G-Merkle: A Hash-Based Group Signature Scheme From Standard Assumptions

Rachid El Bansarkhani

Technische Universität Darmstadt, Germany

elbansarkhani@cdc.informatik.tu-darmstadt.de

Rafael Misoczki

Intel Corporation, USA

Rafael.Misoczki@intel.com

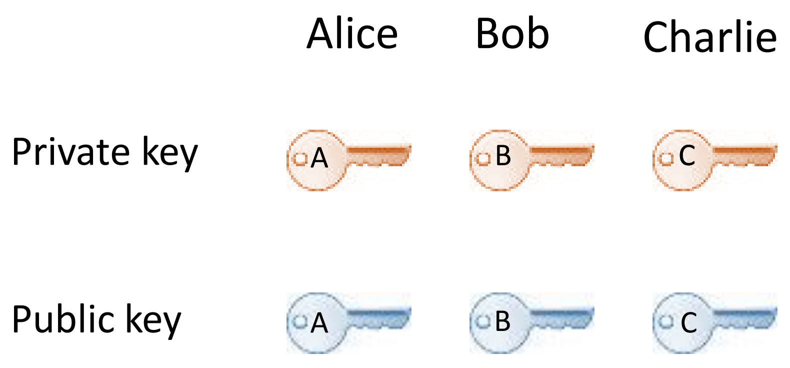
Agenda

- Motivation
- Introduction
  - Group Signatures
  - Hash-Based Signatures
- G-Merkle
  - Definition
  - Security Assessment
  - Performance Results
- Conclusions

Digital Signatures vs. Privacy

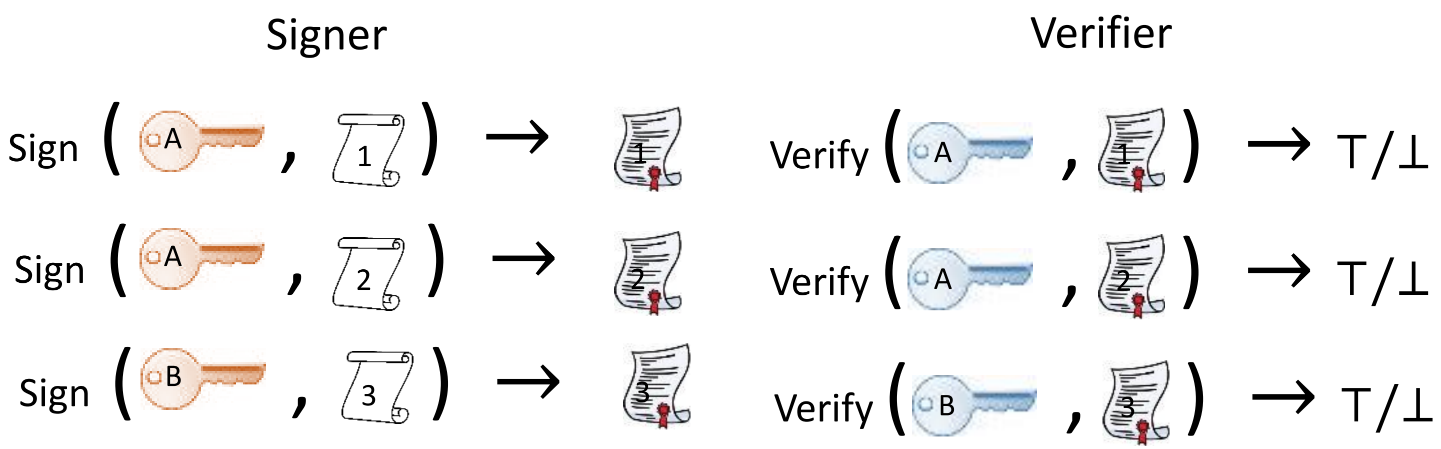
Traditional digital signatures do not offer privacy-preserving features:

Key Setup:



