

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

OKTA, INC.,
Petitioner,

v.

THALES DIS FRANCE SAS,
Patent Owner.

Case PGR2026-00045
Patent 12,395,359

**PETITION FOR *POST-GRANT* REVIEW OF UNITED STATES PATENT
NO. 12,395,359 PURSUANT TO 35 U.S.C. §§ 321-329, 37 C.F.R. § 42**

TABLE OF CONTENTS

I.	REQUIREMENTS FOR PGR.....	1
A.	Standing.....	1
B.	Challenge and Relief Requested	1
C.	Level of Ordinary Skill in the Art	4
II.	THE '359 PATENT.....	4
A.	Specification.....	4
B.	Relevant Prosecution.....	7
C.	Claim Construction	9
	1. Limitations [1.3], [1.4], [6.3], [6.4], [12.3], [12.4], [14.3], [14.4]	10
III.	UNPATENTABILITY GROUNDS.....	14
A.	[GROUND 1A] – Claims 6-9, 11, and 14 Are Obvious Over Wu-Müelich.....	14
	1. Wu-Müelich	14
	2. Analysis.....	28
B.	[GROUND 1B] – Claim 10 Is Obvious Over Wu-Müelich-Tuyls	62
	1. Wu-Müelich-Tuyls.....	62
	2. Analysis.....	65
C.	[GROUND 1C] – Claims 1-3, 5, 12, and 13 Are Obvious Over Wu-Müelich-Yang.....	67
	1. Wu-Müelich-Yang	67
	2. Analysis.....	71

D.	[GROUND 1D] – Claim 4 Is Obvious Over Wu-Müelich-Yang-Tuyls	74
1.	Wu-Müelich-Yang-Tuyls.....	74
2.	Analysis.....	75
E.	[GROUND 2A] – Claims 6-9, 11, and 14 Are Obvious Over Wu-Müelich-Chan.....	75
1.	Wu-Müelich-Chan	75
2.	Analysis.....	78
F.	[GROUND 2B] – Claim 10 Is Obvious Over Wu-Müelich-Chan-Tuyls	80
1.	Wu-Müelich-Chan-Tuyls.....	80
2.	Analysis.....	81
G.	[GROUND 2C] – Claims 1-3, 5, 12, and 13 Are Obvious Over Wu-Müelich-Chan-Sundararajan	81
1.	Wu-Müelich-Chan-Sundararajan.....	81
2.	Analysis.....	84
H.	[GROUND 2D] – Claim 4 Is Obvious Over Wu-Müelich-Chan-Sundararajan-Tuyls	87
1.	Wu-Müelich-Chan-Sundararajan-Tuyls	87
2.	Analysis.....	87
I.	[GROUND 3] – If Narrowly Interpreted to Avoid the Prior Art, Claims 1-14 Lack Written Description Support	88
1.	Background	88
2.	Legal Standard	89
3.	Lack of Written Description Support.....	90

J.	[GROUND 4] – If Narrowly Interpreted to Avoid the Prior Art, Claims 1-14 Are Indefinite	92
1.	Background	92
2.	Legal Standard	93
3.	Indefiniteness	94
IV.	DISCRETIONARY DENIAL IS NOT APPROPRIATE	96
V.	CONCLUSION.....	96
VI.	MANDATORY NOTICES	96
A.	Real Party-In-Interest Under 37 C.F.R. § 42.8(b)(1)	96
B.	Related Matters Under 37 C.F.R. § 42.8(b)(2)	96
C.	Lead And Back-Up Counsel Under 37 C.F.R. § 42.8(b)(3)	96
D.	Service Information.....	97

EXHIBITS

OKTA-1001	U.S. Patent No. 12,395,359 to Duval et al. (“the ’359 Patent”)
OKTA-1002	Prosecution History of the ’359 patent (“the Prosecution History”)
OKTA-1003	Declaration and Curriculum Vitae of Dr. Patrick Traynor
OKTA-1004	U.S. Patent Application No. 2015/0188717 (“Wu”)
OKTA-1005	S. Muelich, & M. Bossert, “A New Error Correction Scheme for Physical Unclonable Functions,” SCC 2017; 11th International ITG Conference on Systems, Communications and Coding. VDE, 2017 (“Muelich”)
OKTA-1006	U.S. Patent Application No. 2021/0051010 (“Yang”)
OKTA-1007	U.S. Patent Application No. 2012/0072737 (“Schrijen”)
OKTA-1008	A. Aghaie & A. Moradi, “TI-PUF: Toward Side-Channel Resistant Physical Unclonable Functions,” published April 2020, IEEE Transactions On Information Forensics And Security (“Aghaie”)
OKTA-1009	U.S. Patent Application No. 2019/0147967 (“Tuyls”)
OKTA-1010	G. Sundararajan, C. Winstead, & E. Boutillon, “Noisy Gradient Descent Bit-Flip Decoding for LDPC Codes,” published 2014, IEEE Transactions on Communications (“Sundararajan”)
OKTA-1011	A. Chan & F. Kschischang, “A Simple Taboo-Based Soft-Decision Decoding Algorithm for Expander Codes,” published 1998, IEEE Communications Letters (“Chan”)
OKTA-1012	USPTO First Inventor to File (FITF) Comprehensive Training presentation (available at

https://www.uspto.gov/sites/default/files/aia_implementation/fitf_comprehensive_training_prior_art_under_aia.pdf)

- OKTA-1013 Declaration of June Munford
- OKTA-1014 Memo, “Interim Processes for PTAB Workload Management,” March 26, 2025, *available at* <https://www.uspto.gov/sites/default/files/documents/InterimProcesses-PTABWorkloadMgmt-20250326.pdf>
- OKTA-1015 “A New Error Correction Scheme for Physical Unclonable Functions,” IEEE Xplore, Institute of Electrical and Electronics Engineers, <https://ieeexplore.ieee.org/abstract/document/7938055> (accessed 8 Apr. 2026)
- OKTA-1016 “A simple taboo-based soft-decision decoding algorithm for expander codes,” IEEE Xplore, Institute of Electrical and Electronics Engineers, <https://ieeexplore.ieee.org/document/703905> (accessed 8 Apr. 2026)
- OKTA-1017 “Noisy Gradient Descent Bit-Flip Decoding for LDPC Codes,” IEEE Xplore, Institute of Electrical and Electronics Engineers, <https://ieeexplore.ieee.org/document/6894606> (accessed 8 Apr. 2026)
- OKTA-1018 E. Stefanov, E. Shi, & D. Song, “Towards Practical Oblivious RAM,” *available at* <https://arxiv.org/abs/1106.3652> (accessed 28 Apr. 2026)
- OKTA-1019 M. Yu & S. Devadas, “Secure and Robust Error Correction for Physical Unclonable Functions,” IEEE Computer Society, published 2010
- OKTA-1020 U.S. Patent Application No. 2020/0059252 (“Fackenthal”)
- OKTA-1021 U.S. Patent Application No. 2015/0381206 (“Fainzilber”)

OKTA-1022	U.S. Patent Application No. 2017/0180140 (“Mai”)
OKTA-1023	U.S. Patent Application No. 2013/0194886 (“Schrijen-886”)
OKTA-1024	Liu et al., “Multi-Threshold Bit Flipping Algorithm for Decoding Structured LDPC Codes,” published Feb. 2015, IEEE Communications Letters (“Liu”)
OKTA-1025	U.S. Patent No. 10,749,694 (“Adams”)
OKTA-1026	U.S. Patent No. 10,574,467 (“Afghah”)
OKTA-1027	U.S. Patent No. 9,628,272 (“Rostami”)
OKTA-1028	Zhang et al., “Tabu-List Noisy Gradient Descent Bit Flipping Decoding of LDPC Codes,” published 2019, 11th International Conference on Wireless Communications and Signal Processing (WCSP) (“Zhang”)
OKTA-1029	Cui & Lin, “An Improved Gradient Descent Bit-Flipping Decoder for LDPC Codes,” published Aug. 2019, IEEE Transactions on Circuits and Systems (“Cui”)
OKTA-1030	RESERVED
OKTA-1031	T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usmai, & I. Takumi, “Gradient Descent Bit Flipping Algorithms for Decoding LDPC Codes,” available at: https://arxiv.org/pdf/0711.0261 (“Wadayama”)
OKTA-1032	Pavithra G. & Naveen Kumar M., “BER Performance Comparison of Bit Flipping Algorithms used for Decoding of LDPC Codes,” available at: https://www.ijert.org/research/ber-performance-comparison-of-bit-flipping-algorithms-used-for-decoding-of-ldpc-codes-IJERTV4IS051311.pdf (“Pavithra”)
OKTA-1033	U.S. Patent Application No. 2019/0305972 A1 (“Yao”)

LISTING OF CHALLENGED CLAIMS

Claim 1 ¹	
[1.pre]	A method for repairing response bits of a physical unclonable function of an electronic device, said physical unclonable function outputting said response bits and additional output bits, said method comprising the following steps:
[1.1]	generating a PUF response from said response bits (S1),
[1.2]	detecting an error in the PUF response (S2),
[1.3]	determining erroneous response bits of the PUF response (S3) comprising replacing one by one, in a random order, each bit of the PUF response by an arbitrary value and determining if the error in the PUF response is [] corrected,
[1.4]	determining a match (S4) between each determined erroneous response bit and a selected additional output bit such that replacing the erroneous response bits by the matching additional output bits corrects the error in the PUF response,
[1.5]	storing said determined match in a repair list (S5),
[1.6]	replacing (S6) in the PUF response said erroneous response bits with the matching additional output bits in the repair list.
Claim 2	
[2.1]	The method according to claim 1, wherein said electronic device is configured for access to a memory storing said repair list comprising at least one match between a response bit and an additional output bit,

¹ Claim 1 issued with a non-substantive printing error in [1.3] that is corrected in this Petition for clarity using bracketed text. *See* OKTA-1001, 7:15-16.

[2.2]	the step of generation of a PUF response (S1) comprises replacing in the PUF response said matched response bits with the matching additional output bits in the repair list.
Claim 3	
[3]	The method of claim 1, wherein said additional output bit are bits among most reliable output bits of said physical unclonable function.
Claim 4	
[4.1]	The method of claim 1, wherein said device is configured for access to a memory storing a reference hash value of said response bits,
[4.2]	detecting an error in the PUF response (S2) comprises: generating a hash value by applying a hash function on said bits of the PUF response, comparing said hash value with said stored reference hash value.
Claim 5	
[5]	The method of claim 3, wherein detecting an error in the PUF response (S2) comprises using an error correcting code.
Claim 6²	
[6.pre]	A method for repairing response bits of a physical unclonable function of an electronic device, said physical unclonable function outputting said response bits and additional output bits, said method comprising the following steps:
[6.1]	generating a PUF response from said response bits (S1),
[6.2]	detecting an error in the PUF response (S2),

² Claim 6 issued with non-substantive printing errors in [6.3] and [6.6] that are corrected in this Petition for clarity using bracketed text. *See* OKTA-1001, 7:55-56, 7:65-66.

[6.3]	determining erroneous response bits of the PUF response (S3) comprising replacing one by one, in a [predetermined] order, each bit of the PUF response by an arbitrary value and determining if the error in the PUF response is corrected,
[6.4]	determining a match (S4) between each determined erroneous bit and a selected additional output bit such that replacing the erroneous response bits by the matching additional output bits corrects the error in the PUF response,
[6.5]	storing said determined match in a repair list (S5),
[6.6]	replacing ([S6]) in the PUF response said erroneous response bits with the matching additional output bits in the repair list.
Claim 7	
[7.1]	The method according to claim 6, wherein said device being configured for having access to a memory storing a preventive toggling list of the least reliable response bits of said physical unclonable function,
[7.2]	determining erroneous response bits of the PUF response to be repaired (S3) comprises replacing first bits of the Preventive toggling list before replacing other bits of the PUF response.
Claim 8	
[8.1]	The method according to claim 6, wherein said electronic device is configured for access to a memory storing said repair list comprising at least one match between a response bit and an additional output bit,
[8.2]	the step of generation of a PUF response (S1) comprises replacing in the PUF response said matched response bits with the matching additional output bits in the repair list.
Claim 9	
[9]	The method of claim 6, wherein said additional output bit are bits among most reliable output bits of said physical unclonable function

Claim 10	
[10.1]	The method of claim 6, wherein said device is configured for access to a memory storing a reference hash value of said response bits,
[10.2]	detecting an error in the PUF response (S2) comprises: generating a hash value by applying a hash function on said bits of the PUF response, comparing said hash value with said stored reference hash value.
Claim 11	
[11]	The method of claim 9, wherein detecting an error in the PUF response (S2) comprises using an error correcting code.
Claim 12	
[12.pre]	An electronic device comprising active electronic primitives, PUF circuits generating a PUF response by running a physical unclonable function outputting response bits and additional output bits, a processor and at least one memory configured to perform the steps of:
[12.1]	generating a PUF response from said response bits (S1),
[12.2]	detecting an error in the PUF response (S2),
[12.3]	determining erroneous response bits of the PUF response (S3) comprising replacing one by one, in a random order, each bit of the PUF response by an arbitrary value and determining if the error in the PUF response is corrected,
[12.4]	determining a match (S4) between each determined erroneous response bit and a selected additional output bit such that replacing the erroneous response bits by the matching additional output bits corrects the error in the PUF response,
[12.5]	storing said determined match in a repair list (S5),
[12.6]	replacing (S6) in the PUF response said erroneous response bits with the matching additional output bits in the repair list.

Claim 13	
[13]	The electronic device of claim 12 wherein detecting an error in the PUF response (S2) comprises using an error correcting code.
Claim 14	
[14.pre]	An electronic device comprising active electronic primitives, PUF circuits generating a PUF response by running a physical unclonable function outputting response bits and additional output bits, a processor and at least one memory configured to perform the steps of:
[14.1]	generating a PUF response from said response bits (S1),
[14.2]	detecting an error in the PUF response (S2),
[14.3]	determining erroneous response bits of the PUF response (S3) comprising replacing one by one, in a predetermined order, each bit of the PUF response by an arbitrary value and determining if the error in the PUF response is corrected,
[14.4]	determining a match (S4) between each determined erroneous response bit and a selected additional output bit such that replacing the erroneous response bits by the matching additional output bits corrects the error in the PUF response,
[14.5]	storing said determined match in a repair list (S5),
[14.6]	replacing (S6) in the PUF response said erroneous response bits with the matching additional output bits in the repair list.

Okta, Inc. (“Petitioner”) petitions for Post Grant Review (“PGR”) of claims 1-14 (“the Challenged Claims”) of U.S. Patent No. 12,395,359 (“the ’359 patent”). Compelling evidence presented throughout this Petition shows that the Challenged Claims are obvious, lack written description support, and are indefinite.

I. REQUIREMENTS FOR PGR

A. Standing

Petitioner certifies that the ’359 patent is available for PGR. Petitioner is not barred or estopped from requesting this review challenging the Challenged Claims on the below-identified grounds. The ’359 patent issued on August 19, 2025, and the instant Petition was timely filed within nine months of issuance.

B. Challenge and Relief Requested

Petitioner requests PGR and cancellation of the Challenged Claims based on the grounds below. Regarding Grounds 1A-2D, Petitioner submits that the cited prior art references are themselves sufficient to establish obviousness pursuant to the analysis in Sections III.A-H. Nevertheless, the analysis in these sections further cites the Declaration of Dr. Patrick Traynor as complementary corroboration and explanation. *See* OKTA-1003.

Ground	Claims	Basis of Unpatentability
1A	6-9, 11, 14	Wu, Muelich (§103)
1B	10	Wu, Muelich, Tuyls (§103)
1C	1-3, 5, 12, 13	Wu, Muelich, Yang (§103)

Ground	Claims	Basis of Unpatentability
1D	4	Wu, Muelich, Yang, Tuyls (§103)
2A	6-9, 11, 14	Wu, Muelich, Chan (§103)
2B	10	Wu, Muelich, Chan, Tuyls (§103)
2C	1-3, 5, 12, 13	Wu, Muelich, Chan, Sundararajan (§103)
2D	4	Wu, Muelich, Chan, Sundararajan, Tuyls (§103)
3	1-14	Lack of Written Description Support (§112(a))
4	1-14	Indefiniteness (§112(b))

The '359 patent claims priority to PCT Application No. PCT/EP2021/075725, which claims priority to EP Application No. 20315413 filed 9/18/2020. OKTA-1001, Cover. While Petitioner does not concede that the '359 patent is entitled to the benefit of its priority claim, each reference asserted in this Petition qualifies as prior art under 35 U.S.C. § 102(a)(1) and/or § 102(a)(2) even if the “Critical Date” is assumed to be '359 patent's claimed priority date.

Reference	Filing Date	Publication Date
Wu	12/26/2013	7/2/2015
Muelich	N/A	2017
Yang	6/9/2020	2/18/2021
Tuyls	4/28/2017	5/16/2019
Chan	N/A	1998
Sundararajan	N/A	2014

Muelich (OKTA-1005) is a technical paper published in February 2017 in the 11th International ITG Conference on Systems, Communications and Coding. IEEE lists Muelich's “Date Added to IEEE *Xplore*” as “05 June 2017.” OKTA-1015.

IEEE was established in 1963 and is one of the world's largest technical professional associations. IEEE has been a prolific and exceedingly well-known publisher for decades. Public availability of Muelich is confirmed in an affidavit by June Munford. OKTA-1013. Ms. Munford confirmed its availability on IEEE *Xplore* on 6/5/2017, shortly after its publication during the SCC 2017, 11th International ITG Conference on Systems, Communications and Coding, held between 2/6/2017 and 2/9/2017 in Hamburg, Germany. *Id.*; *see also* OKTA-1005, 1.

Chan (OKTA-1011) is a technical paper published in the July 1998 edition of the IEEE Communications Letters (Volume 2, Issue 7). IEEE lists Chan's "Date of Publication" as "31 July 1998." OKTA-1016. Public availability of Chan is confirmed in an affidavit by June Munford. OKTA-1013. Ms. Munford confirmed that cataloging dates extracted from library MARC records corroborate public availability by July 1998. *Id.*

Sundararajan (OKTA-1010) is a technical paper published in the October 2014 edition of the IEEE Transactions on Communications (Volume 61, Issue 10). IEEE lists Sundararajan's "Date of Publication" as "09 September 2014." OKTA-1017. Public availability of Sundararajan is confirmed in an affidavit by June Munford. OKTA-1013. Ms. Munford confirmed that cataloging dates extracted from library MARC records corroborate public availability by September 2014. *Id.*

Each of Wu (OKTA-1004), Muelich (OKTA-1005), Tuyls (OKTA-1009), Chan (OKTA-1011), and Sundararajan (OKTA-1010) is prior art under at least 35 U.S.C. § 102(a)(1) as a printed publication before the Critical Date. *See* MPEP § 2128(II)(A); OKTA-1012. Yang (OKTA-1006) is prior art under at least 35 U.S.C. § 102(a)(2) as a published U.S. patent application filed before the Critical Date.

