Security Analysis Of DGM and GM Group Signature Schemes Instantiated With XMSS-T

Mahmoud Yehia(✉), Riham AlTawy(✉), and T. Aaron Gulliver

Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada.
mahmoudyehia@uvic.ca, raltawy@uvic.ca

Abstract. Group Merkle (GM) (PQCrypto 2018) and Dynamic Group Merkle (DGM) (ESORICS 2019) are recent proposals for post-quantum hash-based group signature schemes. They are designed as generic constructions that employ any stateful Merkle hash-based signature scheme. XMSS-T (PKC 2016, RFC8391) is the latest stateful Merkle hash-based signature scheme where (almost) optimal parameters are provided. In this paper, we show that the setup phase of both GM and DGM does not enable drop-in instantiation by XMSS-T which limits both designs in employing earlier XMSS versions with sub-optimal parameters which negatively affects the performance of both schemes. Thus, we provide a tweak to the setup phase of GM and DGM to overcome this limitation and enable the adoption of XMSS-T. Moreover, we analyze the bit security of DGM when instantiated with XMSS-T and show that it is susceptible to multi-target attacks because of the parallel Signing Merkle Trees (SMT) approach. More precisely, when DGM is used to sign 2^{64} messages, its bit security is 44 bits less than that of XMSS-T. Finally, we provide a DGM variant that mitigates multi-target attacks and show that it attains the same bit security as XMSS-T.

Keywords: Digital signatures, Hash-based signature schemes, Group signature schemes, Post-quantum cryptography, Merkle trees.

1 Introduction

A group signature scheme (GSS) incorporates N members in a signing scheme with a single public key. GSS allows any group member to sign anonymously on behalf of the whole group [16]. A group manager is assigned to perform system setup, reveal the identity of a given signer when needed, add new members, and revoke memberships when required. Remote attestation protocols, e-commerce, e-voting, traffic management, and privacy preserving techniques in blockchain applications [8, 35, 3] are applications that utilize group signature schemes. There have been several proposals for group signature schemes [14, 15, 8, 6, 28, 29]. However, the majority rely on number theoretic assumptions that are not secure against post-quantum attacks.

There is now an imperative need to replace the current public key infrastructure with quantum-secure algorithms. This is evidenced by the current NIST post-quantum cryptography standardization competition (PQC) [33]. GSS is a public key infrastructure primitive which has attracted research attention to provide quantum security. The first post-quantum lattice-based group signature scheme was proposed in [20], and other schemes were proposed in [25, 26, 30, 32, 27, 17]. However, unlike the lattice-based signature scheme PQC finalists, their group signature structures are not as efficient [36]. Code-based group signature schemes were developed to provide another alternative for quantum secure GSSs [1, 2, 19], but they have very large signature sizes on the order of Megabytes [4].

In 2018, El Bansarkhani and Misoczki introduced Group Merkle (GM), the first post-quantum stateful hash-based group signature scheme [18]. A year Later, Dynamic Group Merkle (DGM), the latest hash-based group signature scheme, was introduced to solve some of the limitations of GM [12]. GM and DGM provide quantum security with reasonable signature sizes on the order of KBytes and both are general constructions that can be instantiated with any stateful Merkle hash-based signature scheme. The security analysis of both schemes included standard security notions of group signature schemes (anonymity and full-traceability) [5, 13], but no bit security analysis was provided. XMSS$^+$ [21], XMSS^{MT} [23], and XMSS-T [24] are stateful hash-based signature schemes that overcome the performance drawbacks of Merkle Signature Scheme (MSS) [31]. The last version of XMSS-T given in Internet Engineering Task Force (IETF) RFC8391 [22] provides (almost) optimal parameters and mitigates multi-target attacks [9].

Our contributions. The contributions of this work are as follows.

- We show that the setup phase of both GM and DGM restricts them from being directly instantiated by XMSS-T which negatively affects the performance of both schemes because they may use earlier XMSS versions with sub-optimal parameters.
- We introduce simple tweaks to the GM and DGM setup phases that enable their instantiation with XMSS-T.
- We analyze the bit security of DGM when instantiated with XMSS-T and show that it is vulnerable to multi-target attacks due to allowing multiple signing trees to branch out from the same intermediate initial Merkle tree node. Concretely, when the scheme is used to sign 2^{64} messages under the same public key (similar to the NIST PQC requirement [34]), DGM has bit security that is 44 bits less than that of the utilized Merkle signing scheme, i.e. 212 bit security when instantiated with XMSS-T-SHA2 at 256 bit security.
- We propose a DGM variant that mitigates the described multi-target attacks and show that such a variant maintains the same bit security as the utilized Merkle signing scheme.

2 Preliminaries

In this section, we provide the security definitions of hash functions that will be used throughout the paper and introduce the notion of unforgeability in GSSs.

In addition to the standard one wayness, and strong and weak collision resistance security notions, we consider the security notions of hash function families introduced in [24, 9]. In what follows, let $n \in \mathbb{N}$ be the security parameter, $k = poly(n)$, $m = poly(n)$, $\mathcal{H}_n = \{H_K(M) : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$ be a keyed hash function family where $K \in \{0,1\}^k$ is the hash key and $M \in \{0,1\}^m$ is the message. Hash-based signature schemes usually adopt parameterized hash functions with $m, k \geq n$. Note that the success probability of quantum adversaries assumes a Quantum Accessible Random Oracle Model [7].

Definition 1 (Post-Quantum) Distinct-function, Multi-target Second Preimage Resistance (PQ-DM-SPR) *Given a (quantum) adversary $\mathcal{A}$ who is provided with p message-key pairs (M_i, K_i), $1 \leq i \leq p$, the success probability that $\mathcal{A}$ finds a second preimage of a pair (j), $1 \leq j \leq p$ using the corresponding hash function key (K_j) is given by,*

$$\begin{aligned} Succ^{\mathsf{PQ\text{-}DM\text{-}SPR}}_{\mathcal{H}_n,p}(\mathcal{A}) = \mathsf{Pr}[K_i \leftarrow \{0,1\}^k; M_i \leftarrow \{0,1\}^m, 1 \leq i \leq p; \\ (j, M') \leftarrow \mathcal{A}((K_1, M_1), \ldots, (K_p, M_p)) : \\ M' \neq M_j \wedge H(K_j, M_j) = H(K_j, M')] \end{aligned}$$

A generic attack by a classical (resp. quantum) DM-SPR adversary who makes q_h queries to an n bit hash function has success probability of $\frac{q_h+1}{2^n}$ (resp. $\Theta(\frac{(q_h+1)^2}{2^n})$). Note that if the keys of the hash function family are chosen randomly, then the security notion in Definition 1 is referred to as *Multi-Function, Multi-target Second-Preimage Resistance (MM-SPR).*

Definition 2 (Post-Quantum) Multi-target Extended Target Collision Resistance (PQ-M-eTCR) *Given a (quantum) adversary $\mathcal{A}$ who is given a target set of p key-message pairs (K_i, M_i), $1 \leq i \leq p$ and is required to find a different message-key pair (with possibly the same key), whose image collides with any of the pairs in the target set, the success probability of $\mathcal{A}$ is given by,*

$$\begin{aligned} \mathsf{Succ}^{\mathsf{PQ\text{-}M\text{-}eTCR}}_{\mathcal{H}_n,p}(\mathcal{A}) = \mathsf{Pr}[K_i \leftarrow \{0,1\}^k; M_i \leftarrow \{0,1\}^m, 1 \leq i \leq p; \\ (j, K', M') \leftarrow \mathcal{A}((K_1, M_1), \ldots, (K_p, M_p)) : \\ M' \neq M_j \wedge H(K_j, M_j) = H(K', M')] \end{aligned}$$

A generic attack by a classical (quantum) M-eTCR adversary who is given p targets and makes q_h queries to an n bit hash function has a success probability of $\frac{p(q_h+1)}{2^n} + \frac{pq_h}{2^k}$ (resp. $\Theta(\frac{p(q_h+1)^2}{2^n} + \frac{pq_h^2}{2^k})$) when $k \geq n$.