1 key pair per user

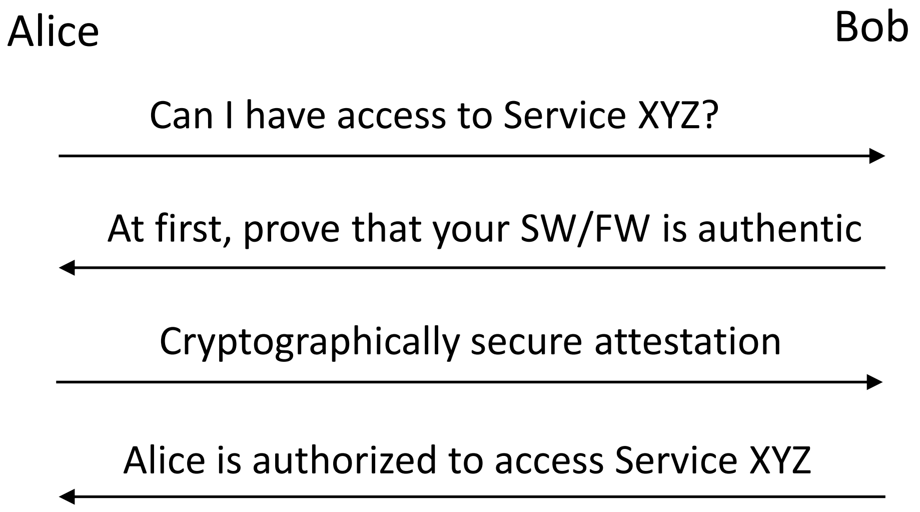
Sign/Verify:



Verifier needs to know the correct public key to verify a given signature (thus able to link signatures, at least)

Remote Attestation

Allows a device to prove the authenticity of its software/firmware



Alice

Bob

Can I have access to Service XYZ?

At first, prove that your SW/FW is authentic

Cryptographically secure attestation

Alice is authorized to access Service XYZ

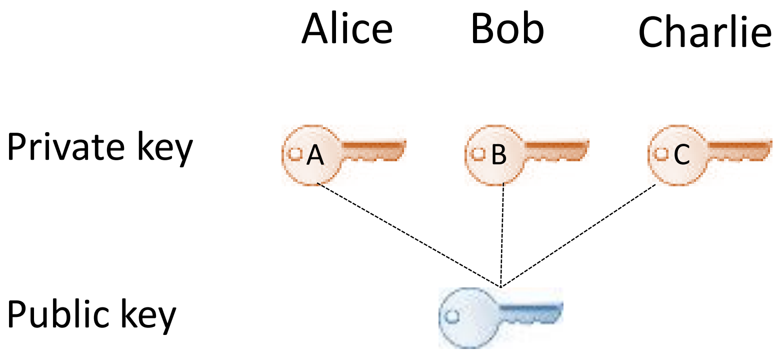
If the cryptographically secure attestation provided by Alice uses traditional digital signatures, Bob may learn what services Alice is interested to get access

Remote Attestation is one example of application that would benefit from privacy-preserving signatures

Privacy-Preserving Signatures

Group signatures allow a group member to anonymously sign messages on behalf of the whole group:

Key Setup:



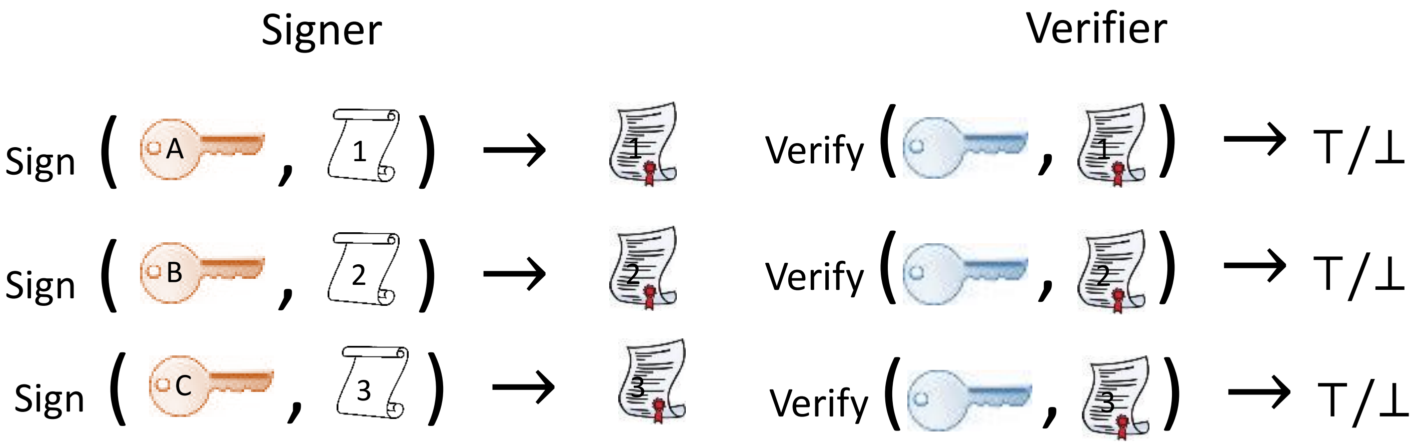
Alice Bob Charlie

Private key

Public key

1 public key per group

Sign/Verify:



Signer

Verifier

Sign (,) → Verify (,) → ⊤/⊥

Sign (,) → Verify (,) → ⊤/⊥

Sign (,) → Verify (,) → ⊤/⊥

Verifier does not know who (among all group members) generate the signatures

Group Signatures

Entities:

- **Group Manager (GM)**: Responsible for the group management, including key generation and signature opening
- **Group Member (User)**: Able to sign data such that his/her identity is concealed from any verifier other than the group manager
- **Verifier**: Able to verify the authenticity of group signatures, but not able to determine the identify of the signer

Algorithms:

- **Parameters**: k security parameter, N number of group members
- $\left(gmsk, gpk, gsk_0, \dots, g_{sk_{N-1}}\right)$ ←**G.KeyGen**($1^k, 1^N$)
- σ ←**G.Sign**(gsk_i, m)
- ⊤/⊥ ←**G.Verify**(σ, m, gpk)
- i ←**G.Open**($gmsk, \sigma, m$)

Main Security Properties:

- **Untraceability**: Given a signature, it is hard to determine the identity of the signer group member
- **Unlinkability**: Given any two signatures, it is hard to determine if they were issued by a same group member

Correctness Properties:

- G.Verify(G.Sign(gsk_i, m), m, gpk) = ⊤
- G.Open($gmsk$, G.Sign(gsk_i, m), m) = i

Group Signatures

- Most group signature schemes are based on number-theory cryptography
  - To mention a few: [ACJT00, CL02, BMW03, CL04, BSZ05, DP06, BW06, BW07, Gro07]
  - Intel® Enhanced Privacy ID (EPID) [BL09]: based on pairing-based cryptography
  - Quantum computers are expected to offer a dramatic speed-up to solve the underlying security problems of number-theory based cryptography [Shor94]

- Recent increasing interest in defining post-quantum group signatures:
  - Lattice-Based Group Signatures: [GKV10, LLLS13, LLNW14, NZZ15, LNW15]
  - Code-Based Group Signatures: [ELL+15]
  - All those schemes rely on additional security assumptions (e.g. shortest vectors in a lattice)

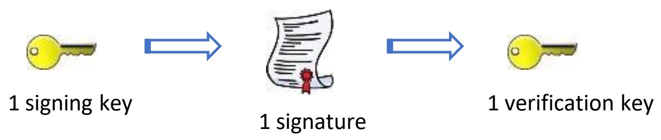
Can we build a group signature scheme out of standard (minimal) assumptions?

Hash-Based Signatures (HBS)

- Security:
  - The security of digital signature schemes with appended message relies on the security of hashing[1] + some other (believed-to-be-hard) problem
  - HBS base their security solely on well-known security notions from hash (e.g. pre-image)
- Efficiency: Keygen/sign/verify operations boil down to hash function calls
- One-Time and Multi-Time Hash-Based Signatures:

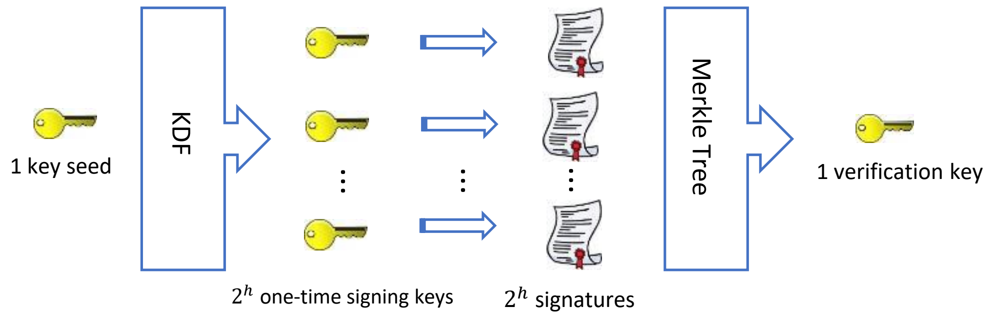
One-Time Signature (OTS):