C. Level of Ordinary Skill in the Art

A person of ordinary skill in the art (“POSITA”) relating to the subject matter of the ’359 patent would have had (1) a bachelor’s degree in computer engineering, electrical engineering, or a related field, and (2) two years of industry experience in designing cryptographic or security systems in electronic devices. OKTA-1003, ¶¶25-27. Additional education could substitute for professional experience, and *vice versa*. *Id.*

II. THE ’359 PATENT

A. Specification

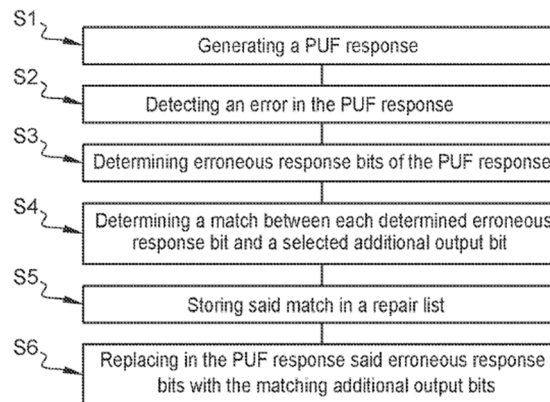
The ’359 patent is directed to “an electronic device and a method thereof for repairing response bits of a physical unclonable function of an electronic device.” OKTA-1001, Abstract, 3:33-4:29. The ’359 patent explains that a “Physically Unclonable Function (PUF) is a function embodied in a physical device whose output depends on the microstructures of the device components generating it.” *Id.*, 1:14-58. PUF characteristics (also referred to as “primitives”) “depend on random

process variations during manufacture that are uncontrolled and cannot be monitored.” *Id.* This makes a PUF “practically impossible to predict or duplicate.” *Id.*; OKTA-1003, ¶32.

Due to the “natural variations in the behavior of electronic components” the output of a PUF (the PUF “response”) “may not always be the same for a given challenge.” OKTA-1001, 1:14-58. In order to achieve a steady challenge-response relationship, the ’359 patent describes a need for “a method enabling to [*sic*] repair the response bits of a PUF when primitives forming the PUF response suffer from damage, making the output bits unsteady or permanently modifying [*sic*] their value.” *Id.* The ’359 patent’s proposed solution involves replacing unreliable response bits with unused output bits of a comparable value (“additional output bits”). *Id.*, 3:33-4:29. “The main idea...is that when a PUF response bit is not stable anymore or was permanently damaged, it may be permanently replaced in the PUF response by another output bit providing the same value than the one the damaged response bit used to provide.” *Id.*, 4:11-16; OKTA-1003, ¶33.

The process of replacing unreliable PUF response bits with more reliable additional output bits is accomplished by “replac[ing] one by one each bit of the PUF response by an arbitrary value and determin[ing] if the error in the PUF response is corrected.” OKTA-1001, 5:1-8. The ’359 patent explains that “[r]eplacing each bit may be performed in a random order in order to prevent an

attacker from deducing the index of erroneous bits from the time spent for identifying them.” *Id.*, 5:9-12. In other embodiments, this process “may be performed in a predetermined order,” such that “advantage can be taken of knowledge about how reliable the PUF response bits are to test the least reliable bits first.” *Id.*, 5:12-25. In either embodiment, the process “determine[s] a match between each determined erroneous response bit and a selected additional output bit such that replacing the erroneous response bits by the matching additional output bits corrects the error in the PUF response.” *Id.*, 5:37-44. Once a match between an unreliable response bit and an additional output bit is made, “the determined match is stored” in a “repair list.” *Id.*, 5:52-64; *see* OKTA-1003, ¶¶34-35 (citing OKTA-1018, 1-11). An example process according to the ’359 patent is shown below in FIG. 2.



OKTA-1001, FIG. 2

B. Relevant Prosecution

The '359 patent issued on August 19, 2025 from U.S. Patent Application No. 18/026,925 (“the '925 application”), which was filed on March 17, 2023 following an earlier international filing on September 17, 2021. OKTA-1002, 3-4. The prosecution of the '925 application included a single Office Action (“the Action”) that presented a §103 rejection based on US 2019/0065734 (“Yao”) and US 2017/0310489 (“Van Der Sluis”). OKTA-1002, 81-94. Additionally, the Examiner indicated in the Action that certain dependent claims were allowable (then claims 7-9, each of which depended on then claims 3, 5, and 6). *Id.* The Examiner indicated that the reason for allowance was that the prior art of record allegedly lacked disclosure or suggestion of these claims. OKTA-1002, 90-92; OKTA-1003, ¶36.

Allowable Subject Matter

6. Claims 7-9 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

OKTA-1002, 90.

Reasons for Allowance

7. The following is an examiner's statement of reasons for placing claims 7-9 under allowable subject matters:

The limitation of dependent claims 7-9 are allowed and the corresponding dependent claims including respective and any intervening claims are not disclosed by the any of the prior art of record. For example, the limitations in claim 9 recites "...wherein said device being configured for having access to a memory storing a Preventive toggling list of the least reliable response bits of said physical unclonable function, determining erroneous response bits of the PUF response to be repaired (S3) comprises replacing first bits of the Preventive toggling list before replacing other bits of the PUF response.." which are not are not taught or fairly suggested by the prior art of record.

OKTA-1002, 91

In response, the Applicant amended the claims to incorporate the allowable subject matter (claim 1 is reproduced below as an example, incorporating then claims 6 and 7). OKTA-1002, 55-63, 71-79. The Examiner then allowed the '925 application, noting Yao and Van Der Sluis as the closest art of record. OKTA-1002, 23-34; OKTA-1003, ¶37.

Listing of amended Claims:

1. (CURRENTLY AMENDED) A method for repairing response bits of a physical unclonable function of an electronic device, said physical unclonable function outputting said response bits and additional output bits,
said method comprising the following steps:
generating a PUF response from said response bits (S1),
detecting an error in the PUF response (S2),
determining erroneous response bits of the PUF response (S3) comprising replacing one by one, in a random order, each bit of the PUF response by an arbitrary value and determining if the error in the PUF response is corrected,
determining a match (S4) between each determined erroneous response bit and a selected additional output bit such that replacing the erroneous response bits by the matching additional output bits corrects the error in the PUF response,
storing said determined match in a repair list (S5),
replacing (S6) in the PUF response said erroneous response bits with the matching additional output bits in the repair list.

OKTA-1002, 56

C. Claim Construction

All claim terms should be construed according to the *Phillips* standard, such that all words of the Challenged Claims are given their ordinary and customary meaning in view of the specification. *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005); 37 C.F.R. §42.200; OKTA-1003, ¶38. The discussion that follows addresses the ordinary and customary meaning of a subset of limitations addressed in Section III. Petitioner submits that no further claim terms need be expressly construed at this time to resolve issues of controversy in this Petition. *Wellman, Inc. v. Eastman Chem. Co.*, 642 F.3d 1355, 1361 (Fed. Cir. 2011). Petitioner reserves

the right to respond to any constructions offered by Patent Owner or adopted by the Board.

1. Limitations [1.3], [1.4], [6.3], [6.4], [12.3], [12.4], [14.3], [14.4]

As discussed in §III.A.2, *infra*, independent claims 1, 6, 12, and 14 recite:

determining erroneous response bits of the PUF response
(S3) comprising replacing one by one...each bit of the PUF
response by an arbitrary value and determining if the error in the
PUF response is corrected,

determining a match (S4) between each determined
erroneous bit and a selected additional output bit such that
replacing the erroneous response bits by the matching additional
output bits corrects the error in the PUF response

Dr. Traynor explains that “a POSITA would have understood (using the ordinary and customary meaning) that this process, based on how the process is described in the ’359 patent specification, requires replacing individual bit values until such replacement corrects the PUF response error and, thus, reveals an erroneous bit.” OKTA-1003, ¶¶39-40 (citing OKTA-1001, 6:29-63, FIG. 4). As discussed below, when an erroneous bit is discovered, it is matched with an additional output bit. *Id.*

The above-recited limitations do not require determining multiple erroneous response bits in a single error-detection pass. In other words, the claimed process of “replacing one by one...each bit of the PUF response...and determining if the error

in the PUF response is corrected” need not reveal more than one erroneous bit. OKTA-1003, ¶41.

While the claim language references “erroneous response bits” (plural), it is well established that the plural form of a claim element “does not [necessarily] imply that multiple [elements] are required by the claim.” *Versa Corp. v. Ag-Bag Intern. Ltd.*, 392 F.3d 1325, 1330 (Fed. Cir. 2004) (“[I]n context, the plural can describe a universe ranging from one to some higher number, rather than requiring more than one item”) (citing *Dayco Products, Inc. v. Total Containment, Inc.*, 258 F.3d 1317, 1328 (Fed. Cir. 2001)). The surrounding claim language makes clear that bit values are replaced one at a time, i.e., “replacing one by one.” And it follows that this iterative process would reveal a single erroneous bit—the bit whose value, when replaced, corrects the erroneous PUF response. The specification confirms this interpretation by juxtaposing the claimed “replacing one by one” approach against an unclaimed alternative where “multiple bits may be replaced” to address a scenario where “multiple bits are erroneous in the PUF response.” OKTA-1001, 5:1-8; OKTA-1003, ¶42.

The above-recited limitations also do not require replacing the value of each and every single bit of the PUF response in a single error-detection pass. Rather than exhaustive bit replacement, the claimed “replacing one by one” scheme is conducted in an iterative fashion until the erroneous bit is discovered by

“determining if the error in the PUF response is corrected.” Once the erroneous bit is discovered, the claims do not require any further bit value replacements. OKTA-1003, ¶43.

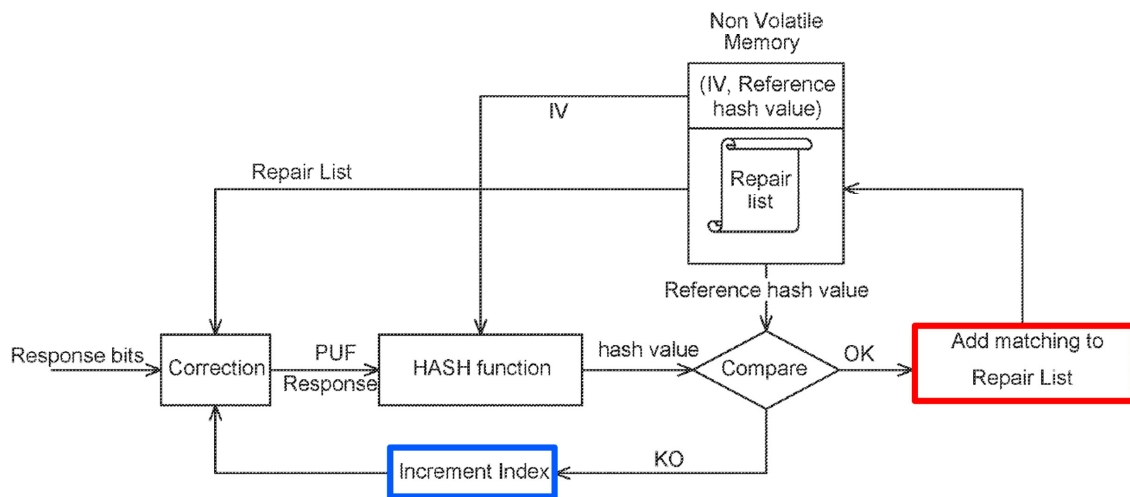
This understanding is supported by the specification’s narrative that the purpose of the alleged invention is to “minimize the computations required to find the erroneous bits of the PUF response” and to “reduc[e] the number of response bits to be replaced.” OKTA-1001, 2:51-52, 3:1-3.³ Conducting needless bit replacement operations defies the ’359 patent’s express purpose. Accordingly, in describing the preferred embodiment of Figure 4, the ’359 patent makes clear that the error-detection process concludes when the erroneous bit is identified and matched with a corresponding additional output bit. *See* OKTA-1001, 6:29-36, FIG. 4. As Dr. Traynor explains, “[i]f it were otherwise, the specification would have expressly stated that the process of replacing bit values continues, even after identifying the erroneous bit, until every single bit of the PUF response has been replaced.” OKTA-1003, ¶44.

Likewise, Figure 4 (below) would have illustrated a sequential flow or operational link from the “**Add matching to Repair List**” step to the “**Increment Index**” step. *Id.* Such an illustration would have indicated that the process continues

³ *See* §II.A for a detailed overview of the Specification.

iteratively examining PUF response bits even after the erroneous bit is discovered and matched with a corrective additional output bit, but it does not exist. Figure 4 instead shows the “**Add matching to Repair List**” step and the “**Increment Index**” step as distinct and separate branches from the “Compare” step. *Id.* In other words, when the comparison indicates that the PUF response has been corrected, the repair list is updated but there is no need to continue incrementing the index because no further bit replacements occur. *Id.*; OKTA-1003, ¶¶45-46.

Fig. 4



OKTA-1001, FIG. 4 (annotated)

In fact, the '359 patent's specification fails to disclose even a single embodiment in which each and every bit of the PUF response is replaced. *See generally*, OKTA-1001; OKTA-1003, ¶47. It is, therefore, not plausible under *Phillips* to interpret the claims in a manner that requires this undisclosed limitation.

To the extent Patent Owner argues otherwise, the Challenged Claims are invalid for lack of written description support (*infra*, § III.I) and indefiniteness (*infra*, § III.J).

III. UNPATENTABILITY GROUNDS

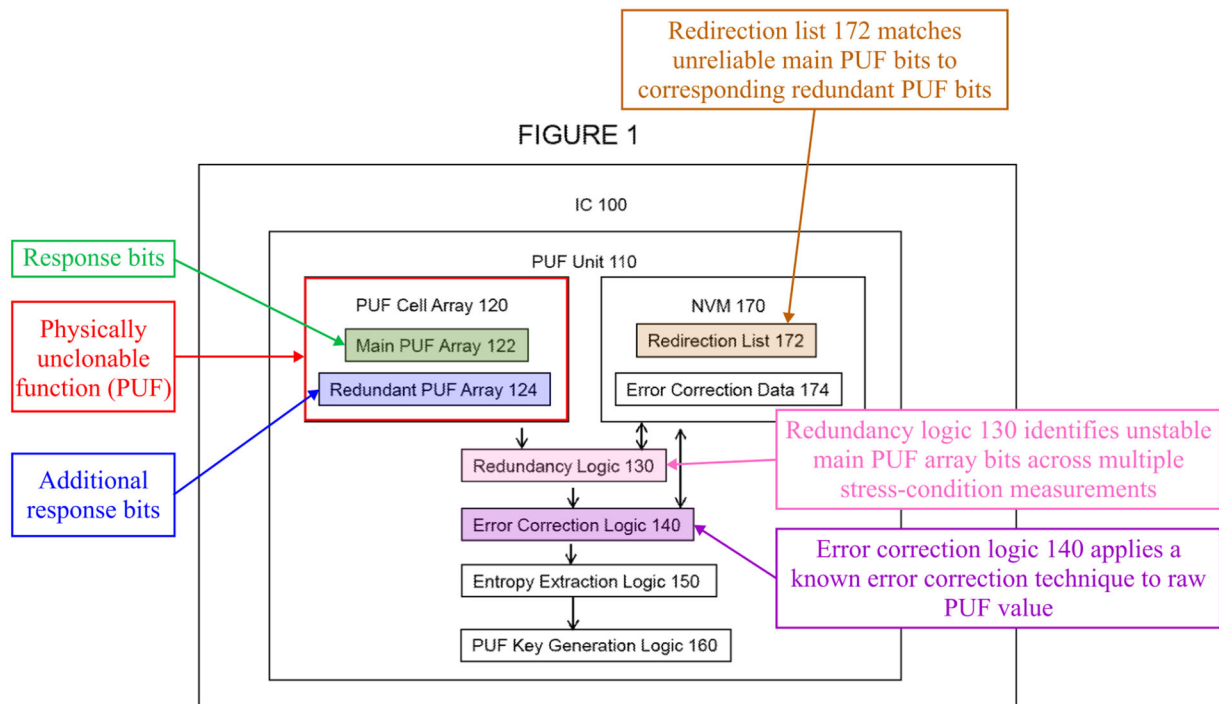
A. [GROUND 1A] – Claims 6-9, 11, and 14 Are Obvious Over Wu-Müelich

1. Wu-Müelich

(a) Wu (OKTA-1004)

Wu discloses techniques “providing for physically unclonable function redundant bits” in the context of electronic devices and integrated circuits. OKTA-1004, [0010], [0015]-[0018]. Like the ’359 patent, Wu teaches that a PUF typically includes redundant bits that can be used for “replacing bad bits in order to reduce the noise in and/or error rate of PUF key generation.” *Id.*, [0020]-[0025], FIG. 1; OKTA-1003, ¶49.

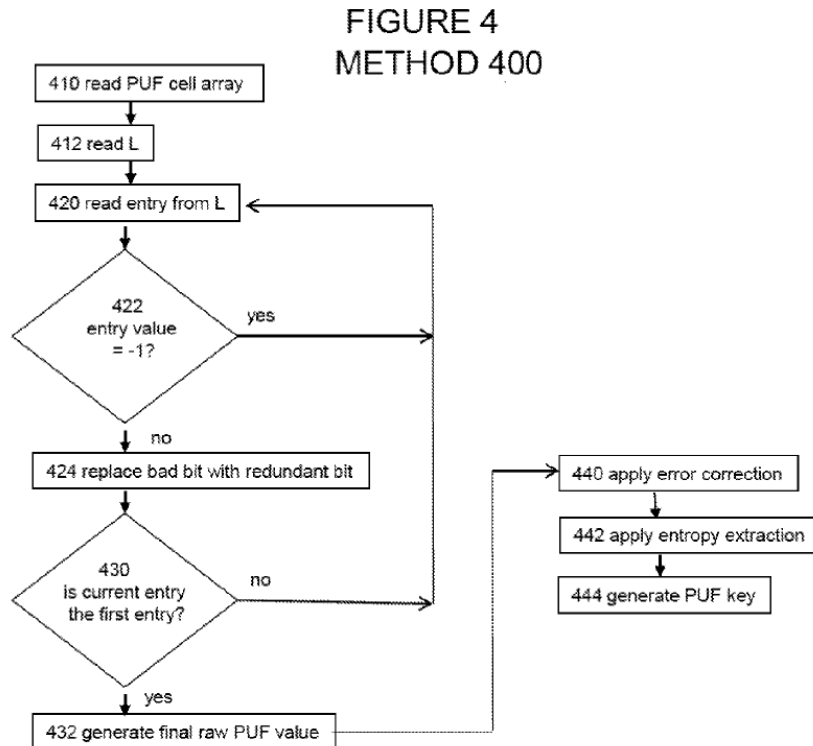
For example, referencing Figure 1 (below), Wu describes a redirection list 172 on an integrated circuit 100, where each entry of the list corresponds to “one of the PUF cells in redundant PUF array 124” and indicates “a particular bit in main PUF array 122...that is to be replaced by the redundant PUF cell identified by that entry.” OKTA-1004, [0028]-[0030], FIG. 1; OKTA-1003, ¶50.



OKTA-1004, FIG. 1 (annotated)

Wu further describes how “redundancy logic 130” is configured to perform the testing, identification, and replacement of “bad” bits. OKTA-1004, [0023]. Wu’s testing and identification of bad bits can occur at different stages of the device’s lifecycle. In one example, Wu teaches that the testing and bad-bit identification is performed “by the manufacturer or vendor,” such that the device is provisioned with a populated redirection list before shipment. *See id.* (before sale or release). In another example, Wu teaches that the testing and identification process can also occur “post-manufacturing (e.g., during customer use)” so as “to counter device aging effects.” *Id.*, [0073]; OKTA-1003, ¶51.

Bad bit replacement by the redundancy logic 130 occurs “during the use of [integrated circuit] 100 (evaluation time), for example, by a customer,” as shown in the exemplary workflow shown in the method 400 of Figure 4 (below). OKTA-1004, [0058]; OKTA-1003, ¶52.