Definition 3 ((Post-Quantum) M-eTCR with Nonce (PQ-NM-eTCR)) *Given a (quantum) adversary $\mathcal{A}$ who is given a target set of p key-message-nonce tuples (K_i, M_i, i), $1 \leq i \leq p$, and are required to find a different key-message-nonce tuple (K', M', j) whose image collides with the j-th tuple in the target set*

(with possibly the same key), the success probability of $\mathcal{A}$ is given by,

$$\mathsf{Succ}^{\mathsf{PQ\text{-}NM\text{-}eTCR}}_{\mathcal{H}_n,p}(\mathcal{A}) = \mathsf{Pr}[K_i \leftarrow \{0,1\}^k; M_i \leftarrow \{0,1\}^m, 1 \leq i \leq p; \\ (K', M', j) \leftarrow \mathcal{A}((K_1, M_1, 1), \ldots, (K_p, M_p, p)) : \\ M' \neq M_j \wedge H(K_j||j, M_j) = H(K'||j, M')]$$

A generic attack by a classical (quantum) NM-eTCR adversary who is given p targets and makes q_h queries to an n bit hash function has a success probability of $\frac{(q_h+p)}{2^n} + \frac{pq_h}{2^k}$ (resp. $\Theta(\frac{(q_h+p)^2}{2^n} + \frac{pq_h^2}{2^k})$) when $k \geq n$

Definition 4 ((Post Quantum) Pseudorandom Function (PQ-PRF)) *$\mathcal{H}_n$ is called a PRF function family if it is efficiently computable and for any (quantum) adversary $\mathcal{A}$ who can query a black-box oracle $\mathcal{O}$ that is initialized with either $\mathcal{H}_n$ function or a random function $\mathcal{G}$, where $\mathcal{G} : \{0,1\}^m \rightarrow \{0,1\}^n$, the success probability of $\mathcal{A}$ distinguishing the output of $\mathcal{O}$ by determining which function it is initialized with, is negligible. Such a success probability is given by,*

$$\mathsf{Succ}^{\mathsf{PQ\text{-}PRF}}_{\mathcal{H}_n}(\mathcal{A}) = \mid \mathsf{Pr}[\mathcal{O} \leftarrow \mathcal{H}_n : \mathcal{A}^{\mathcal{O}(\cdot)} = 1] - \mathsf{Pr}[\mathcal{O} \leftarrow \mathcal{G} : \mathcal{A}^{\mathcal{O}(\cdot)} = 1] \mid$$

A generic attack by a classical (resp. quantum) PQ-PRF adversary who makes q_h queries to an $\mathcal{H}_n$ has a success probability of $\frac{q_h+1}{2^n}$ (resp. $\Theta(\frac{(q_h+1)^2}{2^n})$).

Unforgeability in group signature schemes. A basic security notion of a (group) digital signature scheme is that signatures cannot be forged. More precisely, it is computationally infeasible for an adversary $\mathcal{A}$ who does not know the secret key and is allowed unrestricted queries to the signing oracle to generate a message signature pair (M', Σ') that passes as valid by the verification algorithm.

In what follows, we give the definition of the unforgeability game $\mathsf{EXP}^{forge}_{\mathcal{GS},\mathcal{A}}(n, N)$ for a group signature scheme, $\mathcal{GS}$, with N members and security parameter n. Such a game was described by Bellare *et al.* in [5] as an adaptation from the traceability game where $\mathcal{A}$ is not allowed to corrupt members. Intuitively, $\mathcal{A}$ is successful in winning $\mathsf{EXP}^{forge}_{\mathcal{GS},\mathcal{A}}$ if the forged message is either traced to a group member or cannot be traced to a member.

Definition 5 (Unforgeability) *A group signature scheme $\mathcal{GS}$ is unforgeable if for any ppt adversary $\mathcal{A}$ that is given unrestricted access to the signing and opening oracles, $\mathcal{A}$ cannot generate a valid signature for a message that was not previously queried. $\mathcal{A}$ has a negligible advantage in the experiment $Exp^{forge}_{\mathcal{GS},\mathcal{A}}$ as given in Figure 1*

$$Adv^{forge}_{\mathcal{GS},\mathcal{A}}(n, N) = \mid Pr[Exp^{forge}_{\mathcal{GS},\mathcal{A}}(n, N) = 1] \mid \leq negl(n)$$

3 Specification of Related Schemes

In this section, we provide a brief description of XMSS-T, GM and DGM, the related signing schemes used throughout this paper. Details are given only for the procedures that are relevant to our analysis. For more information, the reader is referred to [24, 22, 18, 12].

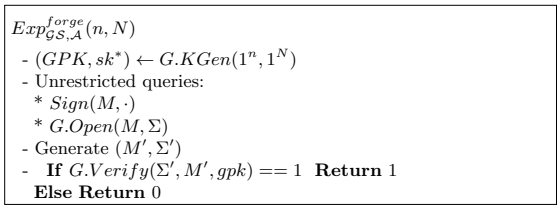

Fig. 1: Unforgeability experiment

3.1 Extended Merkle Signature Scheme-Tightened (XMSS-T)

XMSS-T is a multi Merkle tree construction where the tree leaf nodes are the public keys of the Winternitz One-Time Signature Scheme with Tightened security (WOTS-T) [10]. In what follows, we consider the specification of one tree instance of XMSS-T because this is used in GM and DGM. XMSS-T has a public addressing mapping scheme, $ADRS$, that maps a public seed, $pk.seed$, a leaf/internal node index, i, and a level, j, to generate a (distinct) new hash randomizer, r, and bit-mask, q, for each hash call in the scheme (the hashing in the WOTS-T scheme and the Merkel tree hashing). Such a distinct randomizer enables the scheme to mitigate multi-target attacks. Precisely, a Merkle tree of height h has 2^h leaf nodes (WOTS-T.pk) and the i-th node at level j is denoted by $X_{i,j}$, where $0 \leq i < 2^{h-j}, 0 \leq j \leq h$. The internal nodes are generated by $X_{i,j} = H(r_{i,j}, (X_{2i,j-1}||X_{2i+1,j-1}) \oplus q_{i,j})$, where $r_{i,j}$ and $q_{i,j}$ are the hash randomizer and bit-mask generated by the addressing scheme $(r_{i,j}, q_{i,j}) \leftarrow ADRS(pk.seed, i, j)$. The XMSS-T addressing scheme (see Appendix A for details), takes the leaf index, i, and calculates j according to the hashing sub-structure i.e., OTS hash chains, L-tree hashing or Merkle tree hashing. Then, it generates the required hash randomizer and bit mask. For simplicity, $ADRS$ takes the node level j as input.

The nodes at level 0, $X_{i,0}$, are the leaf nodes, and they are the public keys of the WOTS-T which also utilizes the addressing scheme, $ADRS$, to evaluate the required hash randomizers and bit masks for its hashing. For details of the WOTS-T signing scheme and addressing schemes, the reader is referred to [24] and Appendix A, respectively. Figure 2 depicts a simplified example of XMSS-T with one tree of 8 signing leaves $L_0, \cdots, L_7$. A signature by leaf L_2 (colored in black) has all the gray nodes in its authentication path.