Signing key should not be used to generate more than one signature



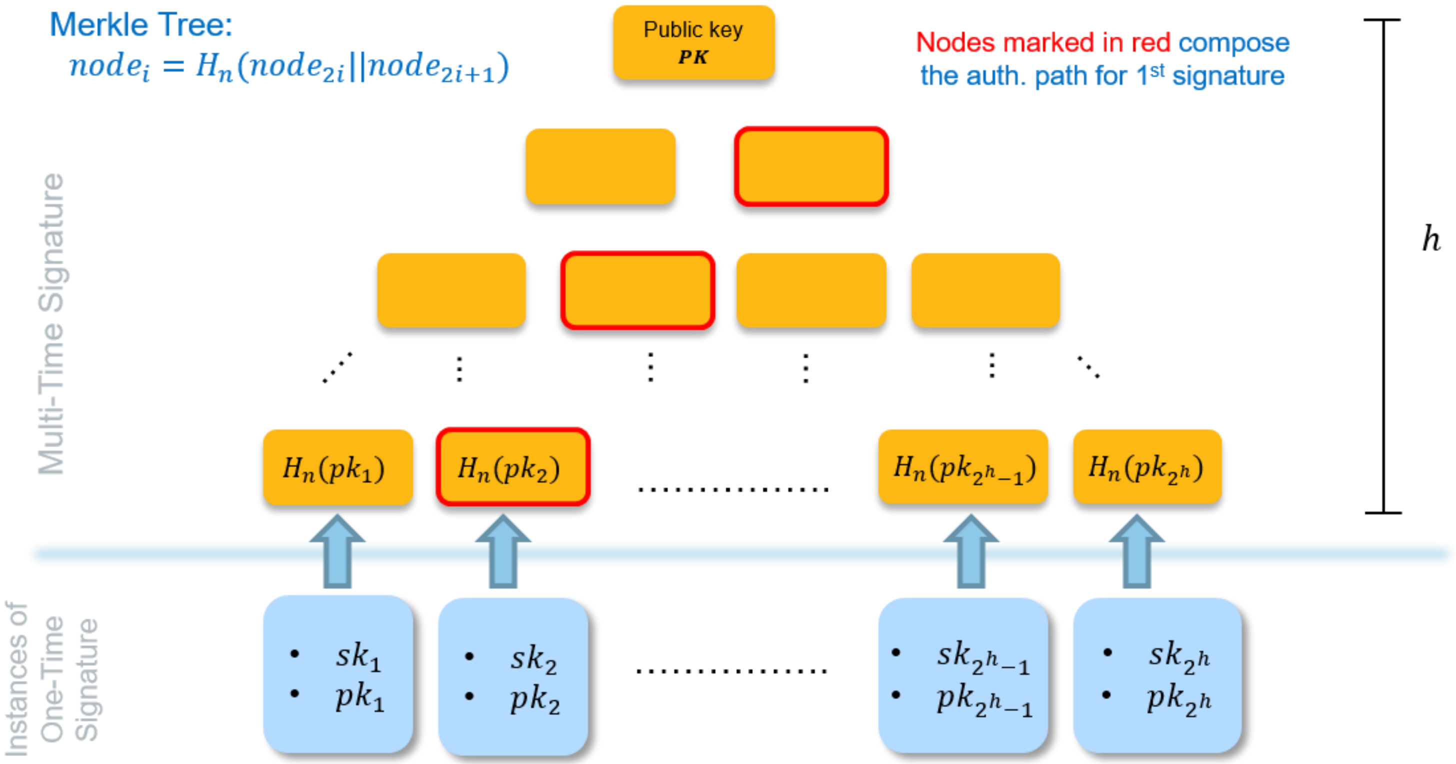
Multi-time Signature (MTS):

Uses **one-time** signature scheme to build **multi-time** signature scheme



[1]: Hash used to map the arbitrary-length input into a fixed-length input.