OKTA-1004, FIG. 4

As shown above, Wu’s exemplary workflow involves reading/evaluating the PUF cell array (step 410) and performing a list-driven replacement of identified “noisy” (bad) bits using redundant bits (step 424). OKTA-1004, FIG. 4, [0063]-[0064]. The “final raw PUF value” is then “processed by error correction logic 140” (step 440), which Wu says “may include any logic circuitry, or other hardware and/or firmware

to provide for correcting errors in the generation of PUF keys.” *Id.*, [0064], [0025]; OKTA-1003, ¶53.

(b) Muelich (OKTA-1005)

Like the ’359 patent and Wu, Muelich acknowledges that error correction is an “indispensable component” to PUFs because “environmental factors” inevitably cause variance in a PUF’s response over time. OKTA-1005, 1. Muelich describes improving error identification and correction in PUF-based key generators using iterative, bit-level correction during decoding. *Id.* Muelich’s “Bitflip Decoding” technique iteratively selects a bit position, flips that bit, and checks whether the decoding condition is satisfied. *Id.*, 4-5 (“We use a simple, iterative bitflip procedure[.]”); OKTA-1003, ¶54.

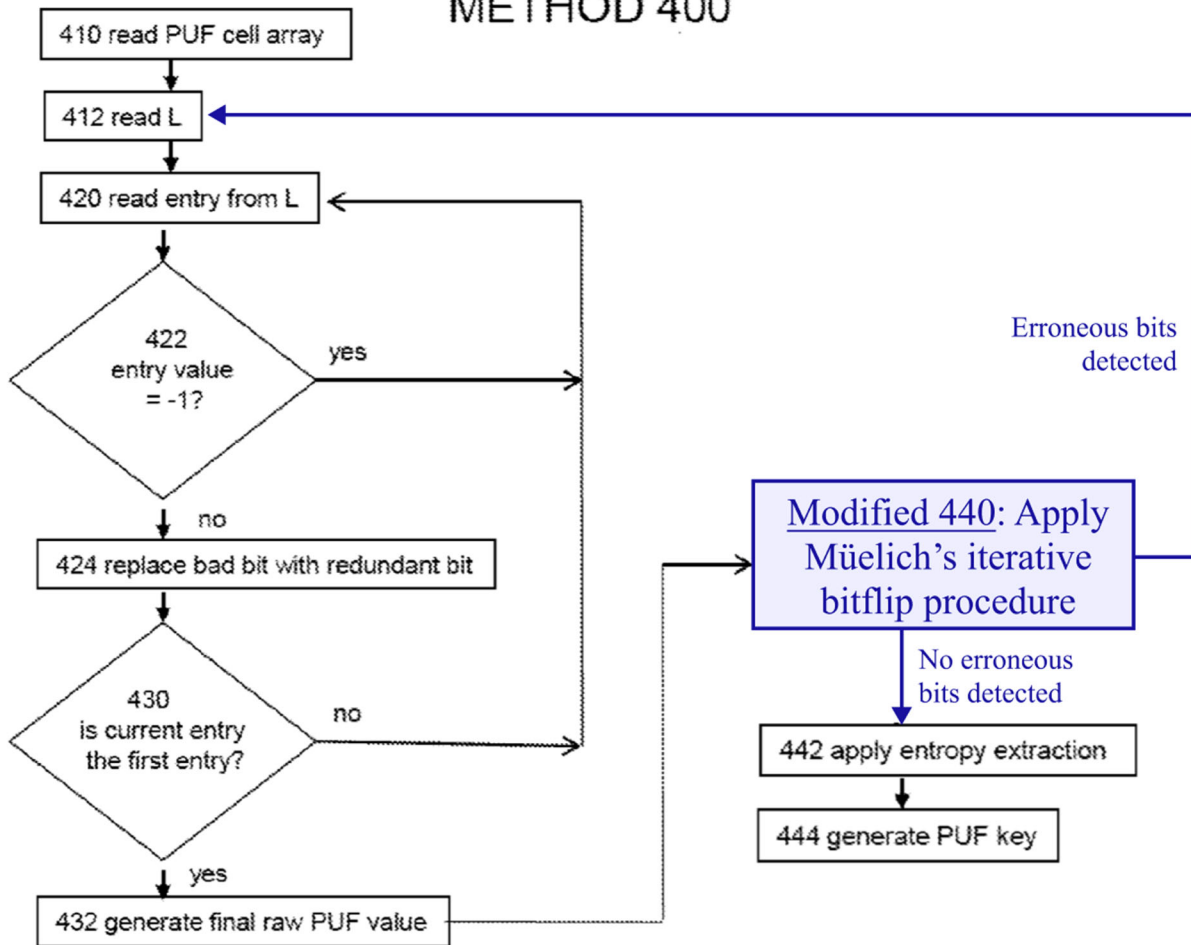
(c) Combination of Wu and Muelich

As discussed below, the Wu-Muelich Combination represents the predictable and expected result of incorporating into Wu’s PUF key generation workflow: **[a]** Muelich’s teachings regarding its iterative bitflip procedure to identify erroneous bits in a PUF response (e.g., Wu’s “final raw PUF value”) for recordation in Wu’s redirection list L; and **[b]** Wu’s suggestion regarding post-manufacturing updating of its redirection list L, which governs list-driven replacement at Wu’s step 424 in subsequent iterations of Wu’s method 400. By incorporating these features, the Wu-Muelich Combination would have predictably improved Wu by enabling Wu’s PUF

key generation to: perform per-bit evaluations through Muelich's iterative bitflip procedure to identify additional erroneous bits, update Wu's redirection list L based on those per-bit evaluations, and perform subsequent corrections (i.e., bit replacements) via list-driven substitution. OKTA-1003, ¶59. The following discussion outlines these aspects of the Wu-Muelich Combination in more detail, and the subsequent sub-sections provide an explicit motivation analysis for each.

As to the combination, a POSITA would have implemented Wu's error correction logic 140 to execute Muelich's iterative bitflip procedure on the "final raw PUF value" produced upstream in Wu's method 400. OKTA-1004, FIG. 4, [0063]-[0064]; OKTA-1005, 4; OKTA-1003, ¶60. In operation, the Wu-Muelich Combination would have proceeded from Wu's evaluation-time readout of the PUF cell array (step 410) to Wu's list-driven replacement stage (step 424) to yield the "final raw PUF value," which would then be processed by a modified step 440. OKTA-1004, [0063]-[0064]. This is shown in the modified version of Wu's Figure 4 below:

FIGURE 4
METHOD 400



Wu, FIG. 4 (modified and annotated)

As shown above, at the modified step 440, Muelich's iterative bitflip procedure evaluates the final raw PUF value on a per-bit basis to identify additional erroneous bits not previously addressed by the prior list-driven replacement at step 424. OKTA-1005, 4-5. If Muelich's procedure detects no additional erroneous bits, method 400 proceeds to Wu's downstream entropy extraction at step 442 and key generation at step 444. OKTA-1004, [0064], FIG. 4. If Muelich's procedure

identifies one or more additional erroneous bits, however, the Wu-Müelich Combination records the identified bit positions in Wu's redirection list L (consistent with Wu's suggestion regarding implementation flexibility of method 400). OKTA-1004, [0072]. This enables bad-bit identification and replacement to occur post-manufacturing during customer use. *Id.*, [0073]. Upon such identification, the Wu-Müelich Combination would have reverted to step 412, where the updated redirection list L is read. OKTA-1004, [0059]-[0061], [0072]-[0073], FIG. 4; OKTA-1003, ¶61. Steps 420, 422, and 424 then traverse the updated L. When the traversal reaches the newly-added entry, step 424 uses that entry to perform list-driven replacement of the bit at the erroneous position identified at modified step 440 with a redundant bit from redundant PUF array 124. OKTA-1004, [0061], FIG. 4.

As would have been clear to a POSITA, the configuration described above relies on existing Wu infrastructure and could be implemented without introducing new subsystems. OKTA-1003, ¶62. For example, Wu already teaches replacing identified bad bits by selecting redundant bits via redirection list L and performing list-driven replacement at step 424, and Wu further teaches storing list L in non-volatile memory 172 for later use. OKTA-1004, [0028]-[0030], [0063]-[0064]. Accordingly, upon Müelich's procedure identifying one or more erroneous bit positions at the modified step 440, the Wu-Müelich Combination would have:

- (i) selected an available redundant-bit source from redundant PUF array 124, and
- (ii) updated the mapping in list L stored in non-volatile memory 172, making the mapping available for the subsequent reversion to step 412 and the list-driven replacement at step 424 discussed above.

OKTA-1004, [0025], [0063]-[0064], [0073]; OKTA-1003, ¶62.

From a POSITA's perspective, integrating Muelich's iterative bitflip procedure into Wu's open-ended "error correction logic" would have involved nothing more than applying known solutions to known problems in the field of cryptographic circuit design to achieve predictable outcomes. OKTA-1003, ¶63. The fundamental aspects of Wu's disclosure—including the redundant PUF array 124, redirection list L, and redundancy logic 130—would have remained intact, supplemented by the well-known teachings of Muelich to arrive at the claimed invention with a reasonable expectation of success. *Id.* Likewise, the POSITA would have expected success when applying Wu's own supplemental teaching to continue identifying bad bits post-manufacturing during customer use. *Id.* The POSITA would have recognized that Muelich's iterative bitflip procedure would readily identify erroneous bit positions in Wu's final raw PUF value, that those positions would be recorded in Wu's redirection list L, and that the bits at those

positions would then be replaced by redundant bits on subsequent iterations of Wu's method 400. *Id.*

- (i) *Applying Muelich's teachings on iterative bitflip procedure in the context of Wu's error correction logic*

A POSITA would have been motivated to incorporate Muelich's iterative bitflip procedure into Wu's "error correction logic" stage (step 440). The predictable result of this aspect of the Wu-Muelich Combination includes a PUF generation system that evaluates Wu's final raw PUF value on a per-bit basis using Muelich's iterative bitflip procedure to identify erroneous bit positions, thereby reducing the error rate of the PUF key output by Wu's key generation logic 160. OKTA-1003, ¶¶64-65.

The following independent reasons, taken separately or collectively, support this aspect of the combination:

First, a POSITA would have been motivated to combine these references because Wu expressly invites the use of known error correction approaches, and Muelich provides an exemplary technique suited for this context. OKTA-1003, ¶66. Wu teaches an "error correction logic" stage (step 440) but leaves the task of selecting an error correction mechanism open to the POSITA, stating that "[a]ny known error correction technique may be used." OKTA-1004, [0025]. Muelich teaches exactly such a technique: an iterative bitflip procedure applied to PUF-based

key generators that evaluates candidate bit positions on a per-bit basis and terminates when the syndrome weight criterion indicates the error is corrected. OKTA-1005, 1, 5. Given Wu’s explicit suggestion to employ “known” techniques, and Muelich’s disclosure of a bitflip procedure applied to the PUF key-reproduction context, it would have been axiomatic for a POSITA to employ Muelich’s technique to implement Wu’s “error correction logic” to achieve the predictable result of correcting errors in the final raw PUF value. OKTA-1003, ¶67.

Second, a POSITA would have been led to the Wu-Muelich Combination because it merely involves the application of a known technique (Muelich’s iterative bitflip procedure) to a known system (Wu’s PUF-based key generation workflow) to achieve predictable results (a system that reliably reproduces PUF responses with reduced error rate). OKTA-1003, ¶68. Each element in the combination merely performs the same function as it does separately. Wu’s system generates and manages the PUF key lifecycle, and Muelich’s iterative bitflip procedure identifies erroneous bit positions in the PUF response, which are then recorded in Wu’s redirection list L and used to govern list-driven replacement at Wu’s step 424 in subsequent iterations of method 400. *Id.* “It’s enough for [the Petition] to show that there was a known problem of [PUF response instability], that [Muelich’s iterative bitflip procedure] helped address that issue, and that combining the teachings of [Wu and Muelich] wasn’t beyond the skill of an ordinary artisan. Nothing more is

required to show a motivation to combine under *KSR*.” *Intel Corp. v. PACT XPP Schweiz AG*, 61 F.4th 1373, 1380-81 (Fed. Cir. 2023).

Third, a POSITA would have been motivated to employ Muelich’s iterative bitflip procedure in the context of Wu because it utilizes a simple, efficient approach that is highly suitable for onboard processing. OKTA-1003, ¶69. Muelich’s procedure evaluates candidate bit flips on a per-bit basis and selects the single flip that most reduces the syndrome weight at each iteration, a straightforward greedy-descent mechanism implementable in minimal hardware. *Id.* Muelich describes this as a “simple, iterative bitflip procedure.” OKTA-1005, 5. A POSITA would have recognized that using such a streamlined approach would have been particularly appropriate in the context of Wu’s system, where the error correction is performed by the “internal operation” of an integrated circuit. OKTA-1004, [0058]. In implementations where circuit footprint and processing capability are naturally constrained, a POSITA would have been motivated to select Muelich’s “simple” routine to limit the computational complexity and hardware overhead required to implement Wu’s “error correction logic.” OKTA-1003, ¶70.

Fourth, a POSITA would have recognized that applying Muelich’s teaching would predictably improve Wu’s system efficiency over the device’s operational lifetime. OKTA-1003, ¶71. As discussed above, each iteration of method 400 in which Muelich’s iterative bitflip procedure identifies an erroneous bit position

results in that position being recorded in Wu's redirection list L. OKTA-1004, [0029], [0059]-[0061], [0072]-[0073]. A POSITA would have recognized that, as L accumulates entries over successive iterations, a greater portion of the device's erroneous bit positions are handled by Wu's deterministic list-driven replacement at step 424 and a correspondingly smaller portion require iterative diagnosis at modified step 440. OKTA-1004, [0061], [0063], FIG. 4. The Wu-Müelich Combination thus predictably reduces the computational workload of the error-correction stage over the device's operational lifetime while maintaining reliable PUF key regeneration. OKTA-1003, ¶71.

- (ii) *Applying Müelich's iterative bitflip procedure to implement Wu's teaching of post-manufacturing identification and replacement of bad bits*

Having implemented Wu's error correction logic at modified step 440 using Müelich's iterative bitflip procedure, a POSITA would have been further motivated to utilize the per-bit erroneous position information produced by that procedure to implement Wu's express teaching regarding post-manufacturing identification and replacement of bad bits to counter device aging effects. *See* OKTA-1004, [0073]. The predictable result is a system that records erroneous bit positions identified at modified step 440 in Wu's redirection list L stored in non-volatile memory 172, such

that subsequent iterations of method 400 substitute redundant bit values for those positions via Wu's list-driven replacement at step 424. OKTA-1003, ¶¶72-74.

The following independent reasons, taken separately or collectively, support this aspect of the combination:

First, a POSITA would have been prompted to pursue this combination based on the express teachings apparent from Wu itself. OKTA-1003, ¶75. Wu explicitly identifies a desire to enable “post-manufacturing (e.g., during customer use) identification and replacement of bad bits,” specifically to “counter device aging effects.” OKTA-1004, [0073]. However, Wu does not detail the specific technique for diagnosing these new bad bits in the field. OKTA-1003, ¶75. Muelich provides such a technique to satisfy this design need. *Id.* Specifically, Muelich's iterative bitflip procedure identifies the bit position responsible for an erroneous PUF response by sweeping each position $i = 1, \dots, n$, computing the syndrome weight that would result from flipping the bit at position i , and selecting the position whose flipping most reduces the syndrome weight (i.e., most corrects the error). OKTA-1005, 5; OKTA-1003, ¶75. The match between Wu's stated goal and Muelich's technical solution would have been apparent to a POSITA, who would have sought to combine them to achieve the predictable result of a PUF system that self-updates Wu's redirection list L as the device ages.

Second, a POSITA would have recognized that updating Wu’s redirection list L based on Muelich’s erroneous bit identifications would predictably improve the long-term reliability of the device. OKTA-1003, ¶76. Wu discloses a “redirection list” specifically designed to map “bad bits” to record the address of each bit to be replaced together with the redundant bit that replaces it, and stores the list in non-volatile memory 172 for persistent access across iterations of method 400. OKTA-1004, [0028]-[0030], [0059]. A POSITA would have recognized that when Muelich’s iterative bitflip procedure identifies a bit position whose flipping drives the syndrome weight to zero, that bit position is effectively “bad.” OKTA-1003, ¶76. Rather than requiring Muelich’s iterative procedure to re-diagnose the same erroneous bit position on every subsequent iteration of method 400, it would have been obvious to persist this diagnosis by updating Wu’s redirection list to map the newly identified bad bit to a redundant bit from redundant PUF array 124. *Id.* This yields a device that incrementally becomes more reliable as L accumulates entries for erroneous bit positions identified over the device’s operational lifetime. *Id.*

Finally, a POSITA would have also had a reasonable expectation of success in implementing the Wu-Muelich Combination. OKTA-1003, ¶77. Wu expressly identifies step 440 as the evaluation-time stage for applying error correction logic to the final raw PUF value and expressly permits the use of “any known error correction technique,” including with error-correction data stored in non-volatile memory for

later use. OKTA-1004, [0025], [0058], [0063], [0064]. Muelich provides an iterative bitflip procedure applicable to PUF-based key reproduction. OKTA-1005, 1, 3, 4. Incorporating Muelich's procedure into Wu's step-440 error-correction logic, and additionally persisting identified erroneous bit positions into Wu's redirection list L stored in non-volatile memory 172, reflects the predictable use of known techniques in the field of PUF error correction to improve an expressly-identified error-correction stage. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 416-417 (2007); MPEP § 2143; OKTA-1003, ¶77. A POSITA would have implemented this predictable integration of Wu and Muelich through routine engineering choices without invention. OKTA-1003, ¶77.

2. Analysis

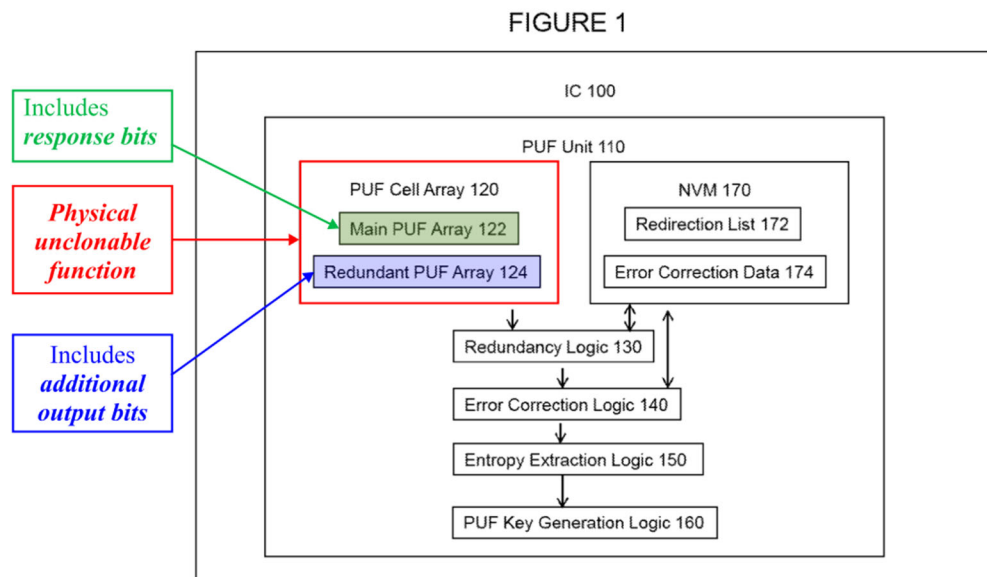
(a) Claim 6

[6.pre] A method for repairing response bits of a physical unclonable function of an electronic device, said physical unclonable function outputting said response bits and additional output bits, said method comprising the following steps:

As discussed in §III.A.1(c), the Wu-Muelich Combination features Wu's evaluation-time workflow (e.g., method 400) augmented by Muelich's iterative bitflip procedure to facilitate error correction (***method of repairing response bits of a physical unclonable function of an electronic device***). OKTA-1003, ¶78.