3.2 Group Merkle (GM)

Group Merkle (GM) is the first post-quantum hash-based group signature scheme. It is a one Merkle tree construction that can be instantiated by any stateful one-tree Merkle hash-based signature scheme that employs a One-Time Signing (OTS) scheme as the underlying signing algorithm. The group manager in GM is responsible for the setup procedure of N group members. In this phase, a member j, $1 \leq j \leq N$, generates their own B OTS keys and sends the corresponding public keys $(OTS.pk_{(j-1)B+1}, OTS.pk_{(j-1)B+2}, \ldots, OTS.pk_{jB})$ to the group manager who labels all the received NB keys from the N members by

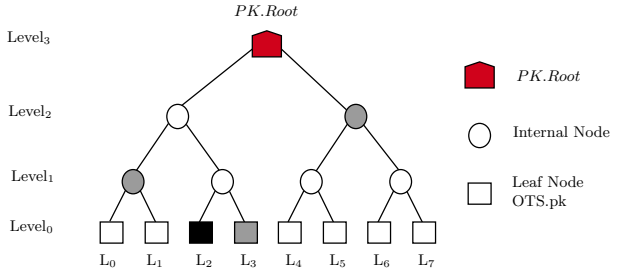

Fig. 2: A one layer XMSS-T, where the leaf nodes are the WOTS-T public keys. The nodes colored in gray are the authentication path for signing with leaf node L_2.

$(1, 2, \cdots, NB)$, where each consecutive $(j-1)B+1, (j-1)B+2, \ldots, jB$ set of keys belongs to the j-th member.

To ensure signer anonymity, the OTS public keys are shuffled by encrypting the corresponding labels by a symmetric encryption algorithm, $pos_i = Enc(i, sk_{gm})$, where sk_{gm} is the group manager's secret key, and $1 \leq i \leq NB$. Thus, the group manager has the pairs $(OTS.pk_1, pos_1), \cdots, (OTS.pk_{N \cdot B}, pos_{N \cdot B})$. These pairs are reordered in ascending order of the encrypted positions to perform the pair permutation. Then, the GM tree is constructed where a leaf node, denoted by $L_i = X_{i,0}$ contains the pair $(OTS.pk_j, pos_j)$ and i is the new permuted position of $OTS.pk_j$. Accordingly, the p-th node at level 1 is calculated by $X_{p,1} = H(X_{2p,0}||X_{2p+1,0}) = H(OTS.pk_x, pos_x||OTS.pk_z, pos_z)$ for $0 \leq p \leq \frac{NB}{2} - 1$, i.e. $L_{2p} = X_{2p,0} = (OTS.pk_x, pos_x)$ and $L_{2p+1} = X_{2p+1,0} = (OTS.pk_z, pos_z)$, because after the permutation, position x is mapped to $2p$ and position z is mapped to $2p+1$. Hashing neighboring nodes continues up the levels until the tree root is evaluated which is the group public key $GM.gpk$. Note that the encrypted position is included in the signature, and is used by to group manager to reveal the identity of the signer. Figure 3 shows a simplified example of a GM tree of two members colored in red and blue where each has 2 OTS key pairs.

3.3 Dynamic Group Merkle (DGM)

DGM [12] combines two types of Merkle trees, one Initial Merkle Tree (IMT) and multiple Signing Merkle Trees (SMTs). The IMT has height 20 and random values for its leaves in order to build the tree whose root is the group public key $DGM.gpk$. The SMTs have variable height and their leaves are OTSs which are used by group members to sign messages. Initially, a group member asks the group manager for B OTS signing keys the group manager randomly chooses B internal nodes from the IMT, i.e. nodes at levels $1, 2, \ldots, 19$, and assigns an OTS from each SMT that is linked to these internal nodes. If all the OTSs of an existing SMT are assigned or an IMT internal node does not have an SMT yet, then a new SMT is generated. The height of an SMT is equal to the level of the internal node that it is linked to.

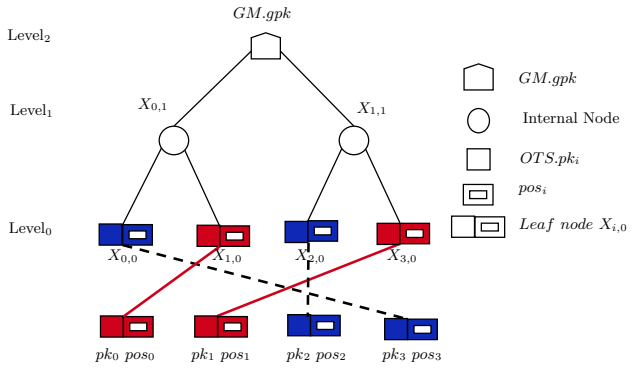

Fig. 3: GM with two members colored in red and blue, each of which has two signing leaves, The leaf Permutation is done by sorting the encrypted positions.

SMT generation. The SMT is constructed in the same manner as the GM tree. However, in DGM, the OTS secret and public key pairs are generated by the group manager and the whole SMT is built without input from the group members. Let $OTS.pk_i$ denote the i-th OTS public key, $0 \leq i \leq z$, where z denotes the total number of signatures supported by the scheme. Let $i = (v, u)$ where u denotes the OTS number within the v-th SMT.

All the OTS public keys are generated and indexed by $DGM.i = (v, u)$. Such indexes are then encrypted with a symmetric encryption algorithm to generate $DGM.pos_i = Enc(DGM.i, sk_{gm})$, where sk_{gm} is the group manger secret key. The OTS public keys are then shuffled by sorting the encrypted positions $DGM.pos$. Afterwards, the SMT leaves are generated, precisely, the j-th SMT leaf node is the hashing of the concatenation of the i-th OTS public key and their encrypted position, $L_j = H(OTS.pk_i||DGM.pos_i)$, where j is the new position of the the i-th OTS after the permutation. These leaves are used to build the SMT and evaluate its root r_{SMT} which is then linked to an IMT internal node called the fallback node, Fn, using a symmetric encryption algorithm. More precisely, r_{SMT}, is linked to Fn by evaluating the fallback key as $Fk = Dec(Fn, r_{SMT})$ which is included in the signature. Note that the verifier has to communicate with the group manager to check the validity of the received Fk and then calculate $Fn = Enc(Fk, r_{SMT})$ to complete the verification process. After all the leaves of an SMT are used, a new SMT is generated and linked to the same Fn. Note that different SMTs linked to the same fallback node Fn have different fallback keys.

Figure 4 depicts a simplified DGM example where the IMT, colored in blue, has height 4. The figure has one SMT colored in red which is linked to the IMT first internal node at level 3, $Fn = X_{0,3}$. When L_2 is used to sign a message M, the resulting signature is given by $\Sigma = (indx, OTS.\sigma_{indx}, DGM.pos_i, Auth)$, where $indx = 2$ is the signing leaf index with respect to the IMT to enable calculating which node is concatenated on its right and left in both the SMT and

IMT from the authentication path in the verification process. $OTS.\sigma_{indx}$ denotes the OTS signature by the leaf index $indx$, $Auth = Auth_{SMT}, Fk, Auth_{IMT}$, where $Auth_{SMT} = L_3, SMT.X_{0,1}, SMT.X_{1,2}$ is the SMT authentication path (colored in pink), and $Auth_{IMT} = IMT.X_{1,3}$, colored in light blue, is the IMT authentication path for the fallback node Fn.

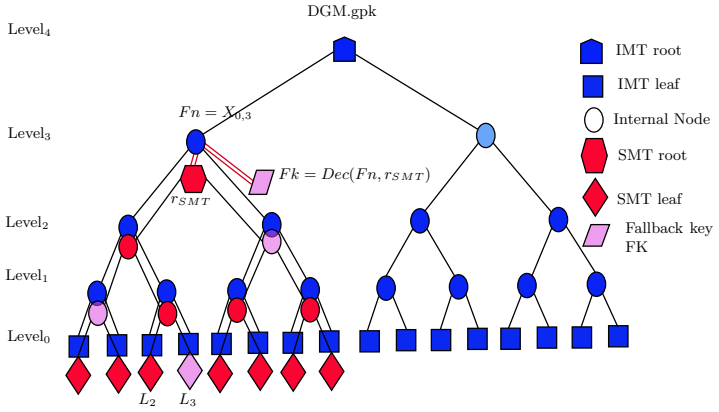

Fig. 4: DGM Example.

4 Instantiating GM And DGM with XMSS-T

In both GM and DGM, the signing leaves which contain the public keys of the OTS used by the group members are first generated and then permuted. Afterwards, the Merkle tree (SMT in DGM) is built using the permuted leaves. In GM, the group members generate their own OTS keys and send the corresponding OTS public keys to the group manager who permutes them and then builds the GM tree. Finally, the group manager distributes all GM tree signing leaves to all the members. On the other hand, in DGM, the group manager generates the OTSs on behalf of the group members, evaluates the signing leaves and permutes them, then constructs the SMTs, and assigns OTSs at random from randomly chosen SMTs.

XMSS-T is the latest stateful MSS variant and has (almost) optimal parameters when compared to other MSS variants which translates to smaller signatures. For instance, with the same parameters (SHA-256 hash function, Winternetiz parameter w=16, tree height 20), the bit security of XMSS-T (resp. XMSS [11]) is 256 (resp. 196). With a bit security of 196, XMSS-T (resp. XMSS) has a signature size of 14,328 (resp. 22,296) bits. XMSS-T uses WOTS-T as the underlying OTS signing scheme which requires the signing leaf index, i, within the Merkle tree to generate the OTS public keys. More precisely, XMSS-T uses an addressing scheme that utilizes the signing leaf index within the Merkle tree as input to generate a distinct hash randomizer and bit mask for each hash call in the hash chains of WOTS-T [24] (see Appendix A). These hash randomizers and

bit masks are used in evaluating the WOTS-T public keys which represent the signing leaves (see Section 3.1).

Instantiating GM and DGM by XMSS-T is not directly achievable because in the specifications of these schemes, a signing leaf index, i, is known only after its corresponding OTS public key has been generated and the associated leaf permuted, while in XMSS-T, WOTS-T requires the leaf index i to evaluate the OTS public key and generate its corresponding leaf. One solution is to employ an earlier XMSS version with an OTS variant that does not require the position of the leaf within the Merkle tree to evaluate the OTS public keys. Such a solution results in using OTS with larger parameters than WOTS-T which negatively affects the performance of the group signature scheme.

GM and DGM with XMSS-T. We provide a tweak in the the setup phase of both GM and DGM which enables their instantiation with XMSS-T. In GM, the setup phase is interactive so we add an extra communication step between the group manager and the group members where the permuted indexes are first sent to the members who can then generate their WOTS-T public keys. More precisely, the permutation in GM is done by encrypting a given position that is associated with an OTS public key, but the encryption itself is independent from the value of the public key, i.e. $pos_i = Enc(i, sk_{gm})$. Accordingly, the group manger can initially permute the indexes of the leaves for all group members before the OTS keys are generated. Afterwards, the permuted indexes are assigned to group members in a manner similar to the original setup phase (see Section 3.2). Each group member uses the assigned indexes within the whole tree as an input to the WOTS-T addressing scheme, $ADRS$, to generate the required hash randomizers and bit masks which are required to generate their WOTS-T public keys. Finally, the WOTS-T public keys are sent back to the group manager who constructs the GM tree using XMSS-T.

In DGM, no extra communication is needed because the group manger generates the OTS signing keys for the group members and their corresponding public keys. Accordingly, the manager may first permute the indexes using symmetric encryption then generate the OTS public keys using the permuted indexes. In other words, the specification of the setup phase stays the same with only the permutation and OTS key generation order swapped.

5 DGM with XMSS-T Security Analysis

In [12], DGM was analyzed with respect to security notions of group digital signature schemes, i.e. anonymity and traceability. However, since DGM was not instantiated with a specific Merkle signing scheme, no bit security analysis for its unforgeability was provided. In this section, we analyze the bit security of the unforgeability of DGM when it is instantiated with XMSS-T. Note that the same analysis is valid if DGM is instantiated with earlier XMSS versions. Henceforth, we refer to DGM when instantiated with XMSS-T as simply DGM.

5.1 Multi-target attacks and XMSS-T

If an n bit hash function is used once in a cryptographic primitive with a security parameter λ whose security is dependent on the second preimage resistance of the

hash function, then finding a second preimage of the generated digest requires 2^n computations, thus it suffices that $n = \lambda$. However, if the same hash function is used t times in the cryptographic primitive, i.e. an adversary has access to t digests generated with the same hash function, then a second preimage may be obtained on any of these t targets with $2^n/t$ computations. Assuming that $n = \lambda$, the security of the scheme is reduced from n to $n - \log t$. A naive remedy to reach n bit security is to use message digests of length $n + \log t$. Alternatively, one may enforce that each hash application is different such that each digest for the t targets is evaluated using a different hash function so that finding a second preimage for any function, i.e. using the same hash key, requires 2^n computations.

In XMSS-T, the addressing scheme, $ADRS$, generates a hash randomizer and bit mask for each hash function call depending on the hash node index in the tree or WOTS-T chain iteration. For a tree with height h, the i-th node at level j is denoted by $X_{i,j}$ where $0 \leq i < 2^{h-j}$, $0 \leq j \leq h$. $ADRS$ is given by $(r_{i,j}, q_{i,j}) \leftarrow ADRS(pk.seed, i, j)$ where $r_{i,j}$ and $q_{i,j}$ are the hash randomizer and bit mask used. The internal nodes are generated as $X_{i,j} = H(r_{i,j}, (X_{2i,j-1}||X_{2i+1,j-1}) \oplus q_{i,j})$, i.e., $H_{r_{i,j}}$ is unique for $X_{i,j}$. Accordingly, if an adversary collects all the signatures supported by the scheme, each element in the WOTS-T signatures and each node in any authentication path is generated by a different hash function. Consequently, finding a forgery requires finding a second preimage of a given node using the corresponding hash function where other nodes are no longer applicable targets.

5.2 Multi-target attacks on DGM

DGM allows multiple SMT trees to branch out of any IMT internal node, fallback node. Accordingly, one may regard DGM as several overlapping parallel trees with heights ranging from 1 to 20. The IMT tree is the only tree with height 20 and the SMTs have heights ranging from 1 to 19. To visualize such a structure, Figure 5 depicts a reduced DGM instance with an IMT, colored in blue, of height 4 and 42 SMTs, colored red, yellow and green. We assume a uniform distribution in the selection of the IMT internal nodes from which keys are assigned from the linked SMTs. Hence, each IMT internal node has the same number of assigned OTS keys (i.e. leaf nodes), and the number of SMTs per node in level j is double the number of SMTs per node in level $(j + 1)$. There are 4, 2, and 1 SMTs branching out from internal IMT nodes at levels 1, 2, and 3, respectively, and their respective colors are green, yellow, and red. This simplified example has 112 signing leaves which can be used to sign 112 messages under the same public key (IMT root). Note that there is no maximum number of SMTs so if more signing leaves are needed, new SMTs can be constructed and linked to a random internal node.

Following the NIST PQC recommendation, a signature scheme should be secure to sign up to 2^{64} messages under the same public key [33]. In what follows, we assume that DGM is used to sign 2^{64} messages. According to the design specifications, when a group member needs B signing keys (leaves), the group

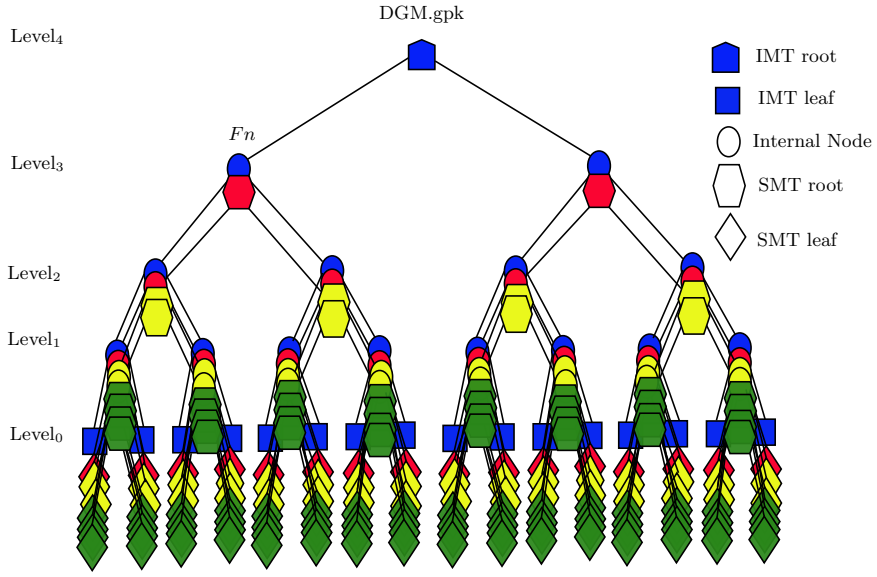

Fig. 5: Simplified DGM example of height 4 with 42 SMTs, 112 signing leaves, and fallback nodes uniformly distributed across the internal IMT nodes

manager randomly selects B internal nodes of the IMT and assigns to that member the next unassigned OTS of each SMT linked to that internal node. The total number of internal nodes excluding the root in an IMT of height 20 is $2^{19} + 2^{18} + \cdots + 4 + 2 = 2^{20} - 2$. Recall that if the SMT OTS leaves linked to any randomly chosen internal node are used up, then a new SMT tree is generated, linked to that fallback node and one of its leaves is assigned. Accordingly, assuming a uniform distribution in the random fallback node selection, to assign 2^{64} OTSs to all group members, each IMT internal node is chosen $2^{64}/(2^{20} - 2) > 2^{44}$ times. This means that each IMT internal node at level j, $1 \leq j \leq 19$, has $2^{44-(j-1)-1} = 2^{44-j}$ SMT trees each of height j, i.e. 2^{43} SMTs of height 1 for each IMT internal node at level 1, 2^{42} SMTs of height 2 for each IMT internal node at level 2, up to 2^{25} SMTs of height 19 for each IMT internal node at level 19.

When DGM is instantiated with XMSS-T, to enable verification of a given signature, a DGM instance is seen as one tree of height 20 which means that wherever the signing SMT is located with respect to the IMT, the leaf indexing is in the set $\{0, 1, \ldots, 2^{20} - 1\}$, i.e. leaf indexing is considered relevant to the IMT where the signing SMT is considered a part of the IMT. Such an indexing restriction is required to enable the verifier to evaluate the position of the nodes in the authentication path of the IMT up to its root (the pale blue nodes in Figure 4), which is essential in determining which nodes are concatenated on its right and left. Consequently, different SMTs that are linked to the same IMT internal node have the same indexing, and accordingly their parallel nodes at

the same position are evaluated with the same hash function, i.e. the same hash randomizer and bit mask. For instance, in Figure 5, any 4 green SMT roots branching from the same level 1 IMT blue node are evaluated with the same hash function as they share the same index within the IMT. Moreover, there are SMT nodes that share the same indexes and nodes of the SMTs that are connected to upper IMT internal nodes, for example, in Figure 5, any 4 green SMT roots at an IMT level one intermediate node share the same indexes with 2 intermediate yellow SMT nodes and one intermediate red SMT node. Therefore, even though XMSS-T is secure against multi-target attacks, employing several parallel instances of it with the same indexing in the form of SMTs makes DGM vulnerable to multi-target attacks. Intuitively, a forgery adversary who collects a set of message-signature pairs, can group them in t-target sets that share common indexes, and then they can find another message whose digest collides with any of the message digests in the set. Note that such sets have t messages with authentication paths that share nodes with the same IMT indexes, so with complexity $2^n/t$ a forgery is obtained.

5.3 DGM bit security

Consider that DGM is used to sign 2^y messages where $y > 20$. Accordingly, each internal IMT node is chosen $2^y/(2^{20} - 2)$ times by the group manager to assign the next available OTS from the linked SMT. Assume an adversary $\mathcal{A}$ is able to collect all 2^y signatures generated by the scheme. The signature given by $\Sigma = (R, indx, OTS.\sigma_{indx}, DGM.pos_i, Auth)$ is signed with the i-th OTS key pair and has index $indx$ relative to its IMT position, i.e. $indx \in \{0, 1, \ldots, 2^{20} - 1\}$ (see Section 3.3). $\mathcal{A}$ can then group the signatures along with their corresponding messages in sets that share the same signing index, $indx$, where each set is expected to have t target message-signatures pairs, i.e. a given target set is denoted by $ts = \{(M_0, \Sigma_0), (M_1, \Sigma_1), \ldots, (M_{t-1}, \Sigma_{t-1})\}$. Assuming a uniform distribution in selecting IMT Fn positions, the number of targets t per set is given by,

$$t = \sum_{j=1}^{j=19} \frac{2^y/(2^{20} - 2)}{2^j} < 2^{y-20} \tag{1}$$

We assume a fully filled tree similar to the example in Figure 5 where all IMT internal nodes have an equal number of assigned leaves, e.g. $2^y/(2^4 - 2) = 112/14 = 8$. Otherwise, the index that has the maximum number of signatures is considered. The maximum number of overlapping SMT nodes is given by $\frac{8}{2} + \frac{8}{2^2} + \frac{8}{2^3} = 7$, so $t = 7$.

In XMSS-T, to sign a message M, its message digest md is initially calculated as $md = H_{msg}(R||DGM.root||indx, M)$ where $H_{msg} : \{H(K, M) : \{0,1\}^m \times \{0,1\}^* \rightarrow \{0,1\}^n$, R is the hash randomizer chosen by the signer and $index$ is the leaf index relative to the IMT. Since ts has t (M, Σ) pairs all with the same $indx$, $\mathcal{A}$ can search for (M', R') pair such that $M' \notin ts$, and the corresponding md' collides with a message digest of any of the messages in ts. Specifically, $\mathcal{A}$

finds (M', R') such that

$$H_{msg}(R'||DGM.root||indx, M') \in \{(H_{msg}(R_0||DGM.root||indx, M_0)), \ldots, (H_{msg}(R_{t-1}||DGM.root||indx, M_{t-1}))\}.$$

Thus, $\mathcal{A}$ can successfully find a forgery for (M', R') with probability $2^{-n+\log_2 t}$. Similar multi-taget attacks can be applied on the OTS public keys or authentication paths in ts. In what follows, we give the security reduction of DGM when it is used to sign 2^y messages with $y > 20$. For completeness and consistency with XMSS-T notation [24], the hash functions used in different contexts within the signature scheme are defined as follows.

- $F : \{F(K, M) : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ used in OTS hash chains
- $H : \{H(K, M) : \{0,1\}^n \times \{0,1\}^{2n} \rightarrow \{0,1\}^n$ used to calculate the Merkle tree hash nodes
- $H_{msg} : \{H(K, M) : \{0,1\}^m \times \{0,1\}^* \rightarrow \{0,1\}^n$ used to calculate the message digests
- $\mathcal{F}_n$ (resp. $\mathcal{F}_m$) is a pseudorandom function family that takes a secret seed as input and outputs the OTS secret keys (resp. the message digest hash randomizer R) each of n bits (resp. m bits $(m = n + y)$).

Theorem 1 *For security parameter $n \in \mathbb{N}$ and parameters y, t as defined above, DGM is unforgeable against an adaptive chosen message attacks if*

- *F and H are PQ-DM-SPR hash function families,*
- *$\mathcal{F}_n$ and $\mathcal{F}_m$ are post-quantum pseudorandom function families, and*
- *H_{msg} is a PQ-NM-eTCR hash function family.*

The insecurity function, $\mathsf{InSec}^{\mathsf{PQ\text{-}forge}}(DGM, \xi, 2^y)$, that describes the maximum success probability over all adversaries running in time $\leq \xi$ against the PQ-forge security of DGM and making a maximum of $qs = 2^y$ queries is bounded by

$$\mathsf{InSec}^{\mathsf{PQ\text{-}forge}}(DGM, \xi, 2^y) \leq \mathsf{InSec}^{\mathsf{PQ\text{-}PRF}}(\mathcal{F}_n, \xi) + \mathsf{InSec}^{\mathsf{PQ\text{-}PRF}}(\mathcal{F}_m, \xi) +$$
$$max[t \times (\mathsf{InSec}^{\mathsf{PQ\text{-}DM\text{-}SPR}}(H, \xi) + \mathsf{InSec}^{\mathsf{PQ\text{-}DM\text{-}SPR}}(F, \xi) + \mathsf{InSec}^{\mathsf{PQ\text{-}NM\text{-}eTCR}}(H_{msg}, \xi))]$$

Proof. The proof is based on the approach of the proof given in [24]. Note that we do not include the proof of $\mathcal{F}_n$ and $\mathcal{F}_m$ with respect to PQ-PRF because they are not affected by instantiating DGM with XMSS-T, hence, the proof is similar to that of XMSS-T in [24]. Assume the adversary $\mathcal{A}$ is allowed to make 2^y queries to a signing oracle running DGM with XMSS-T. $\mathcal{A}$ wins the $\mathsf{EXP}^{forge}_{\mathcal{GS},\mathcal{A}}$, as shown in Figure 1, if they find a valid forgery (M', Σ') where the message M' is not in the queried set of 2^y messages. $\mathcal{A}$ initially groups the signatures that share a given $indx$ in a set ts. Forgery occurs in the following three mutually exclusive cases.

- The message digest of M' under $indx$ results in M' being a second preimage of one of the message digests of the messages in ts. More precisely

$$md = H_{msg}(R'||DGM.root||indx, M') = H_{msg}(R_j||DGM.root||indx, M_j)$$

where $M_j \in ts$. This occurs with success probability $t \times \mathsf{InSec}^{\mathsf{PQ\text{-}NM\text{-}eTCR}}(H_{msg})$ (see Definition 3), i.e. $\mathcal{A}$ is able to break the security of NM-eTCR of the message hash function used, H_{msg}.
- The OTS public key of the forged signature, $OTS.pk'$, exists in the set of OTS public keys of the signatures in ts, i.e. $OTS.pk' \in \{OTS.pk_0, \cdots, OTS.pk_{t-1}\}$. This occurs with success probability $t \times \mathsf{InSec}^{\mathsf{PQ\text{-}DM\text{-}SPR}}(\mathrm{F})$ (see Definition 1), i.e. $\mathcal{A}$ is able to break the security of DM-SPR of hash function F.
- The forged signature contains a node in the authentication path ($X'_{i,j}$, the i-th node in level j), that collides with a node at the same position in the set of authentication paths in ts ($X_{i,j}$, the i-th node in level j), e.g. $H(r_{i,j}, (X'_{2i,j-1}||X'_{2i+1,j-1}) \oplus q_{i,j}) = H(r_{i,j}, (X_{2i,j-1}||X_{2i+1,j-1}) \oplus q_{i,j})$, where the nodes $(X'_{2i,j-1}, X'_{2i+1,j-1})$ are from the forged signature authentication path, the nodes $(X_{2i,j-1}, X_{2i+1,j-1})$ are from an authentication path of a signature in ts, and $r_{i,j}, q_{i,j}$ are the hash randomizer and bit mask used for hashing. This occurs with success probability $t \times \mathsf{InSec}^{\mathsf{PQ\text{-}DM\text{-}SPR}}(\mathrm{H})$ (see Definition 1). Thus, $\mathcal{A}$ is able to break the security of the second preimage resistance of hash function H.

The above proof shows that if DGM is instantiated with the parameters of XMSS-T (RFC 8391), i.e. the message digest length equals the security parameter n, then DGM does not achieve the same bit security level as XMSS-T. In particular, the bit security of DGM decreases by $\log_2 t$ bits compared to that of XMSS-T, so for XMSS-T with security parameter $n = 256$ and DGM used to sign 2^{64} messages, the DGM bit security decreases by $\log_2(\sum_{j=0}^{j=18} 2^{43-j}) = 44$ bits, i.e. DGM achieves 212 bits of security. Therefore, if DGM is required to achieve n bits of security, then XMSS-T should use a hash function with output size $n + \log_2 t$ which decreases the signing performance and increases the signature size. In the following section, we propose a solution that allows DGM to attain optimal parameters whereas XMSS-T attains (almost) optimal parameters [9].

6 DGM$^+$ with Optimal Parameters

In this section we propose DGM$^+$, a DGM-XMSS-T variant that mitigates multi-target attacks (per index) as discussed in Section 5. We modify the addressing scheme such that it outputs different hash randomizers and bit masks for the same hash call location in different SMTs branching from the same IMT internal node, and for overlapped SMTs that share the same indexing for some leaves.

The DGM public parameters contain two values $DGM.root$ and $DGM.seed$, where $DGM.root$ is the IMT root (group public key), and $DGM.seed$ is the public key seed that is used in the XMSS-T addressing scheme to generate the hash randomizers r_i and bit masks q_i for each hash call at address ad_i in the IMT, i.e. $(r_i, q_i) \leftarrow ADRS(DGM.seed, ad_i)$. To enable opening, each SMT leaf has index (v, u) which is encrypted to generate $DGM.pos$, where v is the SMT number and u is the leaf index within the SMT. Note that both u and v are secrets but $DGM.pos$ is not because it is sent in the signature. If we assume

that the bit size of v is equal to the block length, b, of the encryption algorithm used, then we can get ev as the first b bits from $DGM.pos$, where ev denotes the encryption of v. Accordingly, we propose the following.

- IMT uses $DGM.seed$ directly as the seed to generate the hash randomizers and bit masks for each hash call within the IMT.
- Each SMT utilizes (publicly calculated) a different seed, SMT.$seed_v$ for its hash randomizer and bit mask generation. SMT.$seed_v$ is unique for the v-th SMT and is calculated by SMT.$seed_v = PRF(\text{DGM}.seed, ev)$.

For all SMTs that share indexing, we utilize different seed values with each SMT and keep the XMSS-T addressing scheme unchanged [22] (see Section A). Thus, different hash randomizers and bit masks are used at the same IMT location but for different SMTs. Note that for signing, the IMT utilizes DGM.$seed$ in its construction, while the v-th SMT utilizes SMT.$seed_v = PRF(DGM.seed, ev)$ in its construction. Let $SMT.root.level$ denote the level of the fallback node for a given signing SMT. The signature authentication path, $Auth$, contains the whole SMT authentication path, $Auth.SMT_v$ and the top $20 - SMT.root.level$ nodes from the IMT. The latter authentication path starts from the neighboring node of the fallback node linked to the SMT root and up to DGM.$root$.

For verification, the verifier uses two seeds. $DGM.seed$ is used for hash evaluations of the authentication path from the fallback node and up. Moreover, the verifier calculates STM.$seed_v = PRF(DGM.seed, ev)$ which is used in the WOTS-T hash iterations and the SMT authentication path, $Auth.SMT_v$, hash evaluations.

6.1 Message hashing with DM-SPR

It was shown in Section 5 that the security of DGM depends on the ɴM-eTCR of the hash function used where the number of targets, t, is considered per index. We tweak the message hashing such that the security of DGM depends on the DM-SPR of the hash function used (see Definition 1), to prevent multi-targets attacks. This is achieved by using the message hash randomizer $R = F^{w-1}(sk_1)$ as follows

$$md = H_{msg}(R||DGM.root||idx, M) = H_{msg}(F^{w-1}(sk_1)||DGM.root||indx, M)$$

where $F^{w-1}(sk_1)$ is the last iteration, $w-1$, of the first secret key of WOTS-T (see [24] for the details of WOTS-T).

Message hashing tweak rationale. The elements $(R||DGM.root||idx)$ serve as the hash key where R is chosen at random for each new message hashing and $DGM.root$ is fixed. If an adversary $\mathcal{A}$ who has access to the signing oracle is able to get the hash randomizer R before querying the signing oracle, then $\mathcal{A}$ can search to find two messages that have the same image using the same R, i.e. $\mathcal{A}$ looks for a collision. Therefore, $\mathcal{A}$ queries the signing oracle with one message and the other message has the same signature. Nevertheless, as R is chosen randomly and is known to the adversary only after querying the signing oracle, $\mathcal{A}$ works to find a second primage of any of the queried messages when

any hash randomizer R' is used, i.e. for a valid forgery the adversary needs to break the NM-eTCR security of the hash function used.

If we replace the hash randomizer R with the last iteration of the first secret key of the OTS used, $pk_1 = F^{w-1}(sk_1)$ (see [24] for details), then R is not chosen at random and is known publicly only after signing. Accordingly, for a valid forgery, the adversary is restricted to using the same message hash randomizer, $R = F^{w-1}(sk_1)$ (that is sent in the signature), to find a message digest collision with the queried set. Hence, the adversary is required to break the security of MM-SPR of the hash function used which has a lower probability of success than breaking the NM-eTCR security of the hash function. Note that the last chain iteration of the first OTS secret key, $F^{w-1}(sk_1)$, is not a public parameter and is known only after signing with the corresponding leaf node, i.e. it is different than the OTS public key which is the root of the L-Tree (see [24] for the details of the L-Tree).

In the verification procedure, the verifier checks if $pk_1 = F^{w-a_1-1}(\sigma_1)] \stackrel{?}{=} R$, where σ_1 is the first signature element in the OTS signature, otherwise, it returns invalid signature. Accordingly, for a valid forgery the adversary is required to find a second primage using the hash key $F^{w-1}(sk_1)||DGM.root||idx$, i.e. break the MM-SPR of the hash function (see Definition 1).

Note that using the above message hashing to generate R from the OTS public keys may be used to enable XMSS-T [9] to attain optimal parameters. Specifically, when R is bound to a specific signing leaf, it suffices that R is n bits to provide n bit security.

6.2 DGM and DGM$^+$ comparison

This section provides a comparison between DGM and DGM$^+$ when both are instantiated with XMSS-T to provide n bit security and support 2^y messages where $y \geq 20$, and the IMT height is 20.

Secret and public keys sizes. For DGM to achieve n bit security requires a hash output size of $n + \log_2 t$ bits where t is given by Equation 1. Thus, its tree nodes and secret keys will also be $n + \log_2 t$ bits. The DGM public key is the pair $(pk.seed, IMT.root)$ each of $n + \log_2 t$ bits, and the secret key contains $sk.prf$ to generate the message hash randomizer and $sk.seed$ to generate the WOTS-T secret keys. Accordingly, the secret key size is $2(n + \log_2 t)$ bits.

For DGM$^+$, the size of the tree nodes and secret keys is n bits. The DGM$^+$ public key size is $2n$ bits, i.e. $(pk.seed, IMT.root)$ each of n bits. The secret key contains only $sk.seed$ of n bits because it does not require $sk.prf$ as the message hash randomizer is the last hash iteration of the first WOTS-T secret key.

Signature size. A DGM signature contains the message hash randomizer, R, the leaf index, the encrypted position, the WOTS-T signature, the authentication path, and the fallback key. The signature element sizes in DGM$^+$ is n bits while in DGM it is $n + \log_2 t$ bits. This has another impact as the message digest size is increased, the number of WOTS-T elements, l, is increased. This increases both the signature size and the computational cost.

Table 1 provides the size of the keys and signature for both DGM$^+$and DGM at 128, 192, and 256 bit security when they are used to support up to 2^{64} signatures where the signature size is $(22 + l)n + 4$ Bytes and l is the number of elements in the OTS signature. The index is 4 Bytes and we consider the encrypted position and message hash randomizer, R, equal to the node size in the scheme.

Table 1: DGM and DGM$^+$ keys and signature sizes in Bytes at 128, 192, and 256 bit security and 2^{64} signatures.

Algorithm	bit security	node size	pk	sk	l	signature size
DGM	128	22	44	44	47	1522
	192	30	60	60	63	2554
	256	38	76	76	79	3842
DGM$^+$	128	16	32	16	35	916
	192	24	48	24	51	1756
	256	32	64	32	67	2852

7 Conclusion

In this paper, we discussed the challenges of instantiating GM and DGM with XMSS-T and provided a tweak in the setup phases of GM and DGM to overcome the discussed challenges. Moreover, we analyzed the bit security of DGM when instantiated with XMSS-T and showed that because of the parallel multiple XMSS-T instances construction, DGM is vulnerable to multi-target attacks that may enable forgery with 44 bits less effort than that of XMSS-T when the scheme is used to sign 2^{64} messages. Finally, we proposed a solution that mitigates these multi-target attacks and presented a new message hashing mechanism that reduces the associated signature and secret key sizes.

Acknowledgment: The authors would like to thank the reviewers for their valuable comments that helped improve the quality of the paper.

References

[1] Alamélou, Q., Blazy, O., Cauchie, S., and Gaborit, P. A practical group signature scheme based on rank metric. In *International Workshop on the Arithmetic of Finite Fields* (2016), Springer, pp. 258–275.

[2] Alamélou, Q., Blazy, O., Cauchie, S., and Gaborit, P. A code-based group signature scheme. *Designs, Codes and Cryptography 82*, 1-2 (2017), 469–493.

[3] AlTawy, R., and Gong, G. Mesh: A supply chain solution with locally private blockchain transactions. *Proceedings on Privacy Enhancing Technologies 2019*, 3 (2019), 149–169.

[4] Ayebie, B. E., Assidi, H., and Souidi, E. M. A new dynamic code-based group signature scheme. In *International Conference on Codes, Cryptology, and Information Security* (2017), Springer, pp. 346–364.

[5] Bellare, M., Micciancio, D., and Warinschi, B. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *International Conference on the Theory and Applications of Cryptographic Techniques* (2003), Springer, pp. 614–629.

[6] Boneh, D., Boyen, X., and Shacham, H. Short group signatures. In *International Cryptology Conference* (2004), Springer, pp. 41–55.

[7] Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., and Zhandry, M. Random oracles in a quantum world. In *International Conference on the Theory and Application of Cryptology and Information Security* (2011), Springer, pp. 41–69.

[8] Boneh, D., and Shacham, H. Group signatures with verifier-local revocation. In *Proceedings of the ACM Conference on Computer and Communications Security* (2004), pp. 168–177.

[9] Bos, J. W., Hülsing, A., Renes, J., and van Vredendaal, C. Rapidly verifiable XMSS signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021), 137–168.

[10] Buchmann, J., Dahmen, E., Ereth, S., Hülsing, A., and Rückert, M. On the security of the Winternitz one-time signature scheme. In *International Conference on Cryptology in Africa* (2011), Springer, pp. 363–378.

[11] Buchmann, J., Dahmen, E., and Hülsing, A. XMSS-A practical forward secure signature scheme based on minimal security assumptions. In *International Workshop on Post-Quantum Cryptography* (2011), Springer, pp. 117–129.

[12] Buser, M., Liu, J. K., Steinfeld, R., Sakzad, A., and Sun, S.-F. DGM: A dynamic and revocable group merkle signature. In *European Symposium on Research in Computer Security* (2019), Springer, pp. 194–214.

[13] Camenisch, J., and Groth, J. Group signatures: Better efficiency and new theoretical aspects. In *International Conference on Security in Communication Networks* (2004), Springer, pp. 120–133.

[14] Camenisch, J., and Lysyanskaya, A. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *International Cryptology Conference* (2002), Springer, pp. 61–76.

[15] Camenisch, J., and Lysyanskaya, A. Signature schemes and anonymous credentials from bilinear maps. In *International Cryptology Conference* (2004), Springer, pp. 56–72.

[16] Chaum, D., and Van Heyst, E. Group signatures. In *Workshop on the Theory and Application of Cryptographic Techniques* (1991), Springer, pp. 257–265.

[17] Del Pino, R., Lyubashevsky, V., and Seiler, G. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security* (2018), pp. 574–591.

[18] El Bansarkhani, R., and Misoczki, R. G-Merkle: A hash-based group signature scheme from standard assumptions. In *International Conference on Post-Quantum Cryptography* (2018), Springer, pp. 441–463.

[19] Ezerman, M. F., Lee, H. T., Ling, S., Nguyen, K., and Wang, H. Provably secure group signature schemes from code-based assumptions. *IEEE Transactions on Information Theory 66*, 9 (2020), 5754–5773.

[20] Gordon, S. D., Katz, J., and Vaikuntanathan, V. A group signature scheme from lattice assumptions. In *International Conference on the Theory and Application of Cryptology and Information Security* (2010), Springer, pp. 395–412.

[21] Hülsing, A., Busold, C., and Buchmann, J. Forward secure signatures on smart cards. In *International Conference on Selected Areas in Cryptography* (2012), Springer, pp. 66–80.

[22] Hülsing, A., Butin, D., Gazdag, S.-L., Rijneveld, J., and Mohaisen, A. XMSS: Extended Merkle signature scheme. In *RFC 8391*. IRTF, 2018.

[23] Hülsing, A., Rausch, L., and Buchmann, J. Optimal parameters for XMSS-MT. In *International Conference on Availability, Reliability, and Security* (2013), Springer, pp. 194–208.

[24] Hülsing, A., Rijneveld, J., and Song, F. Mitigating multi-target attacks in hash-based signatures. In *Public-Key Cryptography*. Springer, 2016, pp. 387–416.

[25] Laguillaumie, F., Langlois, A., Libert, B., and Stehlé, D. Lattice-based group signatures with logarithmic signature size. In *International Conference on the Theory and Application of Cryptology and Information Security* (2013), Springer, pp. 41–61.

[26] Langlois, A., Ling, S., Nguyen, K., and Wang, H. Lattice-based group signature scheme with verifier-local revocation. In *International Workshop on Public Key Cryptography* (2014), Springer, pp. 345–361.

[27] Libert, B., Ling, S., Mouhartem, F., Nguyen, K., and Wang, H. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *International Conference on the Theory and Application of Cryptology and Information Security* (2016), Springer, pp. 373–403.

[28] Libert, B., Peters, T., and Yung, M. Group signatures with almost-for-free revocation. In *Cryptology Conference* (2012), Springer, pp. 571–589.

[29] Libert, B., Peters, T., and Yung, M. Scalable group signatures with revocation. In *International Conference on the Theory and Applications of Cryptographic Techniques* (2012), Springer, pp. 609–627.

[30] Ling, S., Nguyen, K., and Wang, H. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In *IACR International Workshop on Public Key Cryptography* (2015), Springer, pp. 427–449.

[31] Merkle, R. C. A certified digital signature. In *Conference on the Theory and Application of Cryptology* (1989), Springer, pp. 218–238.

[32] Nguyen, P. Q., Zhang, J., and Zhang, Z. Simpler efficient group signatures from lattices. In *IACR International Workshop on Public Key Cryptography* (2015), Springer, pp. 401–426.

[33] NIST. Post-quantum cryptography project. http://csrc.nist.gov/groups/ST/post-quantum-crypto.

[34] NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/Call-for-Proposals.

[35] Traoré, J. Group signatures and their relevance to privacy-protecting offline electronic cash systems. In *Australasian Conference on Information Security and Privacy* (1999), Springer, pp. 228–243.

[36] Yang, R., Au, M. H., Zhang, Z., Xu, Q., Yu, Z., and Whyte, W. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In *International Cryptology Conference* (2019), Springer, pp. 147–175.

A XMSS-T Addressing Scheme

XMSS-T utilizes a hash function addressing scheme that enumerates each hash call in the scheme and outputs a distinct hash randomizer r and bit mask q for each hash call to mitigate multi-target attacks [22]. XMSS-T has three main substructures, WOTS-T, L-tree, and Merkle tree hash. The first substructure requires for each hash call a hash randomizer and bit mask, each of n bits. The

other two substructures require a hash randomizer of n bits and $2n$ bits for the bit mask. The hash function address consists of 256 bits. There are three address types for the three substructure mentioned above which are described below.

1. WOTS-T hash address: The first field (32 bits) is the tree layer address which indexes a given layer in which the WOTS-T exists (this value is set to zero for DGM). The tree address (64 bits) indexes a tree within the layer (this value is set to zero for DGM), and the addressing type (32 bits) which is equal to zero. The key pair address (32 bits) denotes the index of the WOTS-T within the hash tree. The chain address (32 bits) denotes the number of the WOTS-T secret key on which the chain is applied. The hash address (32 bits) denotes the number of the hash function iterations within a chain. The last field is KeyAndMask (32 bits) which is used to generate two different addresses for one hash function call (it is set to zero to get the hash randomizer R and it is set to one to get the bit mask, each of n bits).

2. L-tree hash address: The first field (32 bits) is the layer address which indexes the layer in which the WOTS-T exists (this value is set to zero for DGM). The tree address (64 bits) indexes a tree within the layer (this value is set to zero for DGM), and the addressing type (32 bits) which is equal to one. The L-tree address (32 bits) denotes the leaf index that is used to sign the message. The tree height (32 bits) encodes the node height in the L-tree, and the tree index (32 bits) refers to the node index within that height. The last field is KeyAndMask (32 bits) which in this substructure is used to generate three different addresses for one hash function call (it is set to zero to get the hash randomizer R, one to get the first bit mask and two to get the second bit mask, each of n bits).

3. Merkle tree hash: The first field (32 bits) is the layer address which indexes the layer in which the WOTS-T exists (this value is set to zero for DGM). The tree address (64 bits) indexes a tree within the layer (this value is set to zero for DGM), and the addressing type (32 bits) which is equal to two. Then a padding of zeros (32 bits). The tree height (32 bits) encodes the node height in the main Merkle tree and the tree index (32 bits) refers to the node index within that height. As the L-tree addressing, the last field is KeyAndMask (32 bits) which is used to generate three different addresses for one hash function call (it is set to zero to get the hash randomizer R, one to get the first bit mask and two to get the second bit mask, each of n bits).