Merkle Signature Scheme (MSS) [Mer79]



Merkle Tree:
$node_i = H_n(node_{2i} || node_{2i+1})$

Public key
PK

Nodes marked in red compose the auth. path for 1st signature

Keys:

- Public-key: Root of the tree
- Private key: Seed **(to generate one-time keys)**

Authentication path:

- Nodes required to recompute the root
- Updating auth. cost: variable latency
- It does not depend on message (offline)

Signature is valid if:

- Computed root == public-key

Drawbacks:

- Stateful: private key changes over time
- (Virtual) limitation on num. of signatures

XMSS [BDH11] & LMS [LM95] Schemes:

- Reduces MSS signature size
- Milder security assumptions than MSS
- IETF drafts in advanced stage

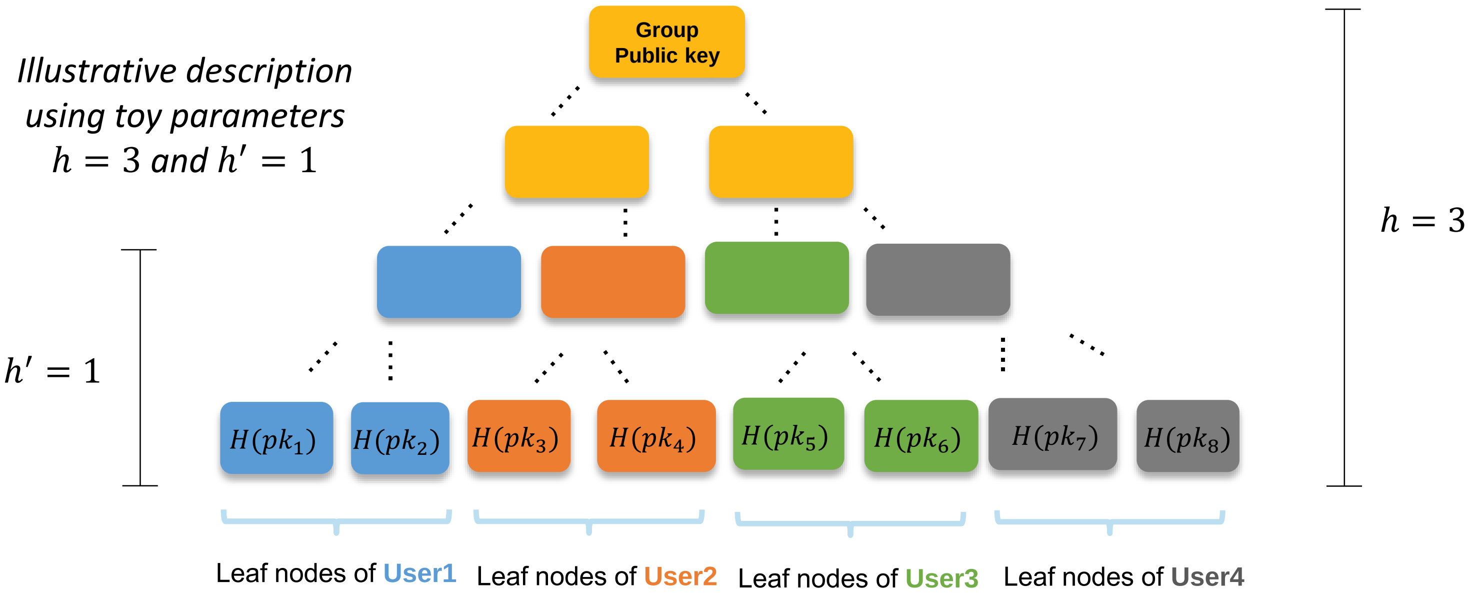
The same simplicity that leads to secure and efficient HBS schemes seemed to prevent the design of more elaborated schemes (e.g. hash-based group signatures)

G-Merkle – Initial Thoughts

Idea: Group shares a same Merkle tree. All group members will use the same group public key (root node)

Naïve approach: Each group member generates its own height-h' sub-tree and append it to the main tree:

Illustrative description using toy parameters $h = 3$ *and* $h' = 1$

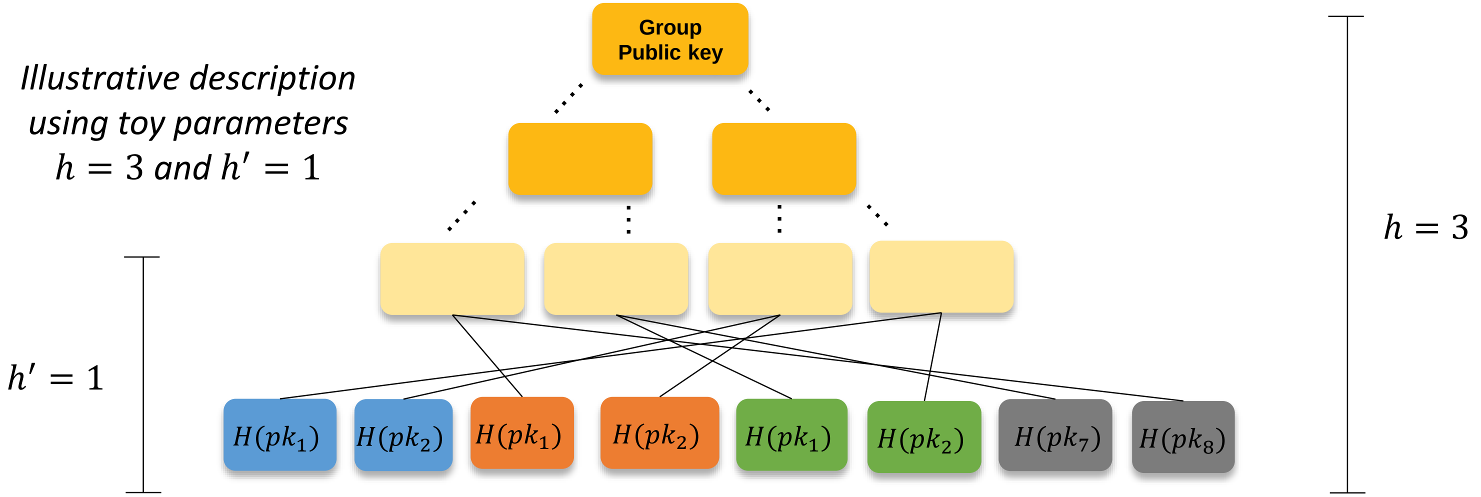


Problem: all signatures from a same user would share at least one node at level h' in the authentication path:

This property could be used to link signatures

G-Merkle – Simplified Description

- Each user owns $2^{h'}$ leaf nodes, as before. However, before building the tree, the leaf nodes are shuffled:



- The shuffling process is computed by the GM by means of a secret key and a Pseudo-Random Permutation (PRP)
- Unlinkability: the authentication path of signatures issued by a same signer will not share nodes, w.h.p.

G-Merkle

- Let:
  - E, D be the encryption and decryption algorithms of a block cipher
  - k: sec. parameter, N: number of group members, B: number of signatures per member
- G.KeyGen($1^k, 1^N$):
  1. GM generates a random symmetric key $gmsk$
  2. Each user $i \in [1, N]$ generates B one-time key pairs from a secret gsk_i
  3. GM shuffles the indices of the leaf nodes: $(j_1, \dots, j_{N \cdot B})$, where $j_s \leftarrow E_{gmsk}(s)$ for $s \in [1, N \cdot B]$
  4. GM builds a Merkle tree using the leaf nodes in shuffled order
  5. GM provides to each user i a list of tuples with their set of shuffled indices:

$$S_i = \{j_{(i-1)B+1}, \dots, j_{iB}\}$$

G-Merkle

- G.Sign(gsk_i, m):
  - As in the Merkle Signature Scheme
- G.Verify(σ, m, gpk):
  - As in the Merkle Signature Scheme
- G.Open($gmsk, \sigma, \mathrm{m}$):
  1. By examining the authentication path, GM recovers the shuffled index
  2. Decrypts the shuffled index to recover the original index and therefore determine the signer identity

G-Merkle

- G-Merkle vs. Lattice-Based Group Signatures [LLNW15]:
  - Group public key size: n vs. $\tilde{O}(logN \cdot n)$
  - Signature size: $O(|\sigma_{OTS}| + n(logN + logB))$ vs. $\tilde{O}(logN \cdot n)^*$
- Computing the authentication path of a signature: a few options
  - The tree is not secret, thus it can be made publicly available
  - It can be obtained by the user from Group Manager in key generation time
  - It can be obtained by the user from Group Manager in an online fashion
- G-Merkle inherits some properties from Merkle Signature Scheme:
  - Stateful signature scheme: state misuse can lead to security issues
  - (Exponentially large, but) limited number of signatures

*: or $\tilde{O}(logN \cdot n^2)$ for unstructure lattices.

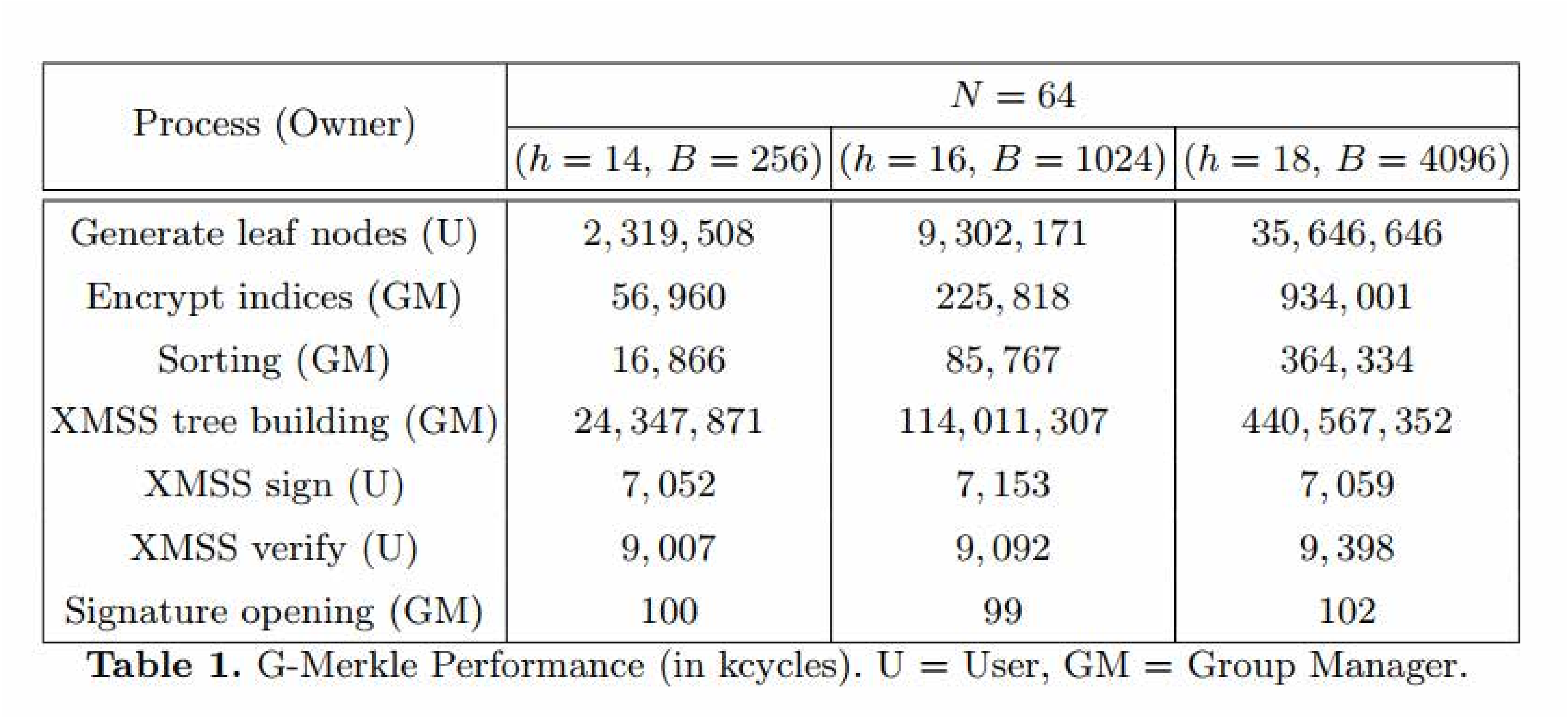
Security Assessment

- For security, we prove Traceability and Anonymity:
- **Traceability:** Allows the group manager possessing the master secret key to unveil the identity of a group signer.
  - G-Merkle is fully-traceable following [BMW03].
  - Proof is based on the existential unforgeability of the underlying Merkle Signature Scheme construction. As long as an adversary cannot sign on behalf of an honest signer or change the index, the real identity of a leaf can be recovered from its index.

Security Assessment

- For security, we prove Traceability and Anonymity:
- **(S)PRP-Anonymity:** An adversary, capable of corrupting signers, is not able to determine, which honest identity/signer (at least 2) actually signed a message in the security game.
    - ”weak” PRP-Anonymity: Adversary must guess the identity of a signer without access to the opening oracle (weak anonymity).
    - SPRP-Anonymity: Adversary is additionally given access to the opening oracle
- Anonymity follows from the pseudorandomness of the used (S)PRP.
- PRPs can be built from PRF following the generic way [NR96]

Performance Results



Process (Owner)	$N = 64$		
	$(h = 14,\ B = 256)$	$(h = 16,\ B = 1024)$	$(h = 18,\ B = 4096)$
Generate leaf nodes (U)	$2,319,508$	$9,302,171$	$35,646,646$
Encrypt indices (GM)	$56,960$	$225,818$	$934,001$
Sorting (GM)	$16,866$	$85,767$	$364,334$
XMSS tree building (GM)	$24,347,871$	$114,011,307$	$440,567,352$
XMSS sign (U)	$7,052$	$7,153$	$7,059$
XMSS verify (U)	$9,007$	$9,092$	$9,398$
Signature opening (GM)	100	99	102

Table 1. G-Merkle Performance (in kcycles). U = User, GM = Group Manager.

C Implementation of G-Merkle based on XMSS/WOTS+ (as defined in [HBGM18]), SHA2-256 and AES-256.

Conclusions

- G-Merkle answers the open question on the feasibility of building Group Signature schemes from standard (minimal) assumptions
- Security relies on hash function and PRPs (e.g. block ciphers)
- Sign and verify just as efficient as Merkle Signature Schemes
- Auth. path approach can be selected depending on the application

G-Merkle: A Hash-Based Group Signature Scheme From Standard Assumptions

Rachid El Bansarkhani

Technische Universität Darmstadt, Germany

elbansarkhani@cdc.informatik.tu-darmstadt.de

Rafael Misoczki

Intel Corporation, USA

Rafael.Misoczki@intel.com

References

- [ACJT00]: Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *Advances in Cryptology CRYPTO 2000*, volume 1880 of *LNCS*, pages 255-270, Santa Barbara, CA, USA, 2000. Springer, Heidelberg, Germany.

- [BL07]: Ernie Brickell and Jiangtao Li. Enhanced Privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. In Proceedings of the 6th ACM Workshop on Privacy in the Electronic Society. ACM Press, October 2007.

- [BL09]: Brickell, E. and Li, J., 2009. Enhanced Privacy ID from Bilinear Pairing. IACR Cryptology ePrint Archive, 2009, p.95.

- [BDH11]: Buchmann, J., Dahmen, E., and A. Huelsing, "XMSS – A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions", Lecture Notes in Computer Science volume 7071. Post-Quantum Cryptography, 2011.

- [ELL+15]: Martianus Frederic Ezerman, Hyung Tae Lee, San Ling, Khoa Nguyen, and
  Huaxiong Wang. A provably secure group signature scheme from codebased assumptions. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology { ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 260{285, Auckland, New Zealand, November 30 {December 3, 2015. Springer, Heidelberg, Germany.

- [LLNW14]: Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Latticebased group signature scheme with verifier-local revocation. In *PKC 2014*,
  pages 345{361. Springer, 2014.

- [LLS13]: Fabien Laguillaumie, Adeline Langlois, Beno^ıt Libert, and Damien Stehl´e. Lattice-based group signatures with logarithmic signature size. In ASIACRYPT 2013, pages 41{61. Springer, 2013.

- [LM95]: Leighton, F., Micali, S.: Large provably fast and secure digital signature schemes based on secure hash functions. 1995. US Patent 5,432,852.

- [LNW15]: San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *LNCS*, pages 427{449. Springer, Germany, March 30 { April 1, 2015.

- [NZZ15]: Phong Q. Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices. In PKC, pages 401{426. Springer, 2015.

- [Mer79]: Ralph Merkle. "Secrecy, authentication and public key systems / A certified digital signature". Ph.D. dissertation, Dept. of Electrical Engineering, Stanford University, 1979.

- [NR96]: Moni Naor and Omer Reingold. On the construction of pseudo-random permutations: Luby-rackoff revisited. IACR ePrint, 1996:11, 1996

- [Shor96]: Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In 35th Annual Symposium on Foundations of Computer Science, pages 124{134, Santa Fe, New Mexico, November 20{22, 1994. IEEE Computer Society Press.

G-Merkle

- Instantiating PRPs from One-Way Functions
  - PRPs can be built from PRFs [LR86]
  - PRFs can be built from One-Way Functions [GGM86]
  - Thus, OWFs suffice to construct secure PRPs
  - By using a OWF in G-Merkle, we ensure that the whole scheme is only based on OWF, which is the minimal requirement for the existence of public key crypto
- Instantiating PRPs from Block Ciphers
  - It is more efficient to instantiate PRP by means of a block cipher
  - In practice, we don’t find block ciphers with such small outputs and high security
  - We can use large block ciphers (e.g. AES128) by defining the new (shuffled) order considering only the *order* of the encrypted indices (uses a sorting algorithm)