Wu describes “redundancy logic 130 [that] provides for replacing bad bits⁴ in order to reduce the noise and/or error rate of PUF key generation.” OKTA-1004, [0024] (emphasis added). Wu also describes an error correction logic 140 that provides for correcting errors in PUF key generation. *Id.*, [0025]; OKTA-1003, ¶79.

As shown in the annotated version of Wu’s Figure 1 below, the Wu-Müelich Combination features an integrated circuit 100 (*electronic device*) having a PUF cell array 120 (*physical unclonable function*) that includes a main PUF array 122 *outputting response bits* and a redundant PUF array 124 *outputting additional output bits*. See OKTA-1004, [0010], [0013], [0015]-[0017], [0020]-[0022], [0027], [0046], [0059], [0063]; OKTA-1003, ¶80.



OKTA-1004, FIG. 1 (annotated)

⁴ All emphasis has been added unless otherwise indicated.

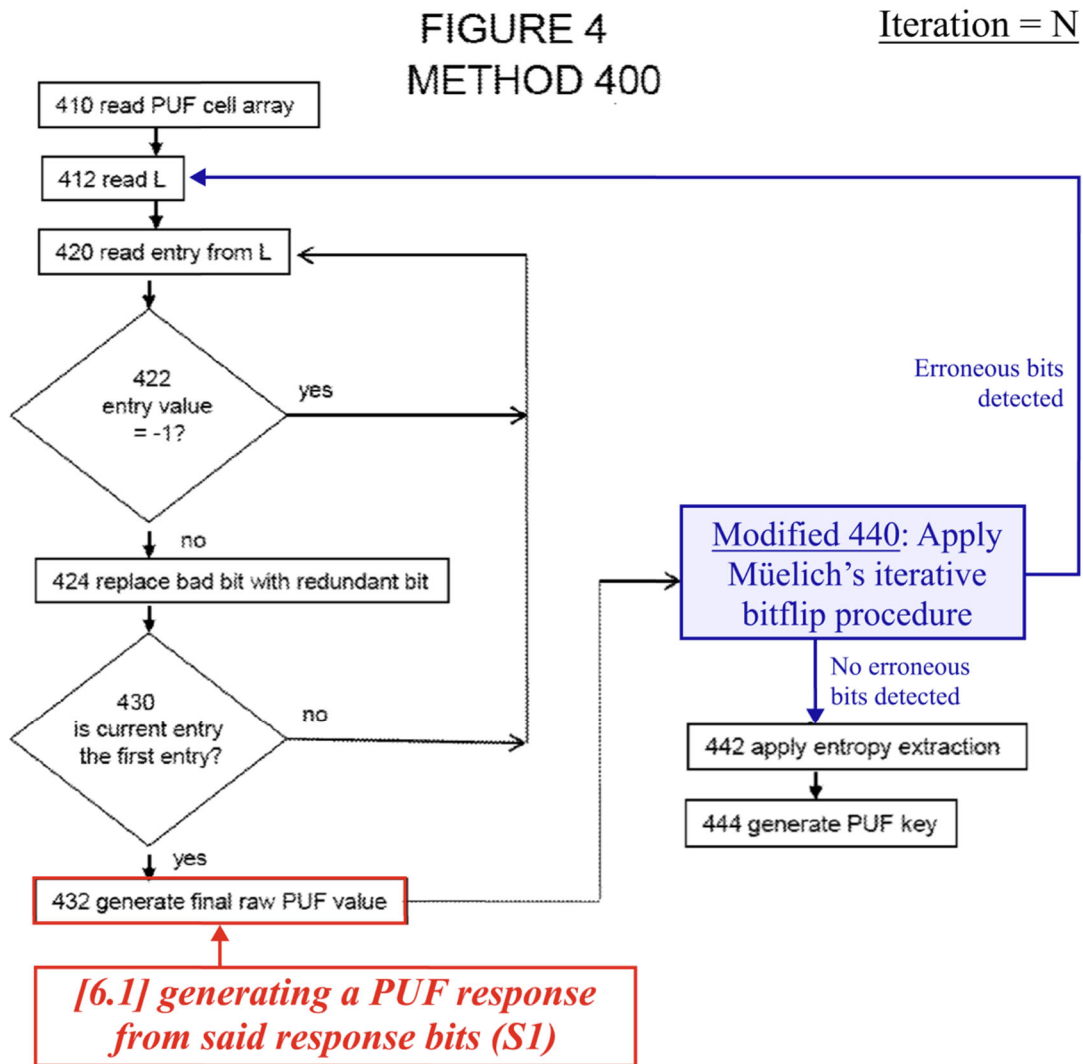
More specifically, the main PUF array 122 includes “bits” (e.g., “bits b_0 through b_{15} ”) (*id.*, [0046]) that are used for “cryptographic and/or other keys to be generated or derived...” (*id.*, [0021]); OKTA-1003, ¶81. The redundant PUF array 124 “may be proportional to the size of the main PUF array 122,” for example, “the ratio of redundant PUF cells to main PUF cells may correspond to the raw error rate, plus extra redundant PUF cells to provide a desired margin based on the expected variability in the raw error rate.” OKTA-1004, [0022]; *see also id.*, [0046] (describing a four-bit redundant PUF array comprising bits b_{16} through b_{19}). In Wu’s nomenclature, “a PUF cell from redundant PUF array 12[4] may be referred to as a redundant bit.” *Id.* As discussed in §III.A.1(c), Wu contemplates **repairing response bits** from the main PUF array 122 by selectively replacing them with redundant bits from the redundant PUF array 124. *See generally* OKTA-1004, [0058]-[0073].

[6.1] generating a PUF response from said response bits (S1),

The Wu-Müelich Combination is configured to produce, from the bits of main PUF array 122 (**response bits**) read at Wu’s steps 410 and 424, a “final raw PUF value” (**PUF response**). OKTA-1003, ¶82; *see supra* §III.A.1(c). Wu’s method 400 reads PUF cell array 120 at step 410 to “obtain an initial raw PUF value.” OKTA-1004, [0058]-[0059], FIG. 4. The method then applies entries in redirection list L at step 424 to substitute redundant-bit values for the bits in the initial raw PUF value

at the positions identified in those entries. *Id.*, [0061], [0063]. At step 432, method 400 generates the “final raw PUF value” by taking “a first portion of the updated PUF value” for a main PUF array of 128 bits, where “the first 128 bits of the updated PUF value are used as the final raw PUF value.” *Id.*, [0063]. The final raw PUF value is a bitstring generated from main PUF array 122’s response bits, with list-driven substitutions (if any) applied in accordance with the then-current redirection list L. OKTA-1004, [0058]-[0063], FIG. 4; OKTA-1003, ¶82.

The final raw PUF value produced at step 432 is also *generated from said response bits*. The final raw PUF value includes response bits as modified by any then-current redirection list L entries. OKTA-1004, [0061], [0063], FIG. 4; OKTA-1003, ¶83. For any iteration of method 400 in which L is empty, the final raw PUF value reduces to the unmodified response-bit readout from main PUF array 122. In contrast, for iterations in which L contains one or more entries written during a prior iteration, the final raw PUF value incorporates the corresponding redundant-bit substitutions at step 424 in accordance with Wu’s baseline list-driven replacement mechanism. OKTA-1004, [0061]; OKTA-1003, ¶83.



OKTA-1004, FIG. 4 (annotated)

[6.2] detecting an error in the PUF response (S2),

The Wu-Muelich Combination would have been configured to detect errors in the final raw PUF value at modified step 440 using Muelich's iterative bitflip procedure (see OKTA-1005, 1-3), where a nonzero syndrome weight ($w \neq 0$) indicates the presence of errors in the final raw PUF value and thereby *detect[s] an error in the PUF response*. OKTA-1003, ¶84.

As discussed in §III.A.1(c), *supra*, Wu's step 440 processes the final raw PUF value using error correction logic 140. OKTA-1004, [0063]. In the Wu-Müelich Combination, this stage is implemented using Müelich's iterative bitflip procedure, which begins by computing the syndrome weight, $w = WT(r)$, of the final raw PUF value (treated as a received word r). OKTA-1005, 5; OKTA-1003, ¶85. A POSITA would have understood that this initial syndrome evaluation *detect[s] an error in the PUF response* because a nonzero syndrome indicates the presence of an error. OKTA-1005, 4-5; OKTA-1003, ¶85. For example, if the syndrome weight is zero ($w = 0$), no error is present. However, if the syndrome weight is non-zero, the final raw PUF value contains an error, which the procedure proceeds to identify and correct (*see* [6.3], *infra*). OKTA-1005, 5 (Algorithm 1, lines 1-4); OKTA-1003, ¶85.

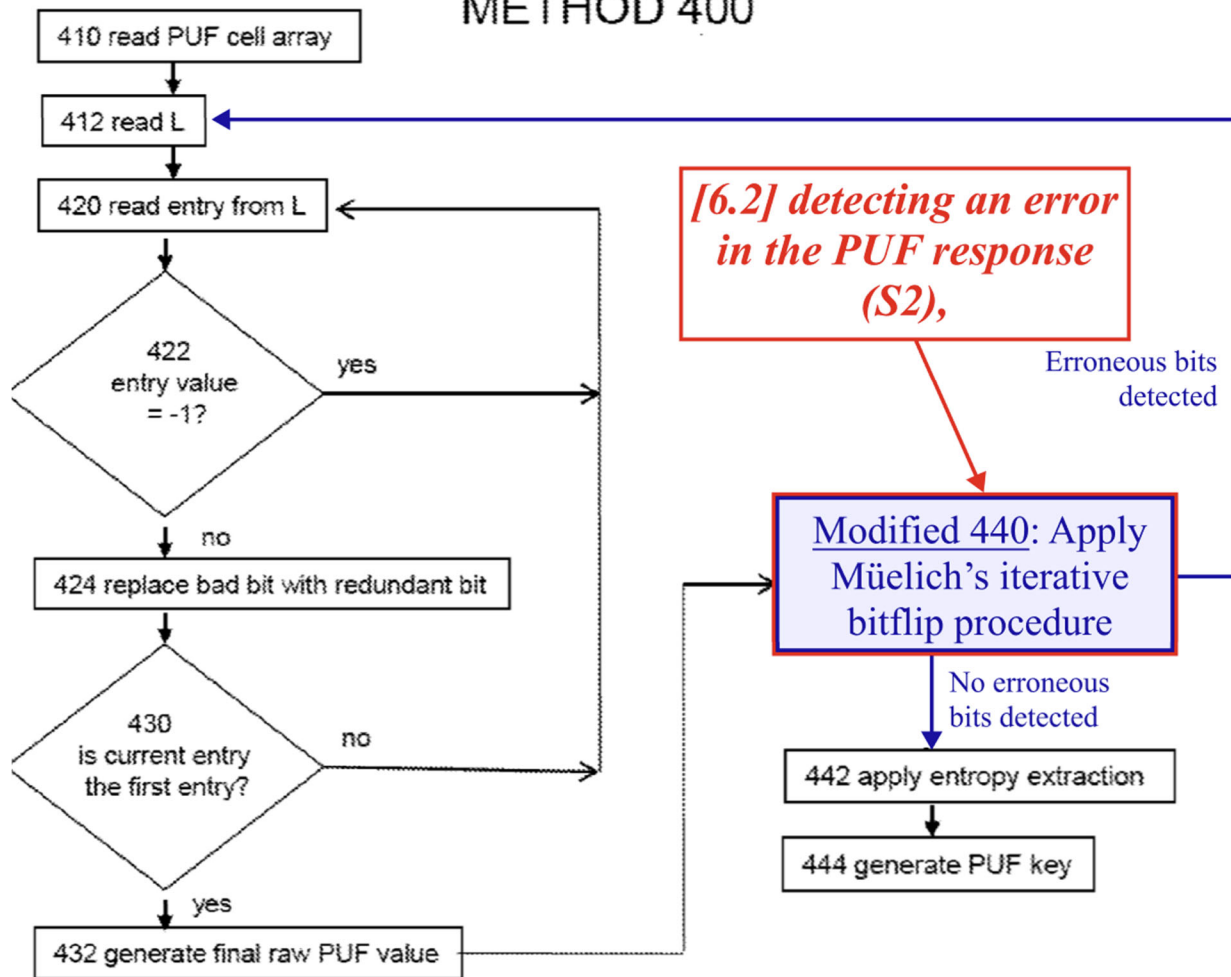
Wu further confirms through a separate manufacturing-time workflow (method 300) that bit-level disagreement in a raw PUF value is the mark of error. Specifically, Wu detects per-cell instability by generating an initial raw PUF value R , changing an operating condition (e.g., temperature/voltage), and generating a second raw PUF value R' under that changed condition. OKTA-1004, [0035]-[0041], FIG. 3. Wu detects cell-level disagreement by performing a bitwise XOR between R and R' , the output of which "may serve to indicate which PUF cells have changed their value between the initial measurement and the new measurement, and may therefore be considered bad." *Id.*, [0039]. Although Wu's method 300

performs this detection at manufacturing time using two measurements under differing conditions, this disclosure demonstrates that Wu at least contemplates bit-level disagreement in a raw PUF value as an indicator of an error. OKTA-1003, ¶86. The Wu-Müelich Combination applies similar underlying principles at evaluation time, detecting such bit-level disagreement in the final raw PUF value via Müelich's syndrome-based mechanism at modified step 440. *Id.*

Wu's redundancy logic 130 and error correction logic 140 together provide multiple layers of error detection and correction in Wu's PUF key generation architecture. OKTA-1004, [0023]-[0025]. In the Wu-Müelich Combination, Müelich's iterative bitflip procedure at modified step 440 operates as the evaluation-time error-detection layer, applied to the final raw PUF value produced in method 400. OKTA-1003, ¶87.

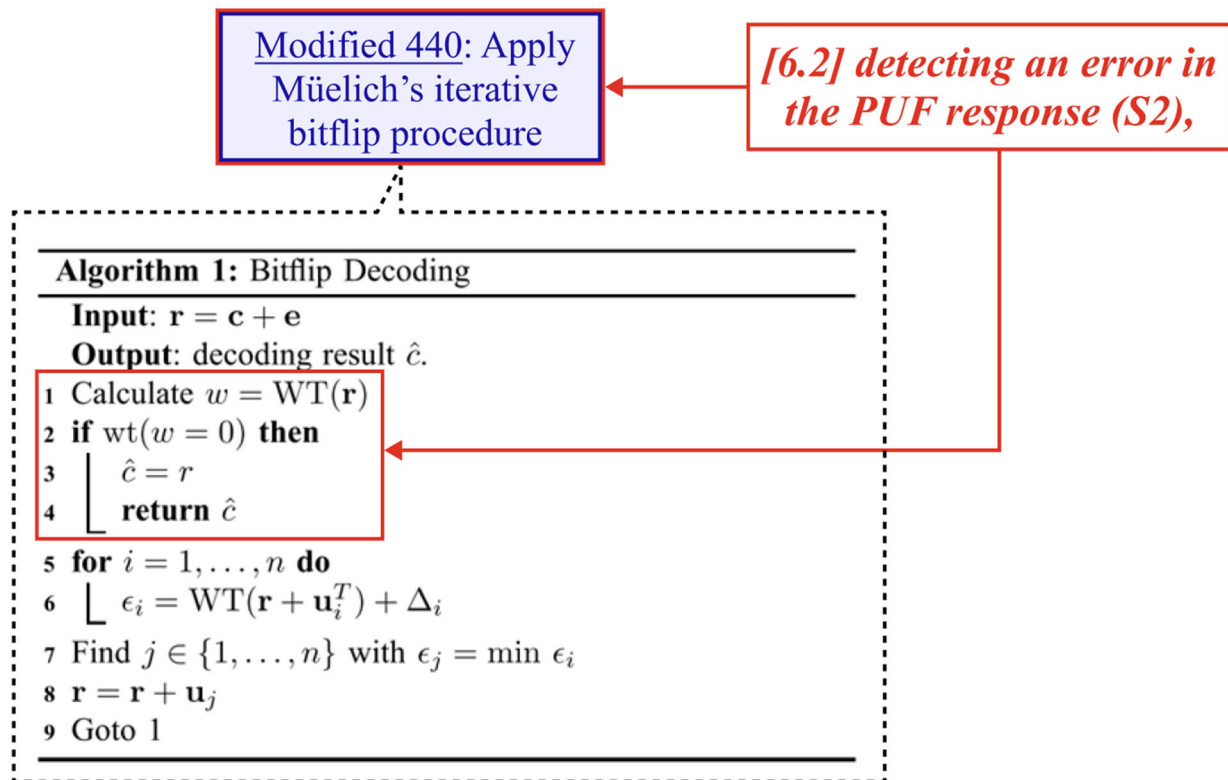
FIGURE 4
METHOD 400

Iteration = N



OKTA-1004, FIG. 4 (annotated)

Iteration = N



OKTA-1005, 5 (annotated)

[6.3] determining erroneous response bits of the PUF response (S3) comprising replacing one by one, in a [predetermined] order, each bit of the PUF response by an arbitrary value and determining if the error in the PUF response is corrected,

As explained in §II.C, *supra*, based on plain meaning, [6.3] does not mandate replacement of every single bit to satisfy the step of **determining erroneous response bits of the PUF response (S3)**. OKTA-1003, ¶88; see OKTA-1001, 6:29-36, FIG. 4. Consistent with this plain meaning understanding, the Wu-Muelich Combination renders [6.3] obvious through Muelich's iterative bitflip procedure at

modified step 440, which, upon detecting a nonzero syndrome weight (*see* [6.2], *supra*), sweeps through each bit position of the final raw PUF value in positional order ($i = 1, \dots, n$), tentatively replaces the bit at each position with its complement, and evaluates the resulting syndrome weight to determine whether that replacement corrects the error. *See* §III.A.1(c), *supra*; OKTA-1004, [0058]-[0064], FIG. 4; OKTA-1005, 3-5; OKTA-1003, ¶88.

As discussed below, the Wu-Müelich Combination would have used an iterative bitflip procedure to tentatively **replace**, one by one, each bit of the final raw PUF value with its complement (**arbitrary value**), and determine whether that replacement corrects the error by evaluating the resulting syndrome weight (**determining if the error in the PUF response is...corrected**). OKTA-1003, ¶89.

More specifically, Wu's error correction logic 140 "provide[s] for correcting errors in the generation of PUF keys from PUF cell array 120," and non-volatile memory 170 stores "error correction data 174" for use by the error correction logic. OKTA-1004, [0058]-[0064], FIG. 4; OKTA-1003, ¶90. During evaluation time, Wu's method 400 involves reading/evaluating the PUF cell array (step 410) and performing list-driven substitution using redirection list L (step 424) to yield a "final raw PUF value," which is then "processed by error correction logic 140" at step 440. *Id.* Wu contemplates an error correction routine applied to a PUF response (final

raw PUF value generated upstream in method 400) as part of the key generation pipeline. *Id.*

Müelich's Algorithm (reproduced below) depicts the iterative bitflip procedure that the Wu-Müelich Combination applies at modified step 440, rendering obvious the step of *replacing...[a] bit of the PUF response by an arbitrary value and determining if the error in the PUF response is corrected.* OKTA-1003, ¶91.

Algorithm 1: Bitflip Decoding

Input: $\mathbf{r} = \mathbf{c} + \mathbf{e}$
Output: decoding result $\hat{\mathbf{c}}$.

```

1 Calculate  $w = \text{WT}(\mathbf{r})$ 
2 if  $\text{wt}(w = 0)$  then
3    $\hat{\mathbf{c}} = \mathbf{r}$ 
4   return  $\hat{\mathbf{c}}$ 
5 for  $i = 1, \dots, n$  do
6    $\epsilon_i = \text{WT}(\mathbf{r} + \mathbf{u}_i^T) + \Delta_i$ 
7 Find  $j \in \{1, \dots, n\}$  with  $\epsilon_j = \min \epsilon_i$ 
8  $\mathbf{r} = \mathbf{r} + \mathbf{u}_j$ 
9 Goto 1

```

OKTA-1005, 5

More specifically, where the syndrome weight is nonzero (indicating the presence of an error per [6.2], *supra*), Müelich's Algorithm 1 proceeds to sweep each bit position, i , of the final raw PUF value in a predetermined positional order ($i = 1, \dots, n$). For each position i , Algorithm 1 computes $\epsilon_i = \text{WT}(\mathbf{r} + \mathbf{u}_i^T) + \Delta_i$, which quantifies the syndrome weight that would result if the bit at position i were replaced with its complement. OKTA-1005, 5 (Algorithm 1, lines 5-6). A POSITA would have understood that this per-bit computation is an example of *replacing one by one, in*

a predetermined order, each bit of the PUF response by an arbitrary value and determining if the error in the PUF response is corrected. Specifically, bit positions are evaluated in a predetermined positional order such that the $r + u_i^T$ term replaces the bit at position i with its complement (*arbitrary value* for a binary bit). Further, $WT(r + u_i^T)$ evaluates whether that replacement would correct the error by measuring the resulting syndrome weight. OKTA-1003, ¶92.

Müelich's per-bit evaluation is performed in a *predetermined order* within the meaning of [6.3] in various ways. For example, Algorithm 1 of Müelich prescribes an *order* of its sweep since line 5 executes the sweep as for $i = 1, \dots, n$, which sweeps the bit positions in fixed positional sequence from position 1 through position n . OKTA-1005, 5 (Algorithm 1, line 5). This sequence is established by the algorithm in advance of the decoder's execution and does not depend on the contents of the received word, the identity of any erroneous bit, or any other data-dependent parameter (i.e., is deterministic) and thus is *predetermined*. OKTA-1003, ¶93.

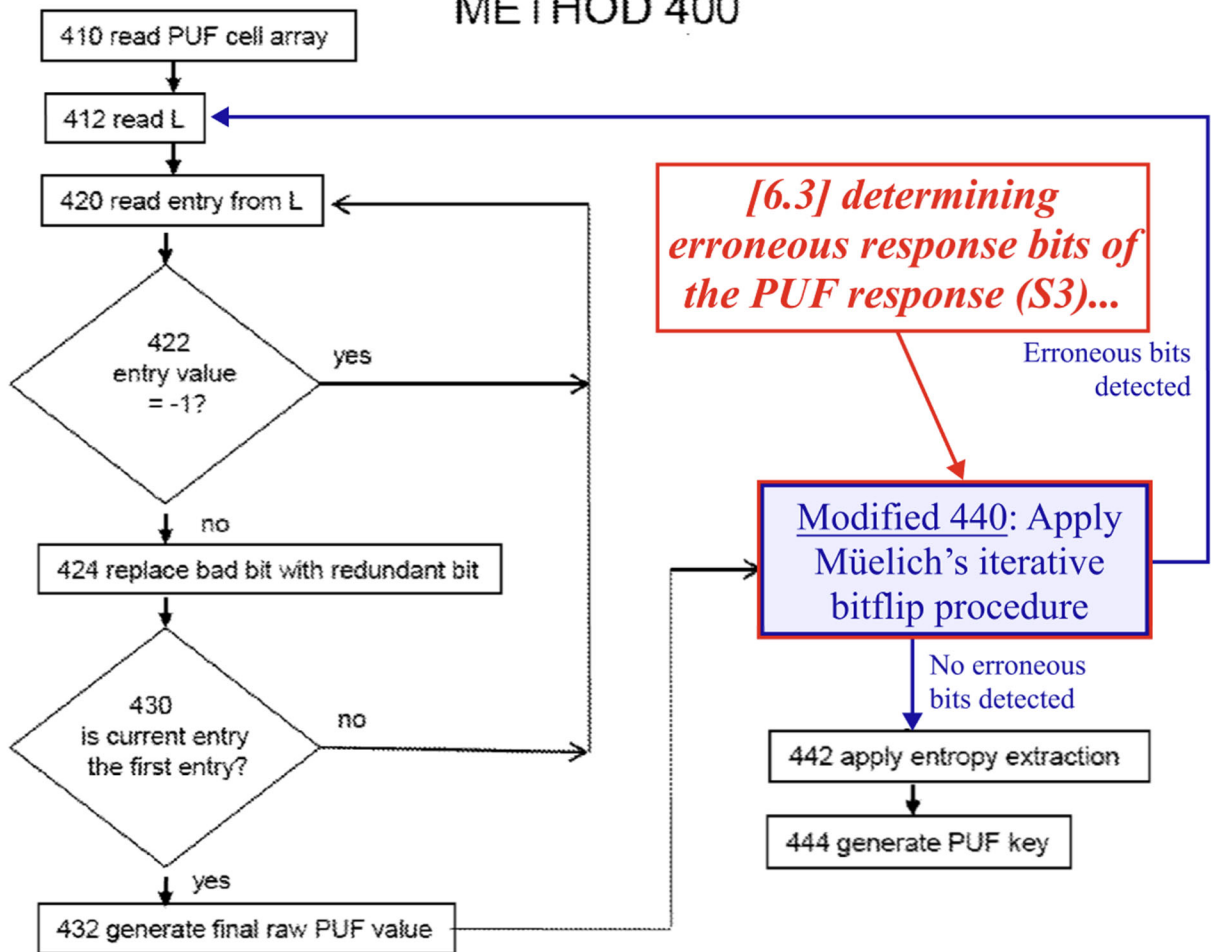
As another example, as would have been apparent to a POSITA, Müelich's process involves bit-by-bit replacements according to the calculation, at each iteration, of a minimal ε_j that indicates the next most likely erroneous bit. OKTA-1003, ¶94. Dr. Traynor explains that "the execution of these steps is deterministic (i.e., predetermined) in that the order in which steps may be executed based on the minimal ε_i calculation at each iteration." *Id.* (citing OKTA-1005, 5). This

understanding is also consistent with the '359 patent specification, which explains that an example of a bit-flipping process being performed in “a predetermined order” involves “knowledge about how reliable the PUF response bits are to test the least reliable bits first.” OKTA-1001, 5:12-25 (identifying “a predetermined order” as “test[ing] the least reliable bits first”). A POSITA would have understood and/or found it obvious that Muelich’s Algorithm 1 similarly may leverage the minimal ε_i calculation at each iteration to execute bit-flipping according to the value of each minimal ε_i , such that the mechanism produces a commit order in which less reliable bit positions are biased towards being committed earlier, with the bias arising from the contribution of Δ_i to minimal ε_i . OKTA-1003, ¶94.

To summarize, as discussed in §III.A.1(c), *supra*, the Wu-Muelich Combination implements Wu’s error correction stage at modified step 440 using Muelich’s iterative bitflip procedure. OKTA-1004, [0064], FIG. 4; OKTA-1005, 4-5; OKTA-1003, ¶95. When Muelich’s procedure sweeps bit positions $i = 1, \dots, n$ of the final raw PUF value and evaluates $\varepsilon_i = \text{WT}(r + u_i^T) + \Delta_i$ at each position, this involves, on a per-bit, predetermined-order basis, replacement and error-correction test operations recited in [6.3]. OKTA-1005, 5 (Algorithm 1, lines 5-6); OKTA-1003, ¶95. The Wu-Muelich Combination therefore satisfies [6.3], or at least renders it obvious, for any iteration of method 400 in which Muelich’s procedure determines a nonzero syndrome weight. OKTA-1003, ¶95.

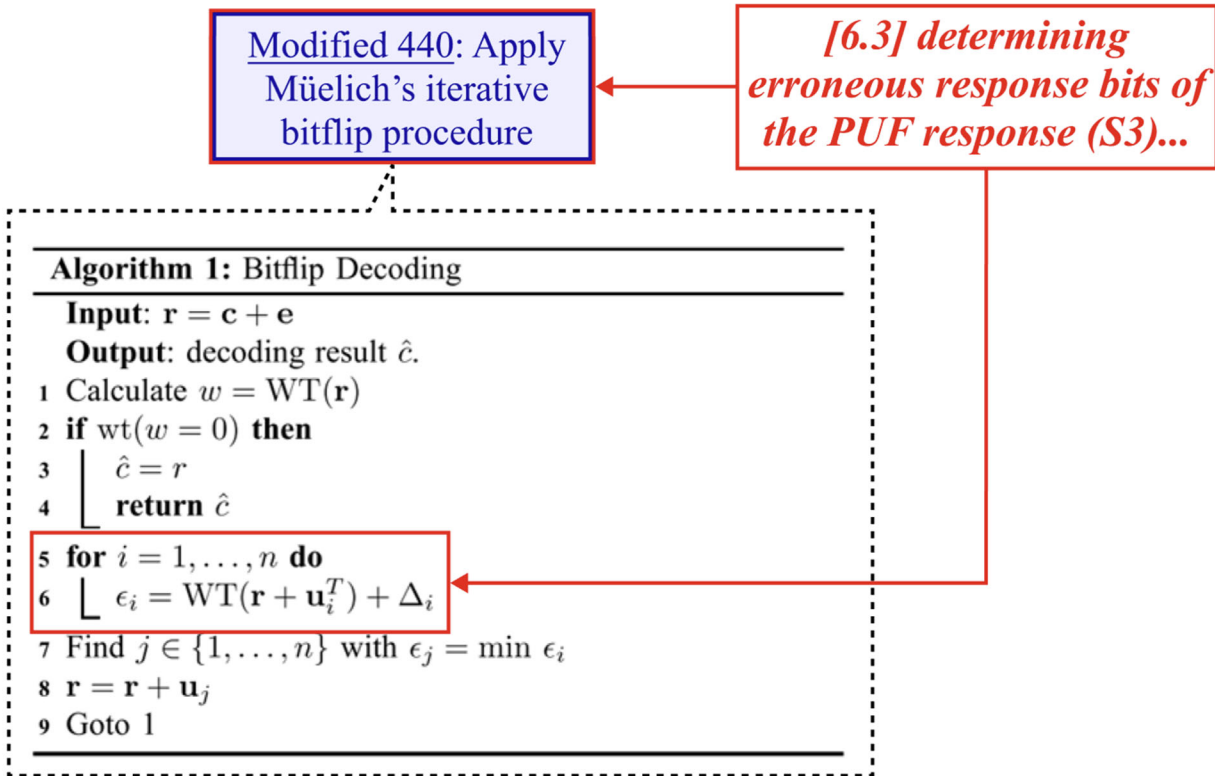
FIGURE 4
METHOD 400

Iteration = N



OKTA-1004, FIG. 4 (annotated)

Iteration = N



OKTA-1005, 5 (annotated)

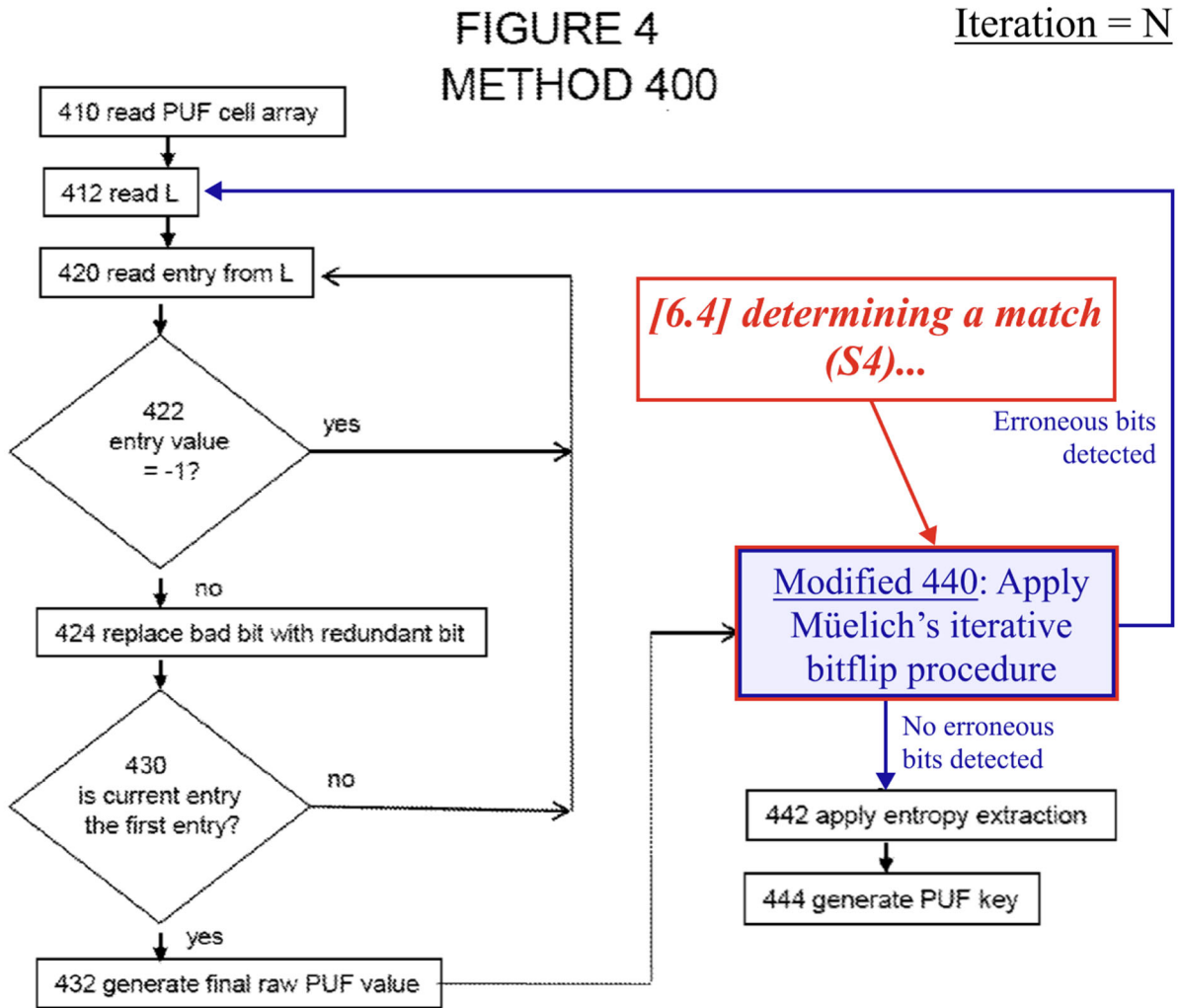
[6.4] determining a match (S4) between each determined erroneous bit and a selected additional output bit such that replacing the erroneous response bits by the matching additional output bits corrects the error in the PUF response,

The Wu-Muelich Combination renders [6.4] obvious in the context of the evaluation-time workflow discussed in [6.3], *supra*, where Muelich's bitflip procedure at modified step 440 (e.g., Algorithm 1, line 7) identifies an erroneous bit position in the final raw PUF value (i.e., a *determined erroneous response bit*). See §III.A.1(c), *supra*; OKTA-1004, [0064], FIG. 4; OKTA-1005, 3-5. Indeed, the Wu-

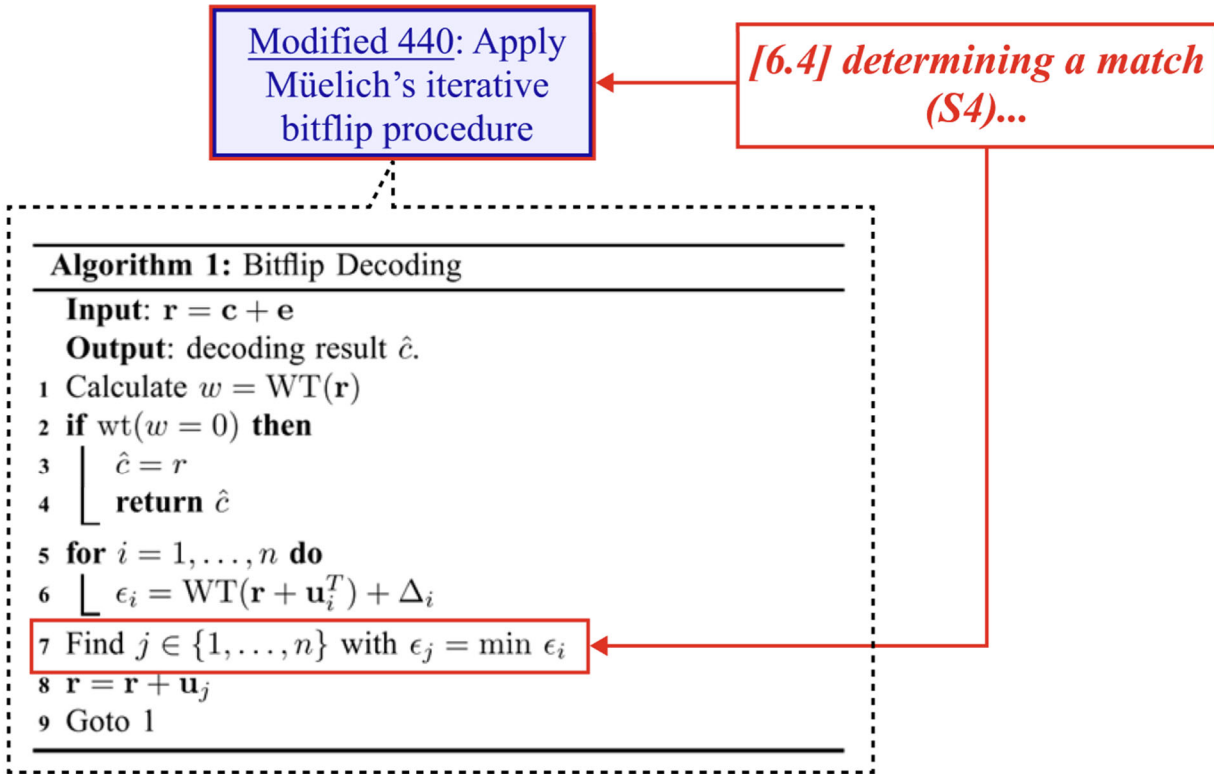
Müelich Combination takes the erroneous bit position, j , identified by Müelich's bit-flip procedure and *matches* it to an available redundant bit from the redundant PUF array 124 (*selected additional output bit*) via Wu's redirection list mapping. *Id.* The resulting (j , redundant-bit) pairing involves a *match* that *correct[s] the error in the PUF response* because Müelich's iterative bitflip procedure selects bit position j as the position whose flipping drives the syndrome weight towards zero (i.e., corrects the error). OKTA-1003, ¶96.

As discussed for [6.pre], *supra*, Wu's PUF cell array 120 includes a main PUF array 122 (*response bits*) and a redundant PUF array 124 (*additional output bits*), and redundancy logic 130 is configured to evaluate PUF cells and replace PUF cells failing stability requirements to reduce noise and error rate in PUF key generation. OKTA-1004, [0023]-[0024]. In Wu, the redirection list L stored in non-volatile memory 172 records, for each main array bit position to be replaced, an entry identifying the redundant bit selected to replace it. OKTA-1004, [0059]-[0060], FIG. 4; OKTA-1003, ¶97. A POSITA would have understood that this association constitutes *determining a match* because the redirection-list entry establishes the correspondence between (i) the identified erroneous bit position in the main array (e.g., bit position j identified at modified step 440 by Müelich's iterative bitflip procedure), and (ii) a particular redundant bit selected from redundant PUF array

124 for use in replacement at that position. See OKTA-1003, ¶97 (citing OKTA-1004, [0024], [0059]-[0060], FIG. 4).



Iteration = N



OKTA-1005, 5 (annotated)

[6.5] storing said determined match in a repair list (S5),

The Wu-Muelich Combination *stores* a redundant bit selected from redundant PUF array 124 in [6.5] in a redirection list L stored in non-volatile memory 172 (*repair list*). OKTA-1003, ¶98; §III.A.1(c), *supra*. Consistent with Wu, each entry within the redirection list L identifies a main-array bit position along with the redundant bit selected to replace it. OKTA-1004, [0028]-[0030]. The stored match persists in non-volatile memory 172 across successive iterations of method 400. At

the beginning of each subsequent iteration, Wu's step 412 reads the current contents of L from non-volatile memory 172 and provides them as input to the sequence of steps 420, 422, and 424. OKTA-1004, [0059]-[0061], FIG. 4. This enables step 424 of a subsequent iteration to apply a stored match as a list-driven replacement. *See* [6.6], *infra*; OKTA-1003, ¶98.

[6.6] replacing ([S6]) in the PUF response said erroneous response bits with the matching additional output bits in the repair list.

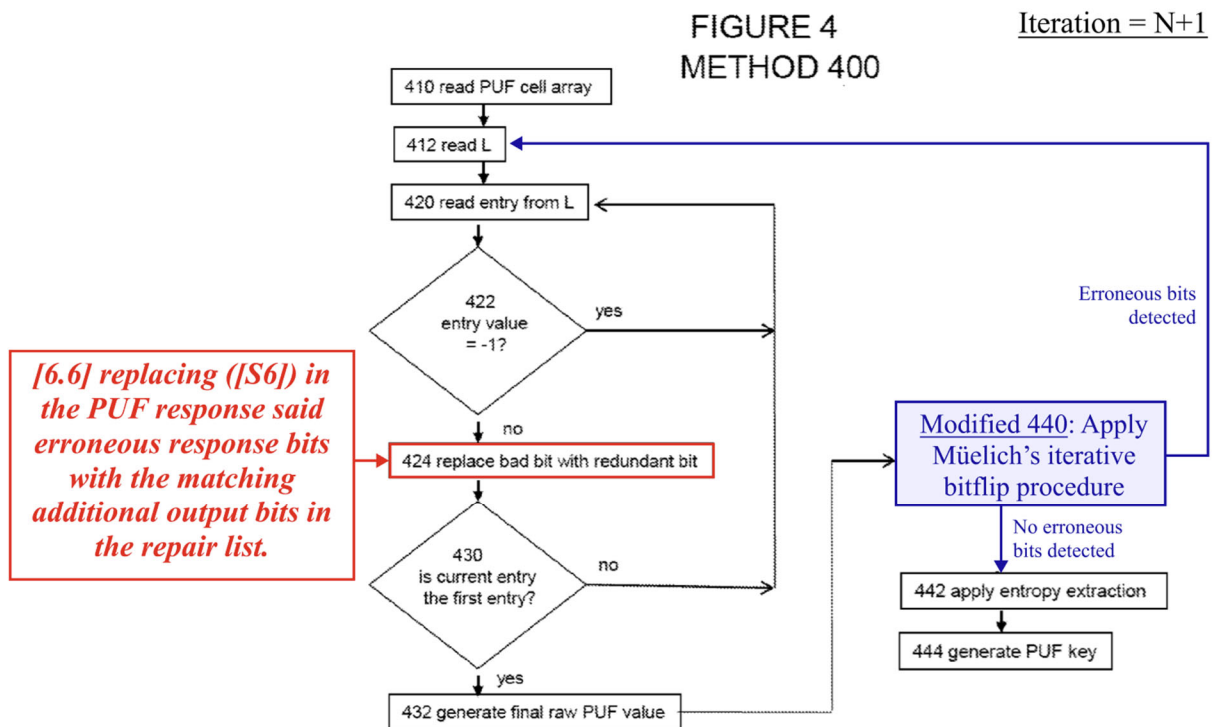
The Wu-Müelich Combination provides the requirements of [6.6] through Wu's list-driven replacement mechanism at step 424, applied during the iteration of the method 400 that follows the iteration in which the current match is written to redirection list L. OKTA-1003, ¶99; *see* §III.A.1(c), *supra*. Following the storage of a current match in non-volatile memory 172, control reverts from modified step 440 to step 412 of a subsequent iteration where the updated redirection list L is read. OKTA-1004, [0059], FIG. 4. The execution of steps 420, 422, and 424 then proceeds through entries of the redirection list L. Once traversal reaches the stored entry, step 424 substitutes the redundant bit's value at bit position *j* in the PUF response (*replacing...erroneous response bits with the matching additional output bits in the repair list*). *Id.*, [0059]-[0063], FIG. 4; OKTA-1003, ¶99.

The configuration of the Wu-Müelich Combination discussed above is also consistent with the '359 specification's disclosure of repair list matching across

successive PUF response generations. For example, as shown in Figure 4, during a generation of a PUF response, “the PUF response bits are first corrected according to the matching stored in the repair list,” after which additional erroneous bits (if any) are identified and added to the list. OKTA-1001, 6:28-40, FIG. 4. Additionally, the method “may be applied repeatedly at successive generations of the PUF response” and that, at the first step of a subsequent generation, previously-stored matchings are used for “replacing in the PUF response the matched response bits with the matching additional output bits in the repair list.” *Id.*, 5:22-28. Consistent with this disclosure, the Wu-Müelich Combination similarly stores a match identified in the modified step 440 in the redirection list L for use in the list-driven substitution at step 424 of a subsequent iteration of method 400. This in-memory update corresponds to replacing, in the PUF response, the erroneous bit at position j with the matching additional output bit (the redundant bit) identified in the repair list updated in the preceding step S5. OKTA-1003, ¶100.

The configuration discussed above follows Wu’s teaching of using the redirection list values to update PUF responses. Wu explains that “the value of the redundant bit corresponding to the current entry is used to replace the value of the bit whose address ... is stored in the current entry,” yielding an updated value where the bad bit is replaced. OKTA-1004, [0061]; OKTA-1003, ¶101. While Wu’s baseline method performs this replacement at step 424 for previously known errors,

in the Wu-Müelich Combination, Müelich's iterative routine identifies the erroneous bit position j , and Wu's redirection list mapping selects an available redundant bit from redundant PUF array 124 as the replacement value for that position (ensuring the final PUF key is generated from corrected data). OKTA-1003, ¶101. Furthermore, because the redirection list L is updated in non-volatile memory (*see* [6.5], *supra*), subsequent invocations of method 400 involve applying this replacement at step 424. OKTA-1004, [0061]; OKTA-1003, ¶101.



OKTA-1004, FIG. 4 (annotated)

(b) Claim 7

[7.1] The method according to claim 6, wherein said device being configured for having access to a memory storing a preventive toggling list of the least reliable response bits of said physical unclonable function,

The Wu-Müelich Combination provides integrated circuit 100 (*said device*) configured to access a sorted list of response-bit positions stored in non-volatile memory 172, where the list is sorted from least reliable to most reliable based on per-bit reliability characterizations produced by Müelich's multi-readout mechanism during device setup (*preventive toggling list of the least reliable response bits*). OKTA-1005, 4; OKTA-1004, [0010], [0017]-[0018], [0028]-[0030]; OKTA-1003, ¶102.

Müelich describes a per-bit reliability characterization mechanism that classifies response-bit positions as least reliable or more reliable. *See* OKTA-1005, 4; OKTA-1003, ¶103. During device setup (e.g., during device enrollment before evaluation-time operation), Müelich's mechanism obtains multiple readouts of each response bit position (e.g., $m = 3$ readouts) and computes, for each bit position $i \in \{1, \dots, n\}$, a reliability score Δ_i based on whether the readouts agree at that position. For example, when $\Delta_i = \delta_1$, all readouts agree (indicating higher reliability), and $\Delta_i = \delta_2$ when the readouts disagree (indicating lower reliability), where $\delta_1 > \delta_2 > 0$. OKTA-1005, 4. A POSITA would have understood that per-bit reliability characterization identifies, for each device, which response bit positions of the PUF

are least reliable (positions that exhibit non-reproducible behavior across readouts). OKTA-1003, ¶103. In Muelich, these reliability scores are used during evaluation-time error correction at Algorithm 1, where Δ_i is incorporated into the ε_i computation (see [6.4], *supra*) favor bit positions that are less reliable. OKTA-1005, 5; OKTA-1003, ¶103.

The configuration discussed above is supported by Wu's suggestion of generating per-bit characterization data during device setup and persistent storage of the characterization data in non-volatile memory. In Wu's method 300, PUF cell array 120 is characterized during manufacturing time under varying conditions to identify stable cells. See OKTA-1004, [0035]-[0045], FIG. 3. The results of that characterization are stored in redirection list L in non-volatile memory 172 and accessible for evaluation-time use during method 400. *Id.*, [0028]-[0030], [0045], [0059]. A POSITA would have recognized that this storage architecture (list-structured per-bit data in non-volatile memory 172) supports persistent storage of Muelich's setup-time reliability characterization for evaluation-time access (e.g., in the modified step 440 as discussed for [6.2]-[6.6], *supra*). OKTA-1003, ¶104.

In the Wu-Muelich Combination, Muelich's setup-time reliability characterization is applied to each response bit position of main PUF array 122, producing a per-bit Δ_i value for each of the n response bit positions. OKTA-1005, 4; OKTA-1003, ¶105. The combination sorts these bit positions from least reliable

($\Delta_i = \delta_2$) to most reliable ($\Delta_i = \delta_1$) and stores the sorted list of bit position references in non-volatile memory 172, which is accessible alongside redirection list L. OKTA-1004, [0028]-[0030]; OKTA-1003, ¶105. Consistent with Wu's architecture for non-volatile memory 172, during evaluation-time operation of method 400, integrated circuit 100 is thereby configured to access the sorted list (stored in memory configured for evaluation-time access) (*configured for having access to a memory storing a preventive toggling list of the least reliable response bits*). OKTA-1004, [0059], FIG. 4; OKTA-1003, ¶105.

[7.2] determining erroneous response bits of the PUF response to be repaired (S3) comprises replacing first bits of the Preventive toggling list before replacing other bits of the PUF response.

The Wu-Müelich Combination renders [7.2] obvious through reliability-weighted operation of Müelich's iterative bitflip procedure, which causes bit positions identified as least reliable by a sorted list of bit position references in non-volatile memory 172 (*preventive toggling list*) to be committed for replacement (*replacing first bits*) before more reliable bit positions are committed (*replacing other bits*). OKTA-1005, 4-5; *see* OKTA-1004, [0024], [0028]-[0030]; OKTA-1003, ¶106.

Specifically, Müelich's Algorithm 1 computes $\varepsilon_i = \text{WT}(\mathbf{r} + \mathbf{u}_i^T) + \Delta_i$ at each bit position i (*see* [6.3], *supra*), where Δ_i is the per-bit reliability score that governs the ordering of bit position references in non-volatile memory 172 (*see* [7.1], *supra*).

OKTA-1005, 4-5. Less-reliable positions are assigned lower Δ_i values and thus their corresponding ε_i values are lower than those of more-reliable positions. OKTA-1003, ¶107. Because ε_i incorporates both the error-correction metric ($WT(r + u_i^T)$) and the reliability weight (Δ_i), Algorithm 1's execution favors error-correcting flips at less reliable bit positions. OKTA-1005, 5; OKTA-1003, ¶107. At each iteration, Algorithm 1 commits the single flip whose bit position minimizes ε_i , such that less-reliable positions (corresponding to the first entries of the preventive toggling list) are committed first, followed by progressively more reliable positions in subsequent iterations until the syndrome weight is driven to zero. OKTA-1005, 5; OKTA-1003, ¶107.

The reliability-weighted commit order described above is consistent with the '359 specification's description of the "preventive toggling list." Specifically, the preventive toggling list "comprise[s] the reference numbers of the response bits ... sorted from the least to the most reliable response bits," such that "[t]he most probable erroneous bits may therefore be tested first." OKTA-1001, 5:15-21. As discussed above, the Wu-Müelich Combination's execution of Müelich's Algorithm 1 (in which least reliable bit positions are evaluated prior to more reliable bit positions) functions in a similar manner. The Wu-Müelich Combination therefore satisfies [7.2], or at least renders it obvious. OKTA-1003, ¶108.

(c) Claim 8

[8.1] The method according to claim 6, wherein said electronic device is configured for access to a memory storing said repair list comprising at least one match between a response bit and an additional output bit,

The Wu-Müelich Combination provides an electronic device (e.g., integrated circuit 100) (*said electronic device*) configured to access non-volatile memory storing a redirection list L (*memory storing said repair list*), where redirection list L specifies at least one match between a main array response bit position and a redundant array additional output bit source (*match between a response bit and an additional output bit*). OKTA-1004, [0058]-[0072]; OKTA-1003, ¶109.

As discussed for [6.5] and [6.6], *supra*, in Wu, non-volatile memory 172 stores redirection list L, and Wu's evaluation-time workflow (method 400) includes reading redirection list L from the non-volatile memory (step 412) for use in repairing a generated PUF response. OKTA-1004, [0059]; OKTA-1003, ¶110. A POSITA would have understood that this configuration (e.g., retrieving list data from non-volatile storage during evaluation-time processing) reflects the Wu-Müelich Combination device *configured for access to a memory storing the repair list*. OKTA-1003, ¶110.

Moreover, as discussed for [6.pre] and [6.4], *supra*, PUF cell array 120 includes a main PUF array 122 (*response bits*) and a redundant PUF array 124 (*additional output bits*). OKTA-1004, [0020]-[0022]; OKTA-1003, ¶111. The

redirection list L records, for a selected main array bit position to be repaired (*response bit* position), which redundant bit is selected as a replacement source (*additional output bit*). OKTA-1004, [0024], [0059], [0067]-[0069]. For example, during evaluation time, entries of the redirection list L are iterated and, for each entry, “the value of the redundant bit corresponding to the current entry is used to replace the value of the bit whose address...is stored in the current entry.” OKTA-1004, [0061]; OKTA-1003, ¶111. A POSITA would have understood that each stored entry embodies at least one match (i.e., an association) between a response bit address in the main PUF array 122 and a selected redundant bit source in the redundant PUF array 124. OKTA-1003, ¶111.

As discussed in §III.A.1(c), in the Wu-Müelich Combination, Wu’s error-correction stage (modified step 440) incorporates Müelich’s iterative bitflip procedure to identify residual erroneous bit positions and then persists those results into Wu’s existing redirection list framework by updating and storing the redirection list L in non-volatile memory for subsequent evaluation-time use across successive iterations of method 400. OKTA-1004, [0024], [0059], [0067]-[0069]; OKTA-1005, 1, 3-5; OKTA-1003, ¶112. The Wu-Müelich Combination thereby incorporates Wu’s configuration for accessing a memory storing the repair list during operation (step 412), and reinforces that configuration by persisting additional match entries (between main PUF array bit positions and redundant bit

sources) into the stored repair list structure over time across successive iterations of method 400, including to address post-manufacturing aging-related instability. OKTA-1004, [0059]; OKTA-1005, 1, 3, 4; OKTA-1003, ¶112.

[8.2] the step of generation of a PUF response (S1) comprises replacing in the PUF response said matched response bits with the matching additional output bits in the repair list.

See [6.6], *supra*. During the step recited in [6.1] (***S1***), the Wu-Müelich Combination forms the final raw PUF value (***PUF response***) using the redirection list L to ***replace*** one or more bits in a PUF response (***said matched response bits*** in the ***PUF response***) with a corresponding redundant bit value (***matching additional output bits***), consistent with Wu's redirection list replacement mechanism. OKTA-1004, [0058]-[0072]; OKTA-1003, ¶113.

More specifically, as discussed for [6.1] and [6.6], *supra*, the evaluation-time workflow provided by the Wu-Müelich Combination involves reading/evaluating the PUF cell array to obtain an initial raw PUF value (step 410) and applying redirection list substitutions (using redirection list L read from non-volatile memory 172 at step 412) to form an “updated PUF value” from which a “final raw PUF value” is generated. OKTA-1004, [0063]; OKTA-1003, ¶114. Wu explains that, for each entry in redirection list L, “the value of the redundant bit corresponding to the current entry is used to replace the value of the bit whose address ... is stored in the current entry” (step 424). OKTA-1004, [0061]; OKTA-1003, ¶114. A POSITA

would have understood that this list-driven replacement is part of generating the final raw PUF value (*PUF response*), and therefore this process involves *replacing* matched response bit positions (*said matched response bits*) with corresponding redundant-bit values identified in the repair list (*the matching additional output bits*). OKTA-1003, ¶114.

Further, as discussed in §III.A.1(c), the Wu-Müelich Combination also persists newly-identified erroneous bit positions identified during the modified error correction stage at modified step 440 (*see* [6.3]-[6.5], *supra*). OKTA-1003, ¶115. This involves updating redirection list L to include additional match entries, which are then available to be applied by the same redirection-list-driven substitution mechanism during subsequent iterations of method 400. OKTA-1004, [0058]-[0072]; OKTA-1005, 1, 3, 4; OKTA-1003, ¶115. Specifically, during a subsequent iteration of method 400, step 412 reads the updated redirection list L from non-volatile memory 172 and the process involving steps 420, 422, and 424 involves applying the newly-added match entry at step 424. This involves substituting the redundant bit's value at the erroneous position *j* during the generation of that subsequent iteration's final raw PUF value. OKTA-1004, [0059]-[0063]; OKTA-1003, ¶115.

(d) Claim 9

[9] The method of claim 6, wherein said additional output bit are bits among most reliable output bits of said physical unclonable function.

As discussed for [7.1], the Wu-Müelich Combination selects redundant bits from redundant PUF array 124 (**additional output bits**) using reliability metrics calculated using Müelich's reliability characterization mechanism (Δ_i) (**most reliable output bits**). OKTA-1005, 4. During device setup, Müelich's multi-readout mechanism characterizes each PUF cell of PUF cell array 120 (including cells of both main PUF array 122 and redundant PUF array 124) by obtaining multiple readouts and computing a per-bit reliability score, Δ_i , based on readout agreement, with $\Delta_i = \delta_1$ for reliable cells (all readouts agree) and $\Delta_i = \delta_2$ for less reliable cells (readouts disagree). *Id.*; see OKTA-1004, [0028]-[0030], [0059]; OKTA-1003, ¶116.

As discussed for [7.1], *supra*, Müelich's mechanism obtains multiple readouts (e.g., $m = 3$ readouts) of each response bit position, evaluates readout agreement, and assigns a reliability score Δ_i to each bit position. OKTA-1005, 4. A POSITA would have understood that this characterization mechanism applies uniformly to PUF cells in PUF cell array 120 (both main PUF array 122 and redundant PUF array 124) because the underlying physical phenomenon (measurement-to-measurement variability in a PUF cell's output) is a property of the cell itself, not of its functional

role as a “main” or “redundant” bit. OKTA-1003, ¶117. The output of the characterization (e.g., Δ_i per bit) is therefore available for every cell in PUF cell array 120, and selection of ***additional output bits*** can draw from those redundant PUF array 124 cells characterized as most reliable ($\Delta_i = \delta_1$). *Id.*

Moreover, the selection of reliable redundant bits is consistent with the repair mechanism provided by the Wu-Müelich Combination because it facilitates execution across successive iterations of method 400. OKTA-1003, ¶118. As discussed in §III.A.1(c) and [6.6], *supra*, over multiple iterations, matches associated with erroneous bit positions, j , are written to redirection list L and persisted in non-volatile memory 172. OKTA-1004, [0059]-[0063], FIG. 4; OKTA-1003, ¶118. A POSITA would have recognized that if redundant bits were themselves unstable (i.e., their value varied across readouts), substitution at step 424 would propagate that instability into the final raw PUF value, undermining the repair mechanism. OKTA-1003, ¶118. Accordingly, to avoid such instability, the POSITA would have configured the Wu-Müelich Combination to select redundant bits from among the most reliable PUF output bits (i.e., cells for which Müelich’s multi-readout characterization produced $\Delta_i = \delta_1$, indicating consistent output across readouts). OKTA-1005, 4; OKTA-1003, ¶118.

The reliability-based selection is also consistent with Wu’s architecture, which treats redundant PUF array 124 as a repair resource for replacing “bad bits”

in main PUF array 122. OKTA-1004, [0023]-[0024]. A POSITA would have understood that this functional role presupposes that the redundant bits are themselves reliable because Wu contemplates device setup characterization of PUF cells under varying conditions (method 300) to identify unstable cells. OKTA-1003, ¶119; *see* OKTA-1004, [0035]-[0045], FIG. 3. A POSITA implementing the Wu-Müelich Combination thereby applies a similar device setup characterization principle to redundant PUF array 124 (e.g., using Wu's method 300 process, Müelich's multi-readout mechanism, or both) and selects the most reliable redundant cells for use as ***additional output bits*** in match entries. OKTA-1003, ¶119.

(e) **Claim 11**

[11] The method of claim 9, wherein detecting an error in the PUF response (S2) comprises using an error correcting code.

The Wu-Müelich Combination renders claim 11 obvious because certain implementations detect PUF errors ***using an error correcting code***, e.g., using Müelich's iterative bitflip procedure incorporated into Wu's modified step 440. OKTA-1003, ¶120; *see* OKTA-1004, [0058]-[0072]; OKTA-1005, 1, 3, 4. Such implementations provide an evaluation-time workflow that generates a "final raw PUF value" and then processes that word using "error correction logic 140" at modified step 440, which Wu permits use of "any known error correction technique." OKTA-1004, [0025], [0064]; OKTA-1003, ¶120. A POSITA would

have understood and found obvious that implementing the error-correction stage of modified step 440 using Muelich's iterative bitflip procedure entails use of an ECC (e.g., an LDPC code defined by a parity-check matrix H) to detect whether the final raw PUF value contains an error condition. OKTA-1005, 1, 3, 4; OKTA-1003, ¶120.

As discussed for [6.2], *supra*, in Muelich, error correction in PUF-based key generators may be improved by constructing an LDPC code (via a parity-check matrix H) and decoding a newly-generated PUF response, r , using syndrome weight criterion. OKTA-1005, 1, 3, 4; OKTA-1003, ¶121. In Muelich's iterative bitflip procedure (e.g., Algorithm 1), a syndrome weight ($w = WT(r)$) is computed for a received word r and evaluated. OKTA-1005, 5.

A POSITA would have understood that syndrome weight evaluation is an example of an ECC-based detection operation. For example, if the syndrome weight is equal to zero (e.g., $w = 0$), the word is treated as a valid codeword (no error), whereas if the criterion is not satisfied (e.g., $w \neq 0$), the received word includes an error condition (i.e., an "error pattern" e) and therefore an error is detected in the PUF response. OKTA-1005, 5; OKTA-1003, ¶122. Only upon such detection does the decoding routine proceed to corrective operations (e.g., evaluating candidate bit flips), confirming that the initial syndrome weight computation is the step by which an error is detected using the ECC. OKTA-1005, 1, 3, 4; OKTA-1003, ¶122.

(f) Claim 14

Claim 14 is substantially similar to claim 6, reciting a similar series of steps in Beauregard⁵ form with generic preamble language identifying conventional computer components. Accordingly, the Wu-Müelich Combination renders the limitations of claim 14 obvious in the same manner as explained above for corresponding claim limitations listed in the table below. OKTA-1003, ¶123. In addition, Wu plainly provides the active electronic primitives, PUF circuits, processor, and memory of [14.pre]. *See, e.g.*, OKTA-1004, [0010], [0015]-[0018], [0020]-[0030], FIG. 1.

Limitation	Corresponding Limitation
[14.pre]	[6.pre]
[14.1]	[6.1]
[14.2]	[6.2]
[14.3]	[6.3]
[14.4]	[6.4]
[14.5]	[6.5]

⁵ Beauregard claims are typically treated as method/process claims. *E.g., Digital-Vending Services Intern., LLC v. University of Phoenix, Inc.*, 672 F.3d 1270, 1275 n.1 (Fed. Cir. 2012).

[14.6]	[6.6]
--------	-------

B. [GROUND 1B] – Claim 10 Is Obvious Over Wu-Müelich-Tuyls

1. Wu-Müelich-Tuyls

(a) Wu (OKTA-1004)

See §III.A.1(a), *supra*; OKTA-1003, ¶¶49-53.

(b) Müelich (OKTA-1005)

See §III.A.1(b), *supra*; OKTA-1003, ¶54.

(c) Tuyls (OKTA-1009)

Tuyls is directed to a “programming device arranged to obtain and store a random bit string in a memory device,” OKTA-1009, [0008]-[0021], FIG. 1. Tuyls discloses techniques for using a “hash over at least a part of the data in the multiple memory cells” for the purpose of identifying “faulty memory cells.” OKTA-1009, [0016], [0100]-[0114]. Specifically, Tuyls discloses an “authentication unit 162” that computes a “first hash” over a set of data. OKTA-1009, [0100]. Next, a “validation unit 164” compares the computed first hash with a “stored first hash” and determines if the two hash values match. OKTA-1009, [0101]-[0114]. If the hash values differ, it is determined that an error has occurred in the data. *Id.*; OKTA-1003, ¶55.

(d) Combination of Wu, Muelich, and Tuyls

As discussed above, Wu contemplates an evaluation-time workflow in which a PUF response is generated and then processed by an “error correction logic” stage (step 440 of method 400 depicted in Figure 4). OKTA-1004, [0058], [0064], FIG. 4. Method 400 is performed “during evaluation time,” e.g., by a customer, including actions performed by the internal operation of IC 100. *Id.*, [0058]. After reading/evaluating the PUF cell array (step 410) and performing a list-driven replacement of identified “noisy” (bad) bits using redundant bits (step 424), a “final raw PUF value” is then “processed by error correction logic 140” at step 440. *Id.*, FIG. 4, [0063]-[0064]; *supra* §III.A.1; OKTA-1003, ¶124.

Wu itself recognizes that it is desirable to enable “post-manufacturing (e.g., during customer use) identification and replacement of bad bits,” for example, to “counter device aging effects.” *Id.*, [0073]. Wu also expressly leaves the choice of the step-440 “error correction logic” open, stating that “[a]ny known error correction technique may be used,” and describes storage of error-correction data in non-volatile memory for later use during operation. *Id.*, [0025]. Consistent with this disclosure, a POSITA would have sought to supplement the error-correction stage of step 440 with well-known PUF-oriented diagnostic/error-correction techniques that iteratively identify/correct erroneous bit positions in the post-manufacturing instances contemplated by Wu. OKTA-1003, ¶125.

Like Muelich, Tuyls discloses techniques for determining when errors have occurred in a PUF. OKTA-1005, 1, 3-4; OKTA-1009, [0008]-[0021], [0095]-[0099]; OKTA-1003, ¶126. Specifically, Tuyls' "authentication unit 162" computes a "first hash" and compares the computed first hash with a "stored first hash" and determines if the two hash values match. OKTA-1009, [0100]-[0114]. If the hash values differ, it is determined that an error has occurred in the data. *Id.* A POSITA would have combined Tuyls' additional error detection techniques into the Wu-Muelich Combination for the following reasons. OKTA-1003, ¶126.

First, Tuyls discloses that its techniques are computationally more efficient than "large error correcting codes." OKTA-1009, [0016]. Tuyls' techniques are thus an efficient way to implement Wu's described "error correction logic" on a portable electronic device where computation resources were limited (e.g., a mobile phone). OKTA-1004, [0058], [0064], FIG. 4; OKTA-1009, [0017]-[0018] (disclosing that the memory may be "a mobile electronic device, such as a mobile phone"); OKTA-1003, ¶127.

Second, Tuyls discloses that its techniques can identify faulty data faster than traditional error correcting codes. OKTA-1009, [0016] ("By computing multiple hashes over overlapping data, faulty memory cells may be found even more quickly."). As with computational efficiency, a POSITA would have recognized that a more time-efficient method of error detection would reduce computational

requirements. OKTA-1004, [0058], [0064], FIG. 4; OKTA-1009, [0017]-[0018]; OKTA-1003, ¶128.

A POSITA would have also had a reasonable expectation of success in implementing Tuyls' additional error detection techniques into the Wu-Müelich Combination. OKTA-1003, ¶129. Wu expressly identifies step 440 as the evaluation-time stage for applying error correction logic to the final raw PUF value and expressly permits the use of "any known error correction technique," including with error-correction data stored in non-volatile memory for later use. OKTA-1004 [0025], [0058], [0063], [0064]. Tuyls provides a computationally efficient method of detecting errors in data based on comparing a hash of the data with a reference hash. OKTA-1009, [0016], [0100]-[0114]. Incorporating Tuyls' error detection techniques into Wu's step-440 error-correction logic and Müelich's iterative bitflip procedure would have been implementable through routine engineering choices without invention. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 415–21 (2007); MPEP § 2143; OKTA-1003, ¶129.

2. Analysis

(a) Claim 10

[10.1] The method of claim 6, wherein said device is configured for access to a memory storing a reference hash value of said response bits,

Tuyls discloses a "validation unit 164" that compares a computed "first hash" to a "stored first hash" that is saved in memory (*said device is configured for access*

to a memory storing a reference hash value of said response bits). OKTA-1009, [0100]-[0114]. As discussed below, the validation unit 164 references the stored first hash for comparison against a computed first hash when determining if errors have occurred in the data represented by the computed first hash. *Id.*; OKTA-1003, ¶130. In the Wu-Müelich-Tuyls Combination, this involves checking if there is an error before performing iterative bit flipping identifying the error using Müelich’s iterative bitflip procedure. *See* §III.B.1(d), *supra*.

[10.2] detecting an error in the PUF response (S2) comprises: generating a hash value by applying a hash function on said bits of the PUF response, comparing said hash value with said stored reference hash value.

Tuyls also discloses an “authentication unit 162” that computes a “first hash” over a set of data (*generating a hash value by applying a hash function on said bits of the PUF response*). OKTA-1009, [0100]. Next, a “validation unit 164” compares the computed first hash with a “stored first hash” and determines if the two hash values match (*comparing said hash value with said stored reference hash value*). OKTA-1009, [0101]-[0114]. If the hash values differ, it is determined that an error has occurred in the data (*detecting an error in the PUF response ...*). *Id.*; OKTA-1003, ¶131.

**C. [GROUND 1C] – Claims 1-3, 5, 12, and 13 Are Obvious
Over Wu-Müelich-Yang**

1. Wu-Müelich-Yang

(a) Wu (OKTA-1004)

See §III.A.1(a), *supra*; OKTA-1003, ¶¶49-53.

(b) Müelich (OKTA-1005)

See §III.A.1(b), *supra*; OKTA-1003, ¶54.

(c) Yang (OKTA-1006)

Yang discloses a memory including a “physically unclonable function (PUF) unit, a controller and a memory array” where the PUF is used to “provide a random bit pool.” OKTA-1006, [0004], [0010]-[0019], FIG. 1. Yang’s controller “extract[s] a random bit sequence from the random bit pool in the PUF unit 105” for the purposes of generating a “true random number” (TRN). OKTA-1006, [0014]-[0015], FIG. 1. Yang discloses that these techniques allow “secure data storage, unique identity generation, true random number generation and secure key storage,” while “protecting data from unauthorized access and enhancing data security.” OKTA-1006, [0019]; OKTA-1003, ¶56.

(d) Combination of Wu, Müelich, and Yang

A POSITA would have known that PUF error correcting techniques presented security vulnerabilities, as acknowledged by Wu and Müelich. OKTA-1004, [0023] (“it is desirable for the testing process to not reveal information ... for that would

compromise the security benefits provided to the customer”); OKTA-1003, ¶132. For example, PUF error correction presents various “side-channels” where information about the PUF can be exposed from monitoring how the PUF was repaired. OKTA-1007, [0017]-[0019].

Contemporaneous literature corroborates the existence of these vulnerabilities and describes that “power consumption” and “time consumption” can be monitored during the repair process to infer information about the PUF – the same vulnerabilities described by the ’359 patent. OKTA-1007, [0017]-[0019], [0155]; OKTA-1001, 5:9-12. Other PUF publications similarly acknowledge such vulnerabilities. OKTA-1008, 1-2; OKTA-1009, [0021]-[0025]. For example, an adversary could time a PUF repair operation and infer what bits of the PUF were replaced based on the elapsed repair time, or alternatively, could measure power consumption of the circuit to determine when repair operations were occurring. *Id.*; OKTA-1003, ¶133.

In view of these well-known vulnerabilities, a POSITA would have been motivated to implement the error correcting techniques of the Wu-Müelich combination in a random and non-deterministic manner. *Id.* Indeed, “[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a POSITA has good reason to pursue the known options within his or her technical grasp.” *KSR*, 550 U.S. at 421. Here, there was a

well-known need to ensure that PUF error correction was performed without exposing PUF information to side-channels. OKTA-1007, [0017]-[0019], [0155]; OKTA-1008, 1-2; OKTA-1009, [0021]-[0025]. And, in the context of the Wu-Müelich combination, a POSITA would have recognized and found obvious that there were only a finite number of ways that bits could be flipped according to Müelich – either in-order, or randomly. *See supra*, §III.A.1(c). From a security-focused perspective, the latter would have been preferable to a POSITA that was mindful of the vulnerabilities presented by side-channels, and specifically, because an in-order repair process could have been used by an attacker to identify the status of specific PUF bits as repairs occurred. OKTA-1007, [0017]-[0019], [0155]; OKTA-1008, 1-2; OKTA-1009, [0021]-[0025]; OKTA-1003, ¶134.

A POSITA mindful of a security-inspired need to implement the Wu-Müelich combination's techniques in a random and non-deterministic manner would have encountered Yang. OKTA-1006, [0004], [0010]-[0019], FIG. 1; OKTA-1003, ¶135. Yang's process of randomly selecting bits from a PUF provide an adequate level of randomness that protects the Wu-Müelich combination from side-channel attacks, as an attacker monitoring repair time or power consumption has no way of correlating repairs to a particular PUF bit. *Id.*; OKTA-1007, [0017]-[0019], [0155]; OKTA-1008, 1-2; OKTA-1009, [0021]-[0025]. For example, randomly selecting bits of the PUF for flipping in Wu-Müelich combination obfuscates the overall repair

time, as an attacker has no way of knowing how many PUF bits were repaired during any given repair operation. *Id.* Further, an adversary monitoring power consumption may have been able to infer how many repairs were performed, but they would be unable to correlate those repairs to a particular PUF bit. *Id.*; OKTA-1003, ¶135.

A POSITA would have also had a reasonable expectation of success in implementing the combination of Wu-Müelich and Yang. OKTA-1003, ¶136. Like Wu and Müelich, Yang is concerned with information security techniques using and relating to PUFs. OKTA-1004, [0010], [0017]-[0018]; OKTA-1005, 1-3; OKTA-1006, [0004], [0010]-[0019]. Further, Yang's techniques are implemented in a memory device including a PUF – structure that is also disclosed by Wu in the form of its PUF Unit 110 that includes non-volatile memory 170. OKTA-1006, [0004], [0010]-[0019]; OKTA-1004, [0015]-[0017], [0028]-[0029]. Finally, Wu envisions that “modifications and alterations” of its techniques can be performed “while retaining the teachings of the invention” (e.g., the modifications of Yang that increase the security of Wu's techniques). OKTA-1004, [0028]. Thus, the modification of Wu-Müelich combination and Yang is nothing more than (i) the application of known techniques (e.g., Yang's random selection of PUF bits) to a known structure (e.g., the PUF repair system of the Wu-Müelich combination) to yield predictable results; (ii) the use of known techniques to improve similar devices

in the same way (e.g., improving the security of a PUF repair system with repair randomization); and (iii) applying a known technique (e.g., Yang's random selection of PUF bits) to a known device ready for improvement to yield predictable results (e.g., implementing repair randomization to mitigate side-channel vulnerabilities). *KSR*, 550 U.S. 398, 415–21; MPEP § 2143; OKTA-1003, ¶136.

2. Analysis

(a) Claim 1

Except as discussed below, Wu-Müelich-Yang renders the limitations of claim 1 obvious in the same manner as explained above for corresponding claim limitations listed in the table below. OKTA-1003, ¶137.

Limitation	Corresponding Limitation
[1.pre]	[6.pre]
[1.1]	[6.1]
[1.2]	[6.2]
[1.4]	[6.4]
[1.5]	[6.5]
[1.6]	[6.6]

[1.3] determining erroneous response bits of the PUF response (S3) comprising replacing one by one, in a random order, each bit of the PUF response by an arbitrary value and determining if the error in the PUF response is s [sic] corrected,

See [6.3], *supra*. The Wu-Müelich-Yang Combination employs Müelich's iterative bitflip procedure at modified step 440 to tentatively **replace**, each bit of the final raw PUF value (**each bit of the PUF response**) with its complement (**arbitrary value** for a binary bit), and then determine whether the error is corrected by evaluating a syndrome weight of the resulting bitstring (**determining if the error in the PUF response is...corrected**). OKTA-1005, 5; OKTA-1003, ¶138.

Further, it would have been obvious for the bit replacement to be performed in **a random order**, consistent with the suggestions in Yang, for the purpose of frustrating side-channel attacks. OKTA-1006, [0014]-[0015], FIG. 1; OKTA-1007, [0017]-[0019], [0155]; OKTA-1008, 1-2; OKTA-1009, [0021]-[0025]. Specifically, Yang discloses a controller that “extract[s] a random bit sequence from [a] random bit pool in [a] PUF unit 105,” which a POSITA would have used with Müelich's Algorithm 1 to determine the order in which bits are swept by the algorithm. OKTA-1006, [0014]-[0015], FIG. 1. In so doing, Müelich's Algorithm 1 sweeps bit positions in a random order (rather than a default positional order discussed for [6.3], *supra*), thereby performing per-bit replacement **in a random order**. *Id.*; OKTA-1003, ¶139.

(b) Claims 2, 3, 5

Wu-Müelich-Yang renders the limitations of claims 2, 3, and 5 obvious in the same manner as explained above for corresponding claim limitations listed in the table below. OKTA-1003, ¶140.

Limitation	Corresponding Limitation
[2.1]	[8.1]
[2.2]	[8.2]
[3]	[9]
[5]	[11]

(c) Claim 12

Claim 12 is substantially similar to claims 1 and 6, reciting a similar series of steps in Beauregard⁶ form with generic preamble language identifying conventional computer components. Accordingly, the Wu-Müelich-Yang Combination renders the limitations of claim 12 obvious in the same manner as explained above for corresponding claim limitations listed in the table below. OKTA-1003, ¶141. In addition, Wu plainly provides the active electronic primitives, PUF circuits,

⁶ Beauregard claims are typically treated as method/process claims. *E.g.*, *Digital-Vending Services Intern., LLC v. University of Phoenix, Inc.*, 672 F.3d 1270, 1275 n.1 (Fed. Cir. 2012).

processor, and memory of [12.pre]. *See, e.g.*, OKTA-1004, [0010], [0015]-[0018], [0020]-[0030], FIG. 1.

Limitation	Corresponding Limitation
[12.pre]	[1.pre], [6.pre]
[12.1]	[1.1], [6.1]
[12.2]	[1.2], [6.2]
[12.3]	[1.3], [6.3]
[12.4]	[1.4], [6.4]
[12.5]	[1.5], [6.5]
[12.6]	[1.6], [6.6]

D. [GROUND 1D] – Claim 4 Is Obvious Over Wu-Müelich-Yang-Tuyls

1. Wu-Müelich-Yang-Tuyls

(a) Wu (OKTA-1004)

See §III.A.1(a), *supra*; OKTA-1003, ¶¶49-53.

(b) Müelich (OKTA-1005)

See §III.A.1(b), *supra*; OKTA-1003, ¶54.

(c) Yang (OKTA-1006)

See §III.C.1(c), *supra*; OKTA-1003, ¶56.

(d) Tuyls (OKTA-1009)

See §III.B.1(c), *supra*; OKTA-1003, ¶55.

(e) Combination of Wu, Muelich, Yang, and Tuyls

A POSITA would have combined both Yang and Tuyls into Wu and Muelich for the reasons discussed above with respect to each individual reference. *See* §§III.B.1(d), III.C.1(d), *supra*; OKTA-1003, ¶142.

2. Analysis

(a) Claim 4

[4.1] The method of claim 1, wherein said device is configured for access to a memory storing a reference hash value of said response bits,

See Ground 1B, [10.1], *supra*; OKTA-1003, ¶143.

[4.2] detecting an error in the PUF response (S2) comprises: generating a hash value by applying a hash function on said bits of the PUF response, comparing said hash value with said stored reference hash value.

See Ground 1B, [10.2], *supra*; OKTA-1003, ¶144.

E. [GROUND 2A] – Claims 6-9, 11, and 14 Are Obvious Over Wu-Muelich-Chan⁷

1. Wu-Muelich-Chan

(a) Wu (OKTA-1004)

See §III.A.1(a), *supra*; OKTA-1003, ¶¶49-53.

⁷ Grounds 2A and 2C apply Chan’s teachings to demonstrate obviousness under a narrow interpretation of the phrase “replace, one by one...each bit of the PUF response” as requiring replacement of all bits of the “PUF response.” As explained in

(b) Muelich (OKTA-1005)

See §III.A.1(b), *supra*; OKTA-1003, ¶54.

(c) Chan (OKTA-1011)

Chan discloses a bit-flipping algorithm with a “taboo” function that “avoids repeated flipping of the same bit” by assigning a “taboo number” to each bit when it is flipped, which only allows subsequent flipping of that bit “if the current iteration number is equal to or less than the taboo number of that variable.” OKTA-1011, 1-3. Chan’s techniques are useful for avoiding “local minima” and result in “improved performance and improved decoding speed” for bit flipping algorithms, “while still maintaining ... simplicity.” *Id.*; OKTA-1003, ¶57.

(d) Combination of Wu, Muelich, and Chan

As discussed above in Ground 1A, a POSITA would have been motivated to apply Muelich’s teachings on iterative bit-flipping error correction to identify and correct erroneous bits in a PUF response (e.g., Wu’s “final raw PUF value”).

§II.C.1 (and applied in Grounds 1A and 1C), the correct ordinary and customary meaning of this phrase is that the process requires replacing individual bit values until such replacement corrects the PUF response error and, thus, reveals an erroneous bit. OKTA-1001, 2:51-52, 3:1-3, 5:1-8; OKTA-1003, §IX.E.

OKTA-1003, ¶145; *see* §III.A.1(c), *supra*. A POSITA would have been motivated to make this combination for multiple reasons. *Id.*

First, Chan teaches that bit-flipping algorithms were vulnerable to converging on spurious local extrema, and a POSITA would have been motivated to introduce taboo numbers to prevent the combination bit-flipping algorithm from converging on an incorrect local solution (and thus wasting computing resources and time). OKTA-1011, 1-3; OKTA-1003, ¶146.

Second, Chan's techniques improve performance of the underlying bit-flipping algorithm while maintaining simplicity (and are thus compatible with PUF environments that may have limited computational resources). *Id.*; OKTA-1003, ¶147.

A POSITA would have also had a reasonable expectation of success in implementing the combination of Chan and the Wu-Müelich combination. OKTA-1003, ¶148. Müelich describes improving error identification and correction using iterative, bit-level correction in a LDPC algorithm – the same category of algorithms described by Chan. OKTA-1005, 4-5; OKTA-1011, 1-3; *see* §III.A.1(b), *supra*. Accordingly, incorporating Chan's taboo numbers into Wu-Müelich's error correction techniques would have been implementable through routine engineering choices without invention, as the Wu-Müelich combination is already structured in

a manner that is explicitly described by Chan. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 415–21 (2007); MPEP § 2143; OKTA-1003, ¶148.

2. Analysis

(a) Claims 6-9, 11, and 14

Except as discussed below, the Wu-Müelich-Chan Combination renders the limitations of claims 6-9, 11, and 14 obvious in the same manner as explained above in Ground 1A. OKTA-1003, ¶149; *see* §III.A.2, *supra*. Ground 2A applies Chan's teachings to demonstrate obviousness under an unduly narrow interpretation of the phrase "replace, one by one...each bit of the PUF response" as requiring replacement of all bits of the "PUF response." As explained in §II.C.1 (and applied in Ground 1A), the correct ordinary and customary meaning of this phrase is that the process requires replacing individual bit values until such replacement corrects the error in the PUF response and, thus, reveals an erroneous bit. OKTA-1001, 2:51-52, 3:1-3, 5:1-8; OKTA-1003, ¶149.

[6.3] determining erroneous response bits of the PUF response (S3) comprising replacing one by one, in a [predetermined] [sic] order, each bit of the PUF response by an arbitrary value and determining if the error in the PUF response is corrected,

See Ground 1A, [6.3], *supra*. The Wu-Müelich-Chan Combination employs Müelich's iterative bitflip procedure at modified step 440 to tentatively ***replace*** each bit of the final raw PUF value (***each bit of the PUF response***) with its complement (***arbitrary value*** for a binary bit), and then determines whether the error is corrected

by evaluating the syndrome weight of the resulting bitstring (*determining if the error in the PUF response is...corrected*). OKTA-1003, ¶150.

For example, Chan discloses a bit-flipping algorithm with a “taboo” function that “avoids repeated flipping of the same bit” by assigning a “taboo number” to each bit when it is flipped, which only allows subsequent flipping of that bit “if the current iteration number is equal to or less than the taboo number of that variable.” OKTA-1011, 1-3. As discussed above, a POSITA would have been motivated to combine Chan with Wu-Müelich to assign taboo numbers to bit positions flipped by Müelich’s iterative bitflip procedure, such that the bit at a given bit position could not be flipped again until Chan’s taboo conditions were satisfied. OKTA-1011, 1-3; OKTA-1003, ¶151; *see* §III.E.1(d), *supra*.

Further, a POSITA would have recognized that, in scenarios involving dense or complex error patterns, use of Chan’s taboo function would result in a process in which *each bit of the PUF response* (i.e., each of the n bit positions of the final raw PUF value) is replaced. OKTA-1003, ¶152; *see* OKTA-1011, 1-3. This is achievable because Chan’s taboo function prevents the bitflip procedure from locally oscillating on a single bit position, thereby forcing it to traverse the sequence of bit positions and address unsatisfied constraints distributed across the final raw PUF value. OKTA-1011, 1-3. While Chan notes that taboos may be ignored if all high-priority candidates are exhausted, a POSITA would recognize that in a high-error

final raw PUF value, valid candidates for flipping (unsatisfied constraints) would exist across the sequence. OKTA-1003, ¶152. In such a “worst-case” convergence scenario, Chan’s taboo function ensures that Muelich’s iterative bitflip procedure iteratively targets and flips these widely distributed bit positions, extending to the entire sequence under certain conditions. *Id.*; see also *Unwired Planet, LLC v. Google Inc.*, 841 F.3d 995, 1002 (Fed. Cir. 2016) (“[P]rior art that sometimes meet the claim elements are sufficient to show obviousness.”).

F. [GROUND 2B] – Claim 10 Is Obvious Over Wu-Muelich-Chan-Tuyls

1. Wu-Muelich-Chan-Tuyls

(a) Wu (OKTA-1004)

See §III.A.1(a), *supra*; OKTA-1003, ¶¶49-53.

(b) Muelich (OKTA-1005)

See §III.A.1(b), *supra*; OKTA-1003, ¶54.

(c) Chan (OKTA-1011)

See §III.E.1(c), *supra*; OKTA-1003, ¶57.

(d) Tuyls (OKTA-1009)

See §III.B.1(c), *supra*; OKTA-1003, ¶55.

(e) Combination of Wu, Muelich, Chan, and Tuyls

A POSITA would have combined both Chan and Tuyls into Wu and Muelich for the reasons discussed above with respect to each individual reference. *See* §§III.B.1(d), III.E.1(d), *supra*; OKTA-1003, ¶153.

2. Analysis

(a) Claim 10

[10.1] The method of claim 6, wherein said device is configured for access to a memory storing a reference hash value of said response bits,

See Ground 1B, [10.1], *supra*; OKTA-1003, ¶154.

[10.2] detecting an error in the PUF response (S2) comprises: generating a hash value by applying a hash function on said bits of the PUF response, comparing said hash value with said stored reference hash value.

See Ground 1B, [10.2], *supra*; OKTA-1003, ¶155.

G. [GROUND 2C] – Claims 1-3, 5, 12, and 13 Are Obvious Over Wu-Muelich-Chan-Sundararajan⁸

1. Wu-Muelich-Chan-Sundararajan

(a) Wu (OKTA-1004)

See §III.A.1(a), *supra*; OKTA-1003, ¶¶49-53.

⁸ Ground 2C applies Sundararajan’s teachings as an alternative basis to demonstrate obviousness with respect to the claimed phrase “random order” within Elements [1.3] and [12.3].

(b) Muelich (OKTA-1005)

See §III.A.1(b), *supra*; OKTA-1003, ¶54.

(c) Chan (OKTA-1011)

See §III.E.1(d), *supra*; OKTA-1003, ¶57.

(d) Sundararajan (OKTA-1010)

Sundararajan describes a modified “Gradient Descent Bit Flipping (GDBF) algorithm” for decoding LDPC bit-flipping codes. OKTA-1010, 1. Sundararajan’s “Noisy GDBF” introduces a “random perturbation” (vector “z” of “identically distributed Gaussian random variables”) that randomizes bit-flipping decisions made by the algorithm. OKTA-1010, 1-4. Introducing this randomness “offers a low-complexity solution to escape spurious local maxima” that a bit-flipping algorithm would otherwise erroneously converge upon. *Id.*; OKTA-1003, ¶58.

(e) Combination of Wu, Muelich, Chan, and Sundararajan

As discussed above in Ground 1A, a POSITA would have been motivated to apply Muelich’s teachings on iterative bit-flipping error correction to identify and correct erroneous bits in a PUF response (e.g., Wu’s “final raw PUF value”). OKTA-1003, ¶156; *see* §III.A.1(c), *supra*. Additionally, a POSITA would have been motivated to combine Sundararajan with Wu-Muelich-Chan to apply a random perturbation to the Wu-Muelich-Chan bit-flipping algorithm to achieve a benefit described in Sundararajan, e.g., by improving performance of bit-flipping algorithms

for “low-complexity” applications. OKTA-1010, 1-4; OKTA-1003, ¶156. A POSITA would have been further motivated to make this combination for multiple reasons. *Id.*

Like Chan, Sundararajan also teaches that bit-flipping algorithms were vulnerable to converging on spurious local maxima, and a POSITA additionally would have been motivated to introduce randomness into the combination’s iterative bit-flipping to prevent the algorithm from converging on an incorrect local solution (and thus wasting computing resources and time). OKTA-1010, 1-4. Further, Sundararajan’s techniques improve performance of the underlying bit-flipping algorithm and are described as “low-complexity” (and thus compatible with PUF environments that may have limited computational resources). *Id.*; OKTA-1003, ¶157.

A POSITA would have also had a reasonable expectation of success in implementing the combination of Sundararajan and the Wu-Müelich-Chan Combination. OKTA-1003, ¶158. Müelich describes improving error identification and correction using iterative, bit-level correction in a LDPC algorithm – the same category of algorithms described by Sundararajan. OKTA-1005, 4-5; OKTA-1010, 1-4; *see* §III.A.1(b), *supra*. Accordingly, incorporating Sundararajan’s random perturbations into Wu-Müelich-Chan’s error correction techniques would have been implementable through routine engineering choices without invention, as the Wu-

Müelich-Chan combination is already structured in a manner that is explicitly described by Sundararajan. *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 415–21 (2007); MPEP § 2143; OKTA-1003, ¶158.

2. Analysis

(a) Claims 1-3, 5, 12, and 13

Except as discussed below, Wu-Müelich-Chan-Sundararajan renders the limitations of claims 1-3, 5, 12, and 13 obvious in the same manner as explained above in Ground 1C with respect to Wu-Müelich. OKTA-1003, ¶159; *see* §III.C.2, *supra*. Ground 2C relies on Sundararajan with respect to the “random order” limitation. Moreover, as in Ground 2A, Ground 2C applies Chan’s teachings to demonstrate obviousness under an unduly narrow interpretation of the phrase “replace, one by one...each bit of the PUF response” as requiring replacement of all bits of the “PUF response.” As explained in §II.C.1 (and applied in Ground 1C), the correct ordinary and customary meaning of this phrase is that the process requires replacing individual bit values until such replacement corrects the PUF response error and, thus, reveals an erroneous bit. OKTA-1001, 2:51-52, 3:1-3, 5:1-8; OKTA-1003, ¶159.

[1.3] determining erroneous response bits of the PUF response (S3) comprising replacing one by one, in a random order, each bit of the PUF response by an arbitrary value and determining if the error in the PUF response is [] corrected,

See Grounds 1A, 2A, [6.3], *supra*. The Wu-Müelich-Chan-Sundararajan combination uses Müelich's iterative bitflip procedure at modified step 440 to tentatively ***replace*** each bit of the final raw PUF value (***each bit of the PUF response***) with its complement (***arbitrary value*** for a binary bit), and then determines whether the error is corrected by evaluating the syndrome weight of the resulting bitstring (***determining if the error in the PUF response is...corrected***). OKTA-1003, ¶160. Further, it would have been obvious for the bit replacement to be performed in ***a random order***, consistent with the suggestions in Sundararajan, for the purpose of improving efficiency and avoiding local extrema. OKTA-1010, 1-4; OKTA-1003, ¶160.

For example, Sundararajan's "Noisy GDBF" introduces a "random perturbation" (vector "z" of "identically distributed Gaussian random variables") that randomizes bit-flipping decisions made by the bitflip procedure. *Id.* As discussed above, a POSITA would have found it obvious that, in the context of the Wu-Müelich-Chan iterative bitflip procedure, tentative replacement using such a random perturbation involves tentative replacement of bit positions of the final raw PUF value ***in a random order***. OKTA-1010, 1-4; OKTA-1003, ¶161. Consistent with Sundararajan, a POSITA would have understood that a well-known

configuration involves applying a random perturbation to the Wu-Müelich-Chan iterative bitflip procedure, such that the selection of a bit position to flip is influenced by a random perturbation, thereby introducing a stochastic element to the gradient descent process. OKTA-1010, 1-4; OKTA-1003, ¶161; *see* §III.G.1(e), *supra*.

A POSITA would have found it obvious that, in the context of the Wu-Müelich-Chan iterative bitflip procedure, tentative replacement of bit positions using such a random perturbation (as described in Sundararajan) involves, in some circumstances, a randomized sequence of bit position replacements (*random order*). OKTA-1010, 1-4; OKTA-1003, ¶162. Specifically, the introduction of random noise to the selection metric renders the sequence in which bit positions are replaced non-deterministic, which would have been understood by a POSITA to represent replacement of bit positions in a *random order* to avoid local minima. *Id.* Because a random perturbation is added to the decision metric for every bit position in every iteration, the relative ranking of candidate bit positions is shuffled by the random noise values, causing bit positions to be selected in a sequence that differs from a deterministic gradient. This stochastic reordering ensures that the specific sequence of bit replacements is not fixed by input data and follows a random trajectory through the search space. *Id.*

H. [GROUND 2D] – Claim 4 Is Obvious Over Wu-Müelich-Chan-Sundararajan-Tuyls

1. Wu-Müelich-Chan-Sundararajan-Tuyls

(a) Wu (OKTA-1004)

See §III.A.1(a), *supra*; OKTA-1003, ¶¶49-53.

(b) Müelich (OKTA-1005)

See §III.A.1(b), *supra*; OKTA-1003, ¶54.

(c) Chan (OKTA-1011)

See §III.E.1(d), *supra*; OKTA-1003, ¶57.

(d) Sundararajan (OKTA-1010)

See §III.G.1(d), *supra*; OKTA-1003, ¶58.

(e) Tuyls (OKTA-1009)

See §III.B.1(c), *supra*; OKTA-1003, ¶55.

(f) Combination of Wu, Müelich, Chan, Sundararajan, and Tuyls

A POSITA would have combined Chan, Sundararajan, and Tuyls into Wu and Müelich for the reasons discussed above with respect to each individual reference.

See §§III.B.1(d), III.E.1(d), III.G.1(e), *supra*; OKTA-1003, ¶163.

2. Analysis

(a) Claim 4

[4.1] The method of claim 1, wherein said device is configured for access to a memory storing a reference hash value of said response bits,

See Ground 1B, [10.1], *supra*; OKTA-1003, ¶164.

[4.2] detecting an error in the PUF response (S2) comprises: generating a hash value by applying a hash function on said bits of the PUF response, comparing said hash value with said stored reference hash value.

See Ground 1B, [10.2], *supra*; OKTA-1003, ¶165.

I. [GROUND 3] – If Narrowly Interpreted to Avoid the Prior Art, Claims 1-14 Lack Written Description Support

1. Background

Independent claims 1, 6, 12, and 14 each require “determining erroneous response bits of the PUF response” by “replacing one by one...each bit of the PUF response by an arbitrary value and determining if the error in the PUF response is corrected.” As discussed, under the ordinary and customary meaning in view of the specification, these limitations do ***not*** require, in a single error-detection pass, [i] determining multiple erroneous response bits or [ii] replacing the value of each and every single bit of the PUF response. *See* §II.C.1, *supra*; OKTA-1003, ¶168; OKTA-1001, 2:51-52, 3:1-3, 5:1-8, 6:29-36, Fig. 4. Thus, the prior art combinations in Grounds 1A–2D render these limitations obvious through a PUF repair architecture that improves Wu’s evaluation-time workflow by implementing error correction using an iterative bit-flipping technique that evaluates candidate single-bit replacements. *See* §§III.A–D, *supra*.

What’s more, adopting any alternative interpretation narrow enough to distinguish these limitations from the prior art would render the Challenged Claims unpatentable based on a lack of written description support. As established below,

the '359 patent's specification does not demonstrate possession of a PUF response repair technique involving either an exhaustive replacement of each and every bit or the simultaneous discovery of multiple erroneous bