq \frac{x_i - c_i^0}{c_i^1 - c_i^0}$$

Algorithm $\mathsf{k\text{-}List}^{\mathrm{H_{ros}}}(w, L, \boldsymbol{\rho}_1, \dots, \boldsymbol{\rho}_{2^w})$

```
// Setup
```
$L_i^w \coloneqq \{\mathrm{H_{ros}}(\boldsymbol{\rho}_i, \mathsf{aux})\}_{\mathsf{aux}\in[1,2^L]}$ **for** $i \in [1, 2^w]$

// Collisions

for $i = w$ **downto** 1:

  for $j \in [1, 2^{i-1}]$:

    $\mathfrak{L}_j^{i-1} = \{a + b : a \in \mathfrak{L}_{2j-1}^i,\ b \in \mathfrak{L}_{2j}^i,\ a + b \in I_i\}$

// Output

look for an element $s = y_1^* + \cdots + y_k^* \in \mathfrak{L}^0 \cap I_{-1}$

if such an element does not exists **then** **return** $\perp$

return $(\mathsf{aux}_1^*, \dots, \mathsf{aux}_k^*)$ such that $y_i^* = \mathrm{H_{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_i^*)$

Fig. 3. The k-list algorithm.

for all $i \in [0, k_2 - 1]$. We remark that, if $x_i = c_i^b$, then $x_i' = b$ (for $b = 0, 1$). Define:

$$\boldsymbol{\rho}_\ell \coloneqq \sum_{i=0}^{k_2-1} 2^i x_i' - \frac{p-1}{2^{(w+1)\cdot L+1}} - \sum_{i=k_2}^{k_1+k_2-1} x_i \ .$$

Run $(\mathsf{aux}_{k_2}, \dots, \mathsf{aux}_\ell) \coloneqq \mathsf{k\text{-}List}^{\mathrm{H_{ros}}}(w, L, \boldsymbol{\rho}_{k_2}, \dots, \boldsymbol{\rho}_\ell)$ (where $k = k_1 + 1 = 2^w$) and define for $i \in [k_2, \ell]$:

$$y_i^* \coloneqq \mathrm{H_{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_i^*) \ ,$$

and $c_i \coloneqq y_i^*$ for $i \in [k_2, \ell - 1]$. Set:

$$s \coloneqq \sum_{i=k_2}^{\ell} y_i^* \quad \in I_{-1} = [-\left\lfloor \frac{p-1}{2^{(w+1)\cdot L+1}} \right\rfloor, \left\lfloor \frac{p-1}{2^{(w+1)\cdot L+1}} \right\rfloor] \ . \tag{2}$$

Write $s + \lfloor (p-1)/2^{(w+1)\cdot L+1} \rfloor$ in binary as:

$$s + \left\lfloor \frac{p-1}{2^{(w+1)\cdot L+1}} \right\rfloor = \sum_{i=0}^{k_2-1} 2^i b_i \quad \in [0, \left\lfloor \frac{p-1}{2^{(w+1)\cdot L}} \right\rfloor] \ , \tag{3}$$

which is possible since $p < 2^\lambda$, $k_2 = \lambda - (w+1) \cdot L$, hence $(p-1)/2^{(w+1)\cdot L} < 2^{k_2}$. Define:

$$\mathsf{aux}_i = \begin{cases} \mathsf{aux}_i^{b_i} & \text{for } i \in [0, k_2 - 1] \ , \\ \mathsf{aux}_i^* & \text{for } i \in [k_2, k_1 + k_2] \text{ from } \mathsf{k\text{-}List}. \end{cases}$$

A outputs: $(\boldsymbol{\rho}_0, \mathsf{aux}_0), \dots, (\boldsymbol{\rho}_\ell, \mathsf{aux}_\ell)$ and:

$$\mathbf{c} \coloneqq (c_0^{b_0}, \dots, c_{k_2}^{b_{k_2}}, c_{k_2+1}, \dots c_{k_2+k_1-1}) \ .$$

We have indeed that:

$$\boldsymbol{\rho}_i(\mathbf{c}) = c_i = \begin{cases} c_i^{b_i} = \mathrm{H_{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_i^{b_i}) & \text{for } i \in [0, k_2 - 1] \ , \\ y_i^* = \mathrm{H_{ros}}(\boldsymbol{\rho}_i, \mathsf{aux}_i^*) & \text{for } i \in [k_2, k_1 + k_2 - 1] \ . \end{cases}$$

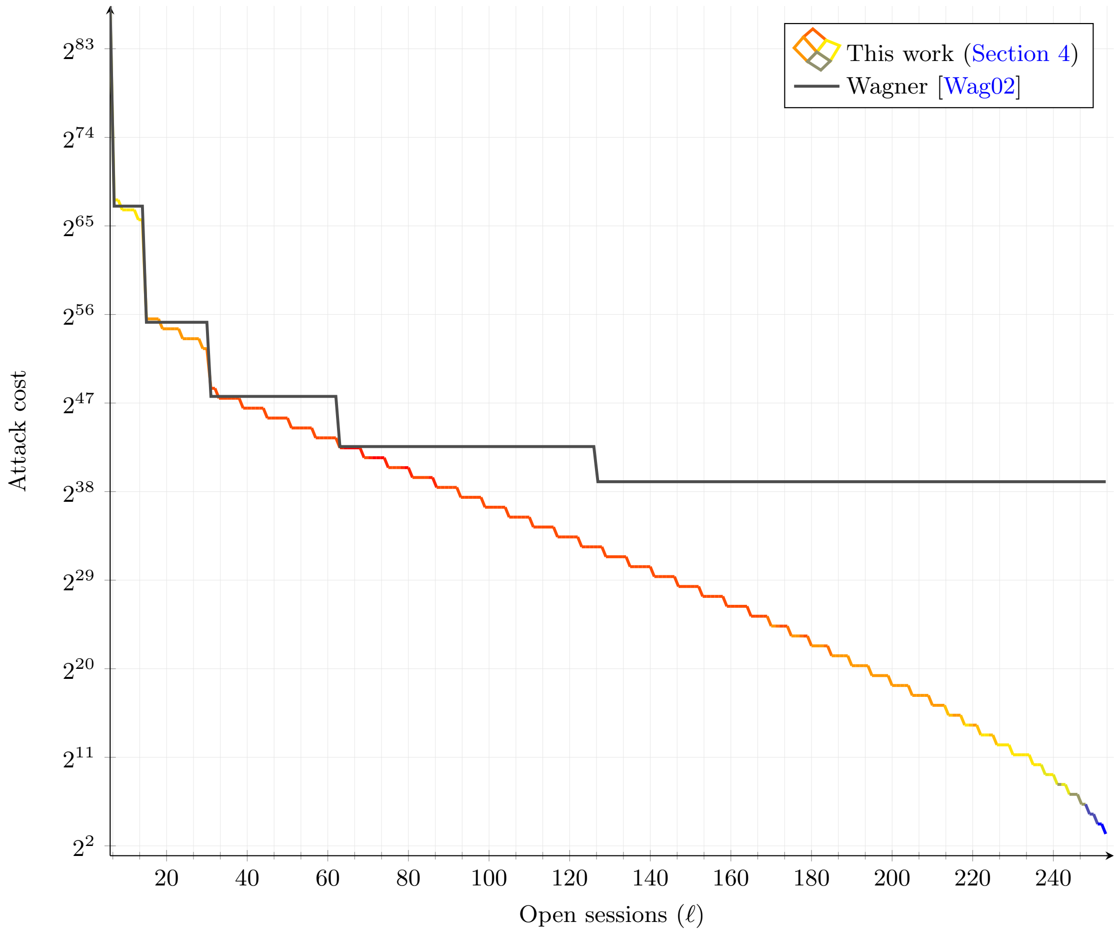

Fig. 4. Concrete cost of our combined attack compared to Wagner's [Wag02] for $\lambda = 256$ and $\ell < 256$. The color key indicates the different values of w used to estimate the cost. For $\ell \geq 256$, the attack of Section 3 applies.

and:

$$\begin{aligned}
\boldsymbol{\rho}_\ell(\mathbf{c}) &= \sum_{i=0}^{k_2-1} 2^i x'_i(\mathbf{c}) - \left\lfloor \frac{p-1}{2^{(w+1)\cdot L+1}} \right\rfloor - \sum_{i=k_2}^{k_1+k_2-1} x_i(\mathbf{c}) \\
&= \sum_{i=0}^{k_2-1} 2^i b_i - \left\lfloor \frac{p-1}{2^{(w+1)\cdot L+1}} \right\rfloor - \sum_{i=k_2}^{k_1+k_2-1} y_i^* \\
&= s - \sum_{i=k_2}^{k_1+k_2-1} y_i^* = y_{k_2+k_1}^* = \mathrm{H}_{\mathrm{ros}}(\boldsymbol{\rho}_\ell, \mathsf{aux}_\ell^*) \ ,
\end{aligned}$$

where the third equality comes from Equation (3) while the fourth equality comes from Equation (2). The attack requires $k_1 + k_2 = \max\{2^w - 1, \lceil 2^w - 1 + \lambda - (w+1) \cdot L \rceil\}$ dimensions, runs in time $O(2^{w+L})$, and in space $O(w2^L)$. $\square$

5 Affected blind signatures

For simplicity and clarity of exposition, we implement only the attack presented in Section 3. Our attack can be easily adapted for the one presented in Section 4.

Throughout the remaining of this manuscript, we will assume the existence of a group generator algorithm $\mathsf{GrGen}(1^\lambda)$ that, given as input the security parameter in unary form outputs the description $\Gamma = (\mathbb{G}, p, G, H)$ of a group $\mathbb{G}$ of prime order p. Similarly to Section 2, we assume that the prime p is of length λ. We use additive notation for the group law.

5.1 Schnorr blind signatures

A Schnorr blind signature [Sch01, FPS20] for a message $m \in \{0,1\}^*$ consists of a pair $(R, s) \in \mathbb{G} \times \mathbb{Z}_p$ such that $sG - cX = R$, where $c \coloneqq \mathsf{H}(R, m)$ and $X \in \mathbb{G}$ is the verification key. A formal description of the protocol can be found in [FPS20, Fig. 6], using the same notation employed here.

We construct a probabilistic (expected) polynomial-time adversary A that is able to produce $\ell + 1$ signatures after opening $\ell \geq \lceil \log p \rceil = \lambda$ parallel sessions. A selects a message $m_\ell \in \{0,1\}^*$ for which a signature will be forged. It opens ℓ parallel sessions, querying $\textsc{Sign}_0()$ and receiving $\mathbf{R} = (R_0, \ldots, R_{\ell-1}) \in \mathbb{G}^\ell$. Let m_i^b be a random message and $c_i^b \coloneqq \mathsf{H}(R_i, m_i^b)$ for $i \in [0, \ell-1]$ and $b \in \{0,1\}$. If $c_i^0 = c_i^1$, two different messages m_i^0 and m_i^1 are chosen until $c_i^0 \neq c_i^1$. Define $\boldsymbol{\rho}_\ell \coloneqq \sum_i 2^i x_i'$ as per Section 3, that is:

$$\boldsymbol{\rho}_\ell(x_0, \ldots, x_{\ell-1}) \coloneqq \sum_{i=0}^{\ell-1} 2^i \cdot \frac{x_i - c_i^0}{c_i^1 - c_i^0} = \sum_{i=0}^{\ell-1} \rho_{\ell,i} x_i + \rho_{\ell,\ell} \quad . \tag{4}$$

Let $R_\ell \coloneqq \boldsymbol{\rho}_\ell(\mathbf{R}) - \rho_{\ell,\ell} \cdot X$. Define $c_\ell \coloneqq \mathsf{H}(R_\ell, m_\ell) = \sum_{i=0}^{\ell-1} 2^i b_i$ and let $\mathbf{c} = (c_0^{b_0}, \ldots, c_{\ell-1}^{b_{\ell-1}})$. Complete the ℓ opened sessions querying $\textsc{Sign}_1(i, c_i^{b_i})$, for $i \in [0, \ell-1]$. The adversary thus obtains responses $\mathbf{s} \coloneqq (s_0, \ldots, s_{\ell-1}) \in \mathbb{Z}_p^\ell$ satisfying:

$$s_i G - c_i^{b_i} X = R_i, \qquad \text{for } i \in [0, \ell-1].$$

Let $s_\ell \coloneqq \boldsymbol{\rho}_\ell(\mathbf{s})$. Then $(m_\ell, (R_\ell, s_\ell))$ is a valid forgery. In fact, by perfect correctness of Schnorr blind signatures, we have:

$$\begin{aligned} R_\ell = \boldsymbol{\rho}_\ell(\mathbf{R}) - \rho_{\ell,\ell} X &= \sum_{i=0}^{\ell-1} \rho_{\ell,i} \cdot R_i + \rho_{\ell,\ell} \cdot (G - X) \\ &= \sum_{i=0}^{\ell-1} \rho_{\ell,i} \cdot (s_i G - c_i^{b_i} X) + \rho_{\ell,\ell} \cdot (G - X) \\ &= \boldsymbol{\rho}_\ell(\mathbf{s}) \cdot G - \boldsymbol{\rho}_\ell(\mathbf{c}) \cdot X \\ &= s_\ell G - c_\ell X, \end{aligned}$$

where $c_\ell = \mathsf{H}(R_\ell, m_\ell) = \boldsymbol{\rho}_\ell(\mathbf{c})$ by Equation (4). Let $m_i \coloneqq m_i^{b_i}$ for $i \in [0, \ell-1]$. The adversary outputs $(m_i, (R_i, s_i))$ for $i \in [0, \ell]$.

Remark 2. The attack does not apply to the clause blind Schnorr signature scheme [FPS20, Sec. 5], which relies on the modified ROS problem.

5.2 Okamoto–Schnorr blind signatures

An Okamoto–Schnorr blind signature [PS00] for a message m consists of a tuple $(R, s, t) \in \mathbb{G} \times \mathbb{Z}_p^2$ such that $sG + tH - cX = R$, where $c \coloneqq \mathsf{H}(R, m)$. The attack of the previous section directly extends to Okamoto–Schnorr signatures: A operates exactly as before until Equation (4). Then, the forgery is constructed as:

$$\big(R_\ell \coloneqq \boldsymbol{\rho}_\ell(\mathbf{R}) + \rho_{\ell,\ell} H - \rho_{\ell,\ell} X, \quad s_\ell \coloneqq \boldsymbol{\rho}_\ell(\mathbf{s}), \quad t_\ell \coloneqq \boldsymbol{\rho}_\ell(\mathbf{t})\big).$$

We stress again that this does not contradict the security analysis of Stern and Pointcheval [PS00], whose security was reduced to $\mathrm{DLOG}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ for a $\mathsf{polylog}(\lambda)$ number of queries.

6 Other constructions affected

In this section, we overview how the attacks presented in Sections 3 and 4 apply to a number of other cryptographic primitives. To simplify exposition, we focus on adapting the attack of Section 3. We note that, in some cases (e.g., multi-signatures), we break the security claims of the papers, while for other primitives (e.g., threshold signatures), our attack illustrates the tightness of the security theorems, which assume either non-concurrent setting, or up to a logarithmic number of concurrent executions.

6.1 Multi-Signatures

A multi-signature scheme allows a group of signers $S_1, \dots, S_n$, each having their own key pair $(\mathsf{pk}_j, \mathsf{sk}_j)$, to collaboratively sign a message m. The resulting signature can be verified given the message and the set of public keys of all signers.

CoSi

CoSi is a multi-signature scheme introduced by Syta et al. [STV+16], that features a two-round signing protocol. The signers are organized in a tree structure, where S_1 is the root of the tree. A signature for a message $m \in \{0,1\}^*$ consists of a pair $(c, s) \in \mathbb{Z}_p^2$ such that $c = \mathsf{H}(sG - c \cdot \mathsf{pk}, m)$, where $\mathsf{pk} = \sum_{j=1}^{n} \mathsf{pk}_j \in \mathbb{G}$ is the aggregated verification key. A formal description of the protocol can be found in [DEF+19, Sec. 2.5]; we use the same notation, except that we employ additive notation xG instead of multiplicative notation g^x.

Attack. We present an attack for a two-node tree where the attacker controls the root S_1. The attack can easily be extended to other settings, similarly to [DEF+19, Sec. 4.2]. Our attack allows the signer S_1 to forge one signature, for an arbitrary message $m_\ell \in \{0,1\}^*$, after performing $\ell \geq \lceil \log p \rceil = \lambda$ interactions with the honest signer S_2. Recall that $\mathsf{pk} = \mathsf{pk}_1 + \mathsf{pk}_2$ where $\mathsf{pk}_i = \mathsf{sk}_i G$. The signing protocol proceeds as follows. First, S_1 obtains a commitment $t_2 = r_2 G$ from S_2, and computes $\bar{t} = t_1 = r_1 G + t_2$ for a random r_1. Then, S_1 computes the challenge $c = \mathsf{H}(\bar{t}, m)$, and sends $(\bar{t}, c)$ to S_2. Next, S_2 returns $s_2 \coloneqq r_2 + c \cdot \mathsf{sk}_2$. Finally, S_1 computes $s \coloneqq s_2 + r_1 + c \cdot \mathsf{sk}_1$ and outputs the signature (c, s) for the message m.

The attack proceeds as follows. S_1 opens ℓ parallel sessions with ℓ arbitrary distinct messages $m_0, \dots, m_{\ell-1} \in \{0,1\}^*$. For each session, S_1 gets the commitments $t_i = r_i G$ from S_2 at the end of the first round of signing. Now, it samples two random values $r_{i,0}, r_{i,1}$ for each $i \in [0, \ell-1]$, defines $\bar{t}_i^0 = r_{i,0}G + t_i$ and $\bar{t}_i^1 = r_{i,1}G + t_i$, and computes $c_i^b = \mathsf{H}(\bar{t}_i^b, m_i)$. (As usual, if $c_i^0 = c_i^1$, S_1 samples again $r_{i,0}$ and $r_{i,1}$ until $c_i^0 \neq c_i^1$.) S_1 then defines the polynomial $\boldsymbol{\rho} \coloneqq \sum_{i=0}^{\ell-1} 2^i x_i/(c_i^1 - c_i^0)$, computes $t_\ell \coloneqq \boldsymbol{\rho}(t_0, \dots, t_{\ell-1})$ and $c_\ell \coloneqq \mathsf{H}(t_\ell, m_\ell)$. S_1 computes $d_\ell = c_\ell - \boldsymbol{\rho}(c_0^0, \dots, c_{\ell-1}^0)$ and writes this value in binary as $d_\ell = \sum_{i=0}^{\ell-1} 2^i b_i$. It then closes the ℓ sessions by using $\bar{t}_i = \bar{t}_i^{b_i}$ and $c_i = c_i^{b_i}$. At the last step of the signing sessions, S_1 obtains values $s_i = r_i + c_i \cdot \mathsf{sk}_2$ from S_2, and closes the sessions honestly using r_{i,b_i}. Finally, S_1 concludes its forgery by defining $s_\ell \coloneqq \boldsymbol{\rho}(\mathbf{s}) + c_\ell \cdot \mathsf{sk}_1$: the pair (c_ℓ, s_ℓ) is a valid signature for m_ℓ. In fact:

$$\begin{aligned}
s_\ell G - c_\ell \cdot \mathsf{pk} &= (\boldsymbol{\rho}(\mathbf{s}) + c_\ell \cdot \mathsf{sk}_1)G - c_\ell \cdot \mathsf{pk} \\
&= \sum_{i=0}^{\ell-1} \frac{2^i s_i}{c_i^1 - c_i^0} G - c_\ell \cdot \mathsf{pk}_2 \\
&= \sum_{i=0}^{\ell-1} \frac{2^i (r_i + c_i^{b_i} \cdot \mathsf{sk}_2)}{c_i^1 - c_i^0} G - c_\ell \cdot \mathsf{pk}_2
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=0}^{\ell-1} \frac{2^i r_i}{c_i^1 - c_i^0} G + \left(\sum_{i=0}^{\ell-1} \frac{2^i c_i^{b_i}}{c_i^1 - c_i^0} - c_\ell \right) \cdot \mathsf{pk}_2 \\
&= \sum_{i=0}^{\ell-1} \frac{2^i t_i}{c_i^1 - c_i^0} + \left(\sum_{i=0}^{\ell-1} 2^i b_i + \sum_{i=0}^{\ell-1} \frac{2^i c_i^0}{c_i^1 - c_i^0} - c_\ell \right) \cdot \mathsf{pk}_2 \\
&= \sum_{i=0}^{\ell-1} \frac{2^i t_i}{c_i^1 - c_i^0} + \underbrace{\left(\sum_{i=0}^{\ell-1} 2^i b_i + \boldsymbol{\rho}(c_0^0, \ldots, c_{\ell-1}^0) - c_\ell \right)}_{= d_\ell - d_\ell = 0} \cdot \mathsf{pk}_2 \\
&= \boldsymbol{\rho}(t_0, \ldots, t_{\ell-1}) = t_\ell \,,
\end{aligned}
$$

and $c_\ell = \mathsf{H}(t_\ell, m_\ell)$ by definition.

Two-round MuSig.

As in [DEF+19], the above technique (with some minor modifications) can be applied to the two-round MuSig as initially proposed by Maxwell et al. [MPSW18a], as the main difference between CoSi and two-round MuSig is in how the public key is aggregated in order to avoid rogue-key attacks. Our attack does not apply to the updated MuSig that uses a 3-round signing algorithm [MPSW18b].

6.2 Threshold signatures

A (t, n)-threshold signature scheme assumes that the secret signing key is split among n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ in a way that allows any subset of at least t out of the n parties to produce a valid signature. As long as the adversary corrupts less than the threshold number of parties, it is not possible to forge signatures or learn any information about the signing key.

GJKR07

Gennaro, Jarecki, Krawczyk, Rabin proposed a threshold signature scheme based on Pedersen's distributed key generation (DKG) protocol in [GJKR07, Section 5.2]. At a very high level, Pedersen's DKG protocol allows to generate a random group element $X = \chi G$ so that its discrete logarithm χ is shared both additively and according to Feldman secret sharing [Fel87] scheme, between a set of "qualified" parties. For the attack we present below, all parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ (included the ones that are controlled by the adversary) will remain qualified.[10] We denote by χ_j the additive share of party P_j. We have $\chi = \sum_{j=1}^n \chi_j$. Importantly for the attack, the adversary controlling for example P_1, can see all the group elements $\chi_2 G, \ldots, \chi_n G$ and then can choose its value χ_1. This is due to the way the Feldman secret sharing is performed.

In the threshold signature scheme of Gennaro et al. [GJKR07], the parties execute a distributed key generation procedure to produce a verification key $\mathsf{pk} \coloneqq \mathsf{sk} \cdot G \in \mathbb{G}$, where the secret key sk is additively shared between the parties: each party P_j has an additive share sk_j, so that $\mathsf{sk} = \sum_{j=1}^n \mathsf{sk}_j$. A signature (R, s) for a message $m \in \{0,1\}^*$ is generated as follows. The participants run once again the distributed key generation protocol to produce a commitment $t = rG \in \mathbb{G}$, where r is additively shared between the parties: each party P_j has a share r_j, so that $r = \sum_{j=1}^n r_j$. Then, each party computes a share of the response:

$$
s_j = r_j + c \cdot \mathsf{sk}_j, \qquad \text{where } c \coloneqq \mathsf{H}(t, m). \tag{5}
$$

Let $s \coloneqq \sum_{j=1}^n s_j$. Then (c, s) is a valid signature on m. In fact:

$$
sG = \sum_{j=1}^n r_j G + c \cdot \sum_{j=1}^n \mathsf{sk}_j \cdot G = t + c \cdot \mathsf{pk}, \tag{6}
$$

[10] We do not use the fact that only a threshold $t+1$ of the parties are required to sign in our attack. We assume that all the parties come to sign, to simplify the description of the attack.

where $c = \mathsf{H}(t, m)$.

Concurrent Setting Insecurity. Gennaro et al. [GJKR07] proved the security of the scheme in a standalone *sequential* setting, where no two instances of the protocol can be run in parallel. We remark that if an adversary is allowed to start $\ell \geq \lceil \log p \rceil$ sessions in parallel, the attack against CoSi in Section 6.1 can be directly adapted to attack this threshold signature scheme for $n = 2$. The attack of both schemes use the fact that the adversary P_1 (or signer S_1 in CoSi) can see the commitment $t_2 = r_2 G$ of the honest party P_2 (or honest signed S_2) and only then choose r_1 that defines the commitment $t = r_1 G + t_2$. The generalization to any $n \geq 2$ is straightforward.

Scope of the attack. Our attack is an attack against the proposed threshold signature scheme when instantiated with Pedersen's DKG, but not an attack against Perdersen's DKG itself (i.e., JF-DKG from [GJKR07, Fig. 1]). Furthermore, the attack does not work when Perdersen's DKG is replaced by the new DKG protocol from [GJKR07, Fig. 2].

Original version of FROST

Komlo and Goldberg FROST [KG20a] proposed an extension of the above threshold signature scheme that was similarly affected by the above concurrent attack. On 19 July 2020, they updated the signing algorithm [KG20b] in a way that is no more susceptible to the above issue: each party now shares (D_j, E_j) and the commitment is computed as $R = \sum_j D_j + h_j E_j$, where $h_j \coloneqq \mathsf{H}((D_j, E_j, j)_{j \in [t]})$. We direct the reader to [KG20b, Fig. 3] for a more detailed illustration of the problem and the fix.

6.3 Partially blind signatures

Partially blind signatures [AO00] are an extension of blind signature schemes that allow the signer to include some public metadata (e.g., expiration date, collateral conditions, server name, etc.) in the resulting signature.

Abe–Okamoto

Abe and Okamoto [AO00, Fig. 1] propose a partially blind signature scheme inspired from Schnorr blind signatures. Given a verification key $X \coloneqq xG$ and some public information info that is hashed into the group $Z \coloneqq \mathsf{H}(\mathsf{info})$, a partially blind signature for the message $m \in \{0,1\}^*$ is a tuple $(r, c, s, d) \in \mathbb{Z}_p$ where $c + d = \mathsf{H}(rG + cX,\ sG + dZ,\ Z,\ m)$.

Concurrent Setting Insecurity. The security of the above partially blind signature is proved up to a poly-logarithmic number of parallel open sessions in the security parameter [AO00]. We show that the security claim is tight by showing that there exists a poly-time attacker against one-more unforgeability in the setting where the adversary can have $\ell = O(\lambda)$ open sessions using the same metadata info. The attack follows essentially the same strategy of Section 5.1. First, the attacker opens ℓ parallel sessions and obtains the commitments $(A_i, B_i) \in \mathbb{G}^2$ for $i \in [0, \ell - 1]$. It then constructs the polynomial $\boldsymbol{\rho}_\ell$ as per Equation (4). The forged signature for an arbitrary message m^* is computed using the challenge:

$$e_\ell \coloneqq \mathsf{H}(\boldsymbol{\rho}_\ell(\mathbf{A}) + \rho_{\ell,\ell} X,\ \boldsymbol{\rho}_\ell(\mathbf{B}) + \rho_{\ell,\ell} Z,\ Z,\ m^*) - \rho_{\ell,\ell}$$

and closing the ℓ sessions as in Section 5.1, i.e., by using the challenges $e_i^{b_i}$ where b_i is the i-th bit of the canonical representation of e_ℓ. Given the signatures $(r_i, c_i^{b_i}, s_i, d_i)$ for $i \in [0, \ell - 1]$, the attacker can finally create its forgery $(\boldsymbol{\rho}(\mathbf{r}), \boldsymbol{\rho}(\mathbf{c}), \boldsymbol{\rho}(\mathbf{s}), \boldsymbol{\rho}(\mathbf{d}))$. The forgery is indeed correct because:

$$\begin{aligned}
\boldsymbol{\rho}(\mathbf{c}) + \boldsymbol{\rho}(\mathbf{d}) &= \sum_i \rho_i (c_i^{b_i} + d_i) + \rho_{\ell,\ell} + \rho_{\ell,\ell} \\
&= \boldsymbol{\rho}(e_0^{b_0}, \ldots, e_{\ell-1}^{b_{\ell-1}}) + \rho_{\ell,\ell} \\
&= \mathsf{H}(\boldsymbol{\rho}_\ell(\mathbf{r}) G + \boldsymbol{\rho}_\ell(\mathbf{c}) X,\ \boldsymbol{\rho}_\ell(\mathbf{s}) G + \boldsymbol{\rho}_\ell(\mathbf{d}) Z,\ Z,\ m^*)\,.
\end{aligned}$$

6.4 Conditional blind signatures

Conditional blind signatures (CBS), introduced by Grontas et al. [ZGP17], allow a user to request a blind signature on messages of their choice, and the server has a secret boolean input which determines if it will issue a valid signature or not. CBS only allow a *designated* verifier to check the validity of the signature; the user will not able to distinguish between valid and invalid signatures. Conditional blind signature have application in e-voting schemes [GPZZ19].

ZGP17

Zacharakis et al. [ZGP17] propose an instantiation of CBS as an extension of Okamoto–Schnorr blind signatures, where the (designated) verifier holds a secret verification key $k \in \mathbb{Z}_p$ and publishes $K = kG$ as public information. During the execution of Okamoto–Schnorr, one of the two responses (s, t) will be computed in $\mathbb{G}$ rather than $\mathbb{Z}_p$, using K as a generator. Only the designated verifier, who knows the discrete log of K can now check the verification equation.

The attack from Section 5.2 directly applies also to their scheme, and leads to a poly-time adversary that with λ queries to the signing oracle for the same bit $b = 1$ can produce one-more forgery with overwhelming probability. This attack does not invalidate the security claims of [ZGP17], which are argued only for a poly-logarithmic number of parallel open sessions.

6.5 Other schemes

The following papers prove rely on the hardness of the ROS problem for their security proofs, and henceforth may not provide the expected security guarantees: blind anonymous group signatures [CFLW04]; blind identity-based signcryption [YW05]; blind signature schemes from bilinear pairings [CHYC05].

7 Conclusions

Our work provides a polynomial attack against $\mathrm{ROS}_\ell(\lambda)$ when $\ell > \log p$, and a sub-exponential attack for $\ell < \log p$. This impacts the one-more unforgeability property of Schnorr and Okamoto–Schnorr blind signatures, plus a number of cryptographic schemes derived from them. Our attacks run in polynomial time only in the concurrent setting, and only for $\ell > \log p$ parallel signing sessions.

Concretely, the cost of the attack and the number of sessions required are rather small: for today's security parameters, the attack could be already mounted with $\ell = 9$ parallel open sessions. As already pointed out by [FPS20], even just $\ell = 16$ open sessions could lead to a forgery in time $O(2^{55})$. For $\ell = 128$, our attack of Section 4 leads to a forgery in time $O(2^{32})$. For $\ell = 256$, our attack of Section 3 produces a forgery in a matter of seconds on commodity hardware. Although 256 parallel signing sessions might seem at first unrealistic, modern large-scale web servers must handle more than 10 million concurrent sessions[11]. Given our attack, the main takeaway of our work is that blind Schnorr signatures are unsuitable for wide-scale deployments.

The easiest countermeasure to our attack could be to allow only for sequential signing sessions, as Schnorr blind signatures are unforgeable in the algebraic group model for polynomially many sessions [KLRX]. Another countermeasure to our attack could be to employ (much) larger security parameters, require the signer to enforce strong ratio limits, *and* perform frequent key rotations, accepting the tradeoffs given by our attacks. Finally, Fuchsbauer et al. [FPS20] recently introduced a variant of blind Schnorr signatures (the *clause* version) which is unaffected by our attack. Unfortunately, it relies on the conjectured hardness of the so-called *modified ROS problem*, which is still relatively new and has not been subject to any significant cryptanalysis.

To conclude, other blind signature schemes that are to this day considered secure, such as the blind RSA signature scheme [Cha82], the blind BLS signature scheme [Bol03], and Abe's blind signature scheme [Abe01, KLRX].

[11] For further information, read the C10K problem ('99) and the C10M problem ('11).

References

Abe01. Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151. Springer, Heidelberg, May 2001.

AO00. Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 271–286. Springer, Heidelberg, August 2000.

Bol03. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003.

CFLW04. Tony K. Chan, Karyin Fung, Joseph K. Liu, and Victor K. Wei. Blind spontaneous anonymous group signatures for ad hoc groups. In *ESAS*, volume 3313 of *Lecture Notes in Computer Science*, pages 82–94. Springer, 2004.

Cha82. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

CHYC05. Sherman S. M. Chow, Lucas Chi Kwong Hui, Siu-Ming Yiu, and K. P. Chow. Two improved partially blind signature schemes from bilinear pairings. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 05*, volume 3574 of *LNCS*, pages 316–328. Springer, Heidelberg, July 2005.

DEF$^+$19. Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101. IEEE Computer Society Press, May 2019.

Fel87. Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th FOCS*, pages 427–437. IEEE Computer Society Press, October 1987.

FPS20. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Heidelberg, May 2020.

GJKR07. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, January 2007.

GPZZ19. Panagiotis Grontas, Aris Pagourtzis, Alexandros Zacharakis, and Bingsheng Zhang. Towards everlasting privacy and efficient coercion resistance in remote electronic voting. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *FC 2018 Workshops*, volume 10958 of *LNCS*, pages 210–231. Springer, Heidelberg, March 2019.

HKL19. Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Heidelberg, May 2019.

HKLN20. Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 500–529. Springer, Heidelberg, August 2020.

KG20a. Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures, 2020. https://crysp.uwaterloo.ca/software/frost/frost-extabs.pdf; version from "January 7, 2020"; accessed 2020-10-04.

KG20b. Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. Cryptology ePrint Archive, Report 2020/852, 2020. https://eprint.iacr.org/2020/852.

KLRX. Julia Kaster, Julian Loss, Michael Rosenberg, and Jiayu Xu. On pairing-free blind signature schemes in the algebraic group model. Cryptology ePrint Archive, Report 2020/1071.

MPSW18a. Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signature with applications to Bitcoin. Cryptology ePrint Archive, Report 2018/068, Revision 20180118:124757, 2018. https://eprint.iacr.org/2018/068/20180118:124757.

MPSW18b. Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signature with applications to Bitcoin. Cryptology ePrint Archive, Report 2018/068, Revision 20180520:191909, 2018. https://eprint.iacr.org/2018/068/20180520:191909.

MS09. Lorenz Minder and Alistair Sinclair. The extended k-tree algorithm. In Claire Mathieu, editor, *20th SODA*, pages 586–595. ACM-SIAM, January 2009.

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

S$^+$20. W. A. Stein et al. *Sage Mathematics Software (Version 9.1)*. The Sage Development Team, 2020. http://www.sagemath.org.

Sch01. Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 1–12. Springer, Heidelberg, November 2001.

STV+16. Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *2016 IEEE Symposium on Security and Privacy*, pages 526–545. IEEE Computer Society Press, May 2016.

Wag02. David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Heidelberg, August 2002.

YW05. Tsz Hon Yuen and Victor K. Wei. Fast and proven secure blind identity-based signcryption from pairings. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 305–322. Springer, Heidelberg, February 2005.

ZGP17. Alexandros Zacharakis, Panagiotis Grontas, and Aris Pagourtzis. Conditional blind signatures. Cryptology ePrint Archive, Report 2017/682, 2017. http://eprint.iacr.org/2017/682.