Lattice-Based Threshold Signature Implementation for Constrained Devices

Patrik Dobias[1][a], Sara Ricci[1][b], Petr Dzurenda[1][c], Lukas Malina[1][d] and Nikita Snetkov[2,3][e]

[1] *Department of Telecommunications, Brno University of Technology, Brno, Czech Republic*

[2] *Cybernetica AS, Mäealuse 2/1, 12618 Tallinn, Estonia*

[3] *Tallinn University of Technology, Akadeemia Tee 15a, 12618 Tallinn, Estonia*

Keywords: Threshold Signature, Lattice-Based Cryptography, Software Implementation, Dilithium, Homomorphic Commitment, Embedded Systems, Constrained Devices.

Abstract: Threshold signatures have gained increased attention especially due to their recent applications in blockchain technologies. In fact, current cryptocurrencies such as Bitcoin, and Cardano started to support multi-signature transactions. Even if the Schnorr-based threshold signatures improve the blockchain's privacy and scalability, these schemes do not provide post-quantum security. In this paper, we propose the optimization of the DS2 lattice-based (n,n)-threshold signature scheme and present its practical implementation. Moreover, we evaluate our optimized implementation of the DS2 scheme on different platforms. The results demonstrate that our implementation is easily portable and executable on constrained devices based on ARM Cortex-A53, ARM Cortex-M3, and ESP32 architectures.

1 INTRODUCTION

In recent years, the use of distributed signature schemes has received increasing attention, mainly because of their usage in the blockchain field. A typical blockchain involves the usage of a signature scheme for the verification of transactions that are signed only with one private key, resulting in a single point of failure. This means that the attacker who compromises the device that stores that private key may create a new transaction signed with this key. In contrast, multi-signature transactions are signed using at least two shares of a private key that can be stored on different devices. The use of multi-signature transactions offers multiple benefits (Freemanlaw, 2022), e.g., 1) `increased security` - shares are stored across multiple devices, eliminating an attack on a single device, 2) `private key backup` - losing the private key share does not lead to access loss as long as there are enough shares, 3) `divided possession` - possession of the cryptocurrency is divided between multiple parties. In this context, threshold signatures can be used to securely sign messages, where t parties of the n authorized parties collaborate in the signing process. Current threshold signatures are mainly based on RSA (Damgård and Koprowski, 2001; Shoup, 2000) and Elliptic Curve Digital Signature Algorithm (ECDSA) signatures (Castagnos et al., 2020; Cogliati et al., 2018; Damgård et al., 2020; Gennaro and Goldfeder, 2018; Lindell and Nof, 2018). Alternative threshold signature (Komlo and Goldberg, 2020; Ricci et al., 2022) are based on Schnorr's protocol. Even if having more communication overload, these latter schemes guarantee robustness as a main feature. For instance, in (Komlo and Goldberg, 2020), honest participants can detect and disqualify a malicious participant if he/she misbehaves. The Schnorr-based multi and threshold signatures received significant interest recently, especially in the context of multi-signature transactions. In fact, several cryptocurrencies (e.g., Bitcoin, Cardano, Polkadot, Kusama, Zilliqa, and Decred) start to integrate the Schnorr signature into their blockchain. This is due to Schnorr-based distributed signatures' simplicity, efficiency, flexibility, and short signature bit length.

All aforementioned schemes rely on Non-deterministic Polynomial (NP) problems, i.e., Inte-

[a] https://orcid.org/0000-0002-7321-7003
[b] https://orcid.org/0000-0003-0842-4951
[c] https://orcid.org/0000-0002-4366-3950
[d] https://orcid.org/0000-0002-7208-2514
[e] https://orcid.org/0000-0002-1414-2080

Dobias, P., Ricci, S., Dzurenda, P., Malina, L. and Snetkov, N.
Lattice-Based Threshold Signature Implementation for Constrained Devices.
DOI: 10.5220/0012112700003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 724-730
ISBN: 978-989-758-666-8; ISSN: 2184-7711

Table 1: Comparison of existing lattice-based threshold signature schemes.

Scheme	Problem	Type	Rounds	Building Blocks
(Benhamouda et al., 2016)	H&S	t-out-of-n	1	Honest majority MPC
(Boneh et al., 2018)	FSwA	t-out-of-n	1	Threshold FHE
(Fukumitsu and Hasegawa, 2020)	FSwA	Multisignature	3	-
(Vakarjuk et al., 2021)	FSwA	2-out-of-2	3	Homomorphic Hash
(Boschini et al., 2022)	FSwA	Multisignature	2 (1 offline)	Linear combination
(Chen, 2023)	FSwA	Multisignature	2	Linear combination
(Laud et al., 2022)	FSwA	2-out-of-2	3	Homomorphic Com.
(Damgård et al., 2022)	FSwA	*n*-out-of-*n*	2	Homomorphic Com.
(Liu et al., 2022)	FSwA	*n*-out-of-*n*	3	Homomorphic Com.

“FSwA” states for s Fiat–Shamir with aborts signatures, ”H&S” for hash-and-sign, and ”MPC” for multi-party computation.

ger Factorization Problem (IFP), Discrete Logarithm Problem (DLP), and Elliptic Curve DLP (ECDLP) that are vulnerable to attacks run on quantum computers. In particular, Shor’s algorithm (Bernstein, 2009) allows solving the aforementioned NP problem in polynomial time. Therefore, the need is to shift towards post-quantum schemes, which are considered safe against quantum computer attacks. To support research on post-quantum schemes, the National Institute of Standards and Technology (NIST) initiated a standardization process in 2016, which led to the selection of three signature schemes for standardization in 2022[1], specifically Dilithium and Falcon, which are lattice-based, and SPHINCS+, which is hash-based. However, this standardization process focuses only on plain digital signatures and does not cover distributed signature schemes. Therefore, distributed signatures’ complexity, their practical usability, and their implementability on constrained devices for the current Internet of Things (IoTs) ecosystem are not researched. It is noteworthy that NIST has recently published a draft for their First Call for Multi-Party Threshold Schemes (Brandao and Peralta, 2023), which signals their interest in advancing research in this area.

1.1 Related Work

Table 1 shows a comparison of existing lattice-based threshold signatures. In (Vakarjuk et al., 2021), authors presented a two-party lattice-based Dilizium signature scheme that follows the Fiat-Shamir with Aborts paradigm. Moreover, the scheme is based on the variant of CRYSTALS-Dilithium scheme (Ducas et al., 2018), which was presented in (Kiltz et al., 2018), so it does not require sampling from a discrete Gaussian distribution. They considered a similar approach to (Damgård et al., 2022), but a homomorphic hash function was deployed instead of homomorphic commitments, as they find homomorphic commitments computationally inefficient. (Laud et al., 2022) presented an updated version of the Dilizium scheme, where they came back to their previous decision and used a homomorphic commitment scheme. Furthermore, they also applied compression mechanisms from the original CRYSTALS-Dilithium proposal to make it closer to the version submitted for the NIST standardization. In (Damgård et al., 2022), authors proposed a two-round *n*-out-of-*n* signature scheme called DS2, a multi-signatures scheme, and a trapdoor commitment from lattices. The signature scheme relies on the homomorphic property of the proposed commitment scheme and they are based on the Dilithium-G scheme (Ducas et al., 2018). (Benhamouda et al., 2016) presented a threshold signature based on GPV hash-and-sign signatures. In particular, the signature size does not grow with the number of parties, but the signature requires computing the lattice Gaussian sampling phase with a multiparty computation. (Boneh et al., 2018) developed a general solution for non-interactive, t-out-of-n threshold signatures based on Fully Homomorphic Encryption (FHE). Their proposal has both signature and verification phases that do not dependent on the number of participants. However, a trusted party is needed in the set-up scheme in contrast with (Damgård et al., 2022). (Fukumitsu and Hasegawa, 2020) proposed a multisignature based on Dilithium with signature size compression techniques on contrary to (Damgård et al., 2022) proposal. (Boschini et al., 2022) introduced a lattice-based multi-signature scheme that does not require any additional primitive in the protocol. In this way, the signature sizes are closer to one of a plain digital signature. Moreover, the scheme consists of only one online round, resulting in lower communication overhead. (Chen, 2023) presented an efficient lattice-based two-round multi-signature scheme. The scheme is based on Dilithium-G and has smaller public keys and signatures compared to (Boschini et al., 2022). Additionally, they also proposed a novel “dual

[1] https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022

signing simulation" technique that simulates multi-signatures in the security proof without trapdoors. At last, (Liu et al., 2022) proposed a multiparty signature scheme with detectability in the industrial IoT. They also implemented this scheme and benchmarked it for different security levels and different numbers of parties. It is important to notice that in a multi-signature each signer has its public key instead of secret shares of a single common public key as in threshold signature schemes.

1.2 Contribution and Paper Structure

In this work, we present the first implementation of DS2 (n,n)-threshold signature scheme (Damgård et al., 2022). We optimize this scheme and define its parameters to increase its computational speed, and reduce its key, signature sizes, and communication overhead. We do several benchmarks of our implementation on devices with different performances, including constrained devices using ARM Cortex-A53, ARM Cortex-M3, and ESP32 architectures. Our implementation is easily portable and executable on constrained devices capable of running C applications.

The paper is organized as follows. Section 2 presents our DS2 architecture, choice of parameters, and detailed description of our lightweight implementation written in C programming language. Section 3 presents the experimental benchmarks of our implementation on various constrained devices. In the last section, we conclude this work.

2 IMPLEMENTATION DETAILS

This section describes our implementation of the DS2 scheme, which was implemented using the C programming language. It depends only on the standard libc library. Furthermore, the implementation uses CMake for project management, and therefore, it is easily portable and executable on different platforms and constrained devices capable of running C applications. Additionally, a lattice-based trapdoor commitment scheme was implemented with the parameters specified in Section 2.1. To be more precise, we implemented the three most important algorithms, i.e., `CGen`, `Commit`, and `Open` algorithms. The `Commit` algorithm was implemented to compute $\mathbf{com} = ck \cdot \mathbf{r} + \begin{pmatrix}0\\x\end{pmatrix}$ reusing polynomial functions for DS2 with different parameters, the commitment key is generated per message using `CGen` algorithm implemented as a hash function $ck := \mathrm{H}_3(\mu, pk)$, and the `Open` algorithm is simply the `Commit` algorithm with the comparison of commitments at the end. We refer to (Damgård et al., 2022) for more details. As the original DS2 design is based on non-optimized version of Dilithium-G (Ducas et al., 2018) that would be inefficient to implement, we apply changes to each phase based on the optimized version of Dilithium-G. For each phase, a modified algorithm is shown and implementation details are provided. The red marked text indicates our changes in individual algorithms compared to the original algorithms of DS2 (Damgård et al., 2022). Our implementation is open-source and available on the public GitLab repository[2].

Table 2: Choice of parameters for DS2 scheme.

n	α	σ	B	M
2	22	298694	27034695	1.7272
5	55	746735	67586737	1.2440
10	110	1493470	135173474	1.1153
20	220	2986940	270346948	1.0561
50	550	7467350	675867370	1.0221

2.1 Chosen Parameters

Since (Damgård et al., 2022) does not provide the instantiation of the parameters, the recommended parameters for Security Level 128b from the Dilithium-G specification (Ducas et al., 2018) were used. Specifically, $N = 256, q = 8380417, (k,\ell) = (4,4), \eta = 5, \kappa = 60, d = 12$. Additional parameters specific to the DS2 scheme were chosen to satisfy the required bounds specified in (Damgård et al., 2022) and to reduce the complexity of communication and the size of the resulting signature. For the different number of parties n and targeting the total number of rejections during the signing phase $M_n = 3$, the parameters are presented in Table 2. In particular, M states for "the expected number of restarts until a single party can proceed", B for "the maximum L^2-norm of signature share $\mathbf{z}_j$ for $j = 1,\ldots,n$, σ for the "standard deviation of the Gaussian distribution", and α for a "parameter defining σ and M". Note that a new commitment needs to be sent any time a rejection appears, so the transmitted data increases by $d_{com} = 5888B$ for each rejection. Moreover, the total transmitted data d_{sig} increases linearly with the number of rejections. Furthermore, the parameters for the lattice-based commitment scheme were chosen to satisfy the specified bounds. Specifically, the following parameters were chosen: $\bar{s} = 2059, s = 815, B = 48414, l = 23, w = 23$.

[2] https://gitlab.com/brno-axe/chess/ds2/ds2-opt-c-impl

2.2 Key Generation Phase

In the original DS2 scheme, the individual shares of **A** matrix are exchanged between the parties followed by the shares of part of public key **t**. In contrast, in our implementation, participants have to collaborate in creating the seed ρ which is then used to expand matrix **A** as highlighted in Algorithm 1.

The matrix **A** is then sampled uniformly from this seed, ensuring that it remains indistinguishable from a uniformly distributed random matrix. During the computation of the combined public key $\mathbf{t}_1$, we used `power2round` rounding, as it is proposed in the Dilithium-G scheme, to reduce the size of the transmitted data. The total data transferred by a single party is only dependent on the number of parties and is calculated as $d_{gen}(n) = 1472 \times (n-1)$ B.

Algorithm 1: `KeyGen`(1^κ).

```
$\rho_n \leftarrow \{0,1\}^{256}$
Send out $\rho_n$ and receive $\rho_i; i \in [n-1]$
$\rho = \mathrm{H}(\rho_1|\rho_2|\ldots\rho_n)$
$\bar{\mathbf{A}} := [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$; $\mathbf{A} \leftarrow R_q^{k\times\ell} := \mathtt{Sam}(\rho)$
$\mathbf{t}_n := \bar{\mathbf{A}}\mathbf{s}_n; \mathbf{s}_n \leftarrow S_\eta^{\ell+k}$
$(\mathbf{t}_{n_1}, \mathbf{t}_{n_0}) := \mathtt{Power2Round}(\mathbf{t_n}, d); g_n := \mathrm{H}_2(\mathbf{t}_{n_1})$
Send out $g_n$ and receive $g_i; i \in [n-1]$
Send out $\mathbf{t}_{n_1}$, receive $\mathbf{t}_{i_1}$ and check $g_i = \mathrm{H}_2(\mathbf{t}_{i_1}); i \in [n-1]$
$\mathbf{t}_1 := \sum_{i\in[n]} \mathbf{t}_{i_1}$
$tr \in \{0,1\}^{384} := \mathtt{CRH}(\rho||\mathbf{t}_1)$
return $pk = (\rho, \mathbf{t}_1)$, $sk = (\rho, tr, \mathbf{s}_n, \mathbf{t}_{n_0})$
```

2.3 Signing Phase

During the signing phase, n parties collaborate to generate a valid signature. To reduce the size of the signature, we use the Fiat-Shamir trick of expressing the signature as $(c, \mathbf{z}, \mathbf{r})$ instead of $(\mathbf{com}, \mathbf{z}, \mathbf{r})$, as shown in Algorithm 2. In this way, only a small challenge seed c is sent to the verifier instead of the full commitment **com**. As a result of this, the resulting signature size is reduced approximately by 3%. For example, with 2 parties, we transfer 218128 B instead of 224256 B. The commitment is then computed during the verification phase using the reconstructed vector of polynomials **w**. The signature is still large compared to the standard signature size due to the commitment randomness. Additionally, since we use `Power2Round` algorithm during the key generation phase to reduce communication, computation of the signature part $\mathbf{z}_{n_2}$ needs to be altered as highlighted on line 8 in Algorithm 2. During the signing phase, there are two messages exchanged between each party. First, the commitment to $\mathbf{w}_n$ with size $d_{com_n} = (|\mathbf{com}| \cdot \lceil log_2 q\rceil \cdot N \cdot k)/8 = 5888$ B. Followed by the signature part $\mathbf{z}_n$ and $\mathbf{r}_n$ with sizes $d_{z_n} = (\lceil log_2 q\rceil \cdot (k+\ell) \cdot N)/8 = 5888$ B and $d_{r_n} = (\lceil log_2 q\rceil \cdot (l+2w) \cdot N \cdot k)/8 = 203136$ B. The total data transferred by a single party during signing also depends on the number of rejections. Therefore only the average number can be computed as $d_{sig}(n) = 226688 \times (n-1)$ B.

Algorithm 2: `Sign`$(pk, sk_n, \mu \in M)$.

```
1:  $ck := \mathrm{H}_3(\mu, pk)$
2:  $\mathbf{w}_n := \bar{\mathbf{A}}\mathbf{y}_n; \mathbf{y}_n \leftarrow D_s^{\ell+k}$
3:  $com_n := \mathtt{Commit}_{ck}(\mathbf{w}, \mathbf{r_n}); \mathbf{r}_n \leftarrow D(S_r)$
4:  Send out $com_n$ and receive $com_i; i \in [n-1]$
5:  $c := \mathrm{H}_0(com, \mu, tr); com = \sum_{i\in[n]} com_i$
6:  $\mathbf{z}_n := \begin{pmatrix}\mathbf{z}_{n_1}\\ \mathbf{z}_{n_2}\end{pmatrix} := c\mathbf{s}_n + \mathbf{y}_n$
7:  Rejection sampling on $(c\mathbf{s}_n, \mathbf{z}_n)$, with probability $min(1, D_s^{l+k}(\mathbf{z}_n)/(M \cdot D_s^{l+k}(\mathbf{z}_{c\mathbf{s}_n,n})))$ continue, otherwise goto 2
8:  $\mathbf{z}_{n_2} = \mathbf{z}_{n_2} - c\mathbf{t}_{n0}$
9:  Send out $(\mathbf{z}_n, \mathbf{r}_n)$ and receive $(\mathbf{z}_i, \mathbf{r}_i); i \in [n-1]$
10: for $i \in [n-1]$ do
11:     $\mathbf{w}_i := \bar{\mathbf{A}}\mathbf{z}_i - c\mathbf{t}_{i_1} \cdot 2^d$
12:     if $||z_i||_2 > B$ or $\mathtt{Open}_{ck}(com_i, \mathbf{r}_i, \mathbf{w}_i) \neq 1$ then abort
13:     $\mathbf{z} := \sum_{i\in[n]} \mathbf{z}_i; \mathbf{r} := \sum_{i\in[n]} \mathbf{r}_i$
14: return $\Sigma = (c, \mathbf{z}, \mathbf{r})$
```

2.4 Verification Phase

Algorithm 3: `Verify`$(pk, \Sigma, \mu \in M)$.

```
$ck' := \mathrm{H}_3(\mu, pk)$
$\mathbf{w}' := \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d$
$c' := \mathrm{H}_0(\mathtt{Commit}_{ck'}(\mathbf{w}', \mathbf{r}), \mu, \mathtt{CRH}(pk))$
if $||\mathbf{z}||_2 \leq \sqrt{n}B$ and $c = c'$ then return 1
else return 0
```

The verification phase is similar to the standard Dilithium-G scheme. In the original DS2 design, commitments must be opened with the reconstructed vector **w** and verified. As we used the Fiat-Shamir trick and the commitments are not part of the signature, the commitments are computed from the reconstructed vector **w**, then a new challenge is computed using the hash function and is compared with the challenge that is a part of the signature. The changes are highlighted in Algorithm 3. This phase is independent of the number of parties and anyone with access to the common public key can verify the signature. The correctness of our proposal is proven in the following theorem.

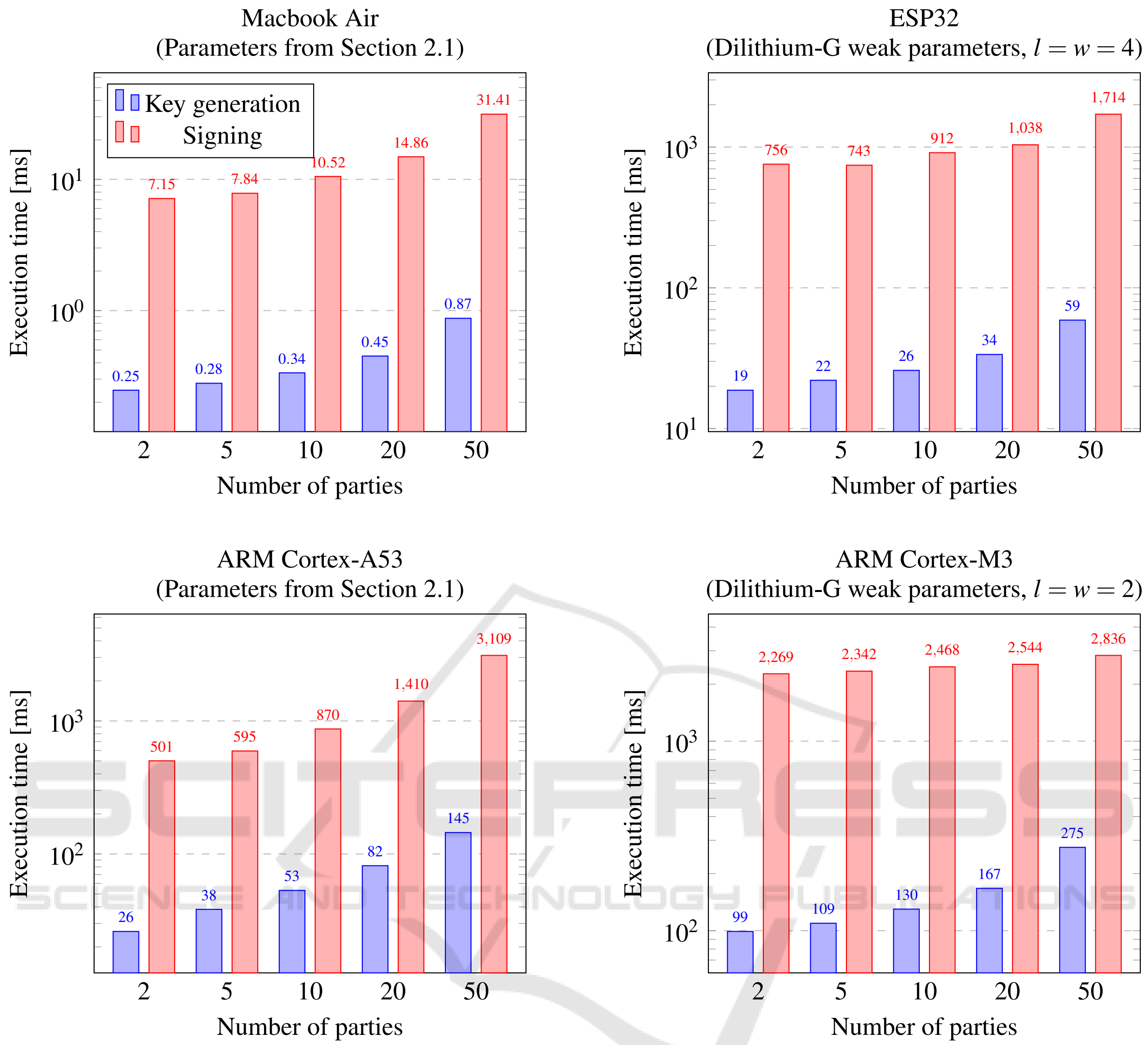

Figure 1: Execution times during key generation and signing phases.

Theorem 1 (Correctness)**.** *The Verification process in Algorithm 3 is correct.*

Proof. For the signature to be accepted, we need to show that the newly computed c' is equal to c, that is $\mathrm{H}_0(com, \mu, tr)$ is equal to $\mathrm{H}_0(\texttt{Commit}_{ck}(\mathbf{w}', \mathbf{r}), \mu, \texttt{CRH}(pk))$. This can be done by proving that com is equal to $\texttt{Commit}_{ck}(\mathbf{w}', \mathbf{r})$. In fact, by definition, $tr := \texttt{CRH}(\rho||\mathbf{t}_1)$, where pk is a public value and, likewise, $ck' := \mathrm{H}_3(\mu, pk)$ is equal to ck. Therefore, the commitments equality can be proven as follows:

$$\begin{aligned}
\texttt{Commit}_{ck}(\mathbf{w}', \mathbf{r}) &= \texttt{Commit}_{ck}(\bar{A}z - c\mathbf{t}_1 \cdot 2^d, \mathbf{r}) \\
&= \texttt{Commit}_{ck}(\sum_{i \in [n]} \bar{A}z_i - c\mathbf{t}_{i1} \cdot 2^d, \sum_{i \in [n]} \mathbf{r}_i) \\
&= \sum_{i \in [n]} \texttt{Commit}_{ck}(z_i - c\mathbf{t}_{i1} \cdot 2^d, \mathbf{r_i}) \\
&= \sum_{i \in [n]} Com_i = Com
\end{aligned}$$

Note that it is straightforward that $||\mathbf{z}||_2 \leq \sqrt{n}B$ since the signers only output a signature shares that respect $||\mathbf{z}_i||_2 \leq B$. We refer to (Damgård et al., 2022), Lemma 2 for more details. □

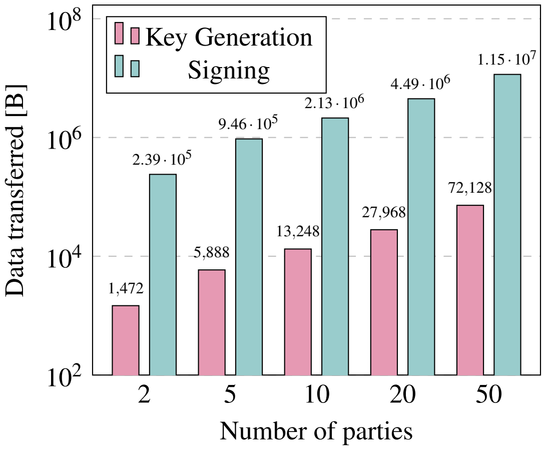

Figure 2: Number of data transferred (from/to one device) needed during key generation and signing phases.

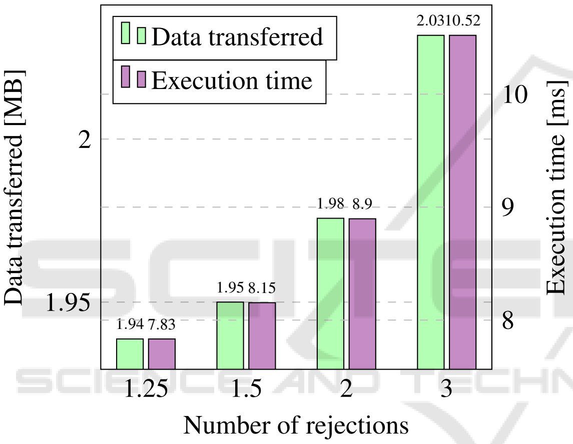

Figure 3: Number of data transferred needed and execution times during signing with 10 parties.

3 EXPERIMENTAL RESULTS

In this section, we present the experimental results of our implementation. We used devices with different computing power such as Macbook Air 2020 (CPU Apple M1, 3.2 GHz) representing a personal computer, and constrained devices based on ARM Cortex-A53 (represented by Raspberry Pi Zero), ARM Cortex-M3 (represented by Arduino Due), and ESP32 processors. We benchmark the scheme depending on the number of signers and the number of rejections. Due to the high memory requirements of the DS2 scheme, we had to use smaller parameters during the benchmarking on ARM Cortex-M3 and ESP32. This involved using Dilithium-G weak parameters and setting the lattice commitment $l = w = 2$ for ARM Cortex-M3 and $l = w = 4$ for ESP32.

3.1 Communication Complexity and Performance Results

For execution times, only computation time was measured, not including the time needed for communication. The results are presented in Figure 1. The average times for 1000 runs are used for the signature generation phase. The execution time of verification does not depend on the number of parties or the number of rejections, so it is the same for all measurements and it was 1.87 ms on Macbook, 102.55 ms on ESP32, 179.26 ms on ARM Cortex-A53, and 342.50 ms on ARM Cortex-M3. In Figure 2, the data transferred from/to one device in bytes during the key generation and signing phases are presented for the different number of parties. In Figure 3, the execution times and data transferred are presented for different number of rejections. The measurements are presented for the constant number of parties $n = 10$ on Macbook.

4 CONCLUSION

In this work, we propose optimization techniques and parameter selection for the DS2 scheme. To our best knowledge, we provide the first implementation of this scheme which is easily portable and executable on different platforms and constrained devices capable of running C applications. Further, we provide benchmarks of our DS2 implementation on various devices. The experimental results show the efficiency of our optimized implementation even on constrained devices, where the signing phase (without a communication overhead) takes less than 2 s on ESP32 and less than 3 s on ARM Cortex-M3. Our future work will focus on further optimization of algorithms, e.g., the use hint algorithm.

ACKNOWLEDGEMENTS

This work is supported by Ministry of the Interior (CZ) under grant VJ01010008 and by the European Union under Grant Agreement No. 101087529 CHESS. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Research Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

Benhamouda, F., Krenn, S., Lyubashevsky, V., and Pietrzak, K. (2016). Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *Computer Security–ESORICS 2015: 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*, pages 305–325. Springer.

Bernstein, D. J. (2009). Introduction to post-quantum cryptography. In *Post-quantum cryptography*, pages 1–14. Springer.

Boneh, D., Gennaro, R., Goldfeder, S., Jain, A., Kim, S., Rasmussen, P. M., and Sahai, A. (2018). Threshold cryptosystems from threshold fully homomorphic encryption. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part I 38*, pages 565–596. Springer.

Boschini, C., Takahashi, A., and Tibouchi, M. (2022). Musig-l: Lattice-based multi-signature with single-round online phase. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part II*, pages 276–305. Springer.

Brandao, L. and Peralta, R. (2023). Nist first call for multi-party threshold schemes.

Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., and Tucker, I. (2020). Bandwidth-efficient threshold ec-dsa. In *IACR International Conference on Public-Key Cryptography*, pages 266–296. Springer.

Chen, Y. (2023). Dualms: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. *Cryptology ePrint Archive*.

Cogliati, B., Dodis, Y., Katz, J., Lee, J., Steinberger, J., Thiruvengadam, A., and Zhang, Z. (2018). Provable security of (tweakable) block ciphers based on substitution-permutation networks. In *Annual International Cryptology Conference*, pages 722–753. Springer.

Damgård, I., Jakobsen, T. P., Nielsen, J. B., Pagter, J. I., and Østergård, M. B. (2020). Fast threshold ecdsa with honest majority. In *International Conference on Security and Cryptography for Networks*, pages 382–400. Springer.

Damgård, I. and Koprowski, M. (2001). Practical threshold rsa signatures without a trusted dealer. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 152–165. Springer.

Damgård, I., Orlandi, C., Takahashi, A., and Tibouchi, M. (2022). Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Crypatology*, 35(2):1–56.

Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., and Stehlé, D. (2018). Crystals–dilithium: Digital signatures from module lattices.

Freemanlaw (2022). Cryptocurrency transactions: Multi-signature arrangements explained.

Fukumitsu, M. and Hasegawa, S. (2020). A lattice-based provably secure multisignature scheme in quantum random oracle model. In *Provable and Practical Security: 14th International Conference, ProvSec 2020, Singapore, November 29–December 1, 2020, Proceedings 14*, pages 45–64. Springer.

Gennaro, R. and Goldfeder, S. (2018). Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1179–1194.

Kiltz, E., Lyubashevsky, V., and Schaffner, C. (2018). A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part III 37*, pages 552–586. Springer.

Komlo, C. and Goldberg, I. (2020). Frost: flexible round-optimized schnorr threshold signatures. In *International Conference on Selected Areas in Cryptography*, pages 34–65. Springer.

Laud, P., Snetkov, N., and Vakarjuk, J. (2022). Dilizium 2.0: Revisiting two-party crystals-dilithium. *Cryptology ePrint Archive*.

Lindell, Y. and Nof, A. (2018). Fast secure multiparty ecdsa with practical distributed key generation and applications to cryptocurrency custody. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1837–1854.

Liu, J., Wen, J., Zhang, B., Dong, S., Tang, B., and Yu, Y. (2022). A post quantum secure multi-party collaborative signature with deterability in the industrial internet of things. *Future Generation Computer Systems*.

Ricci, S., Dzurenda, P., Casanova-Marqués, R., and Cika, P. (2022). Threshold signature for privacy-preserving blockchain. In *Business Process Management: Blockchain, Robotic Process Automation, and Central and Eastern Europe Forum: BPM 2022 Blockchain, Germany, September 11–16, 2022, Proceedings*, pages 100–115. Springer.

Shoup, V. (2000). Practical threshold signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 207–220. Springer.

Vakarjuk, J., Snetkov, N., and Willemson, J. (2021). Dilizium: A two-party lattice-based signature scheme. *Entropy*, 23(8):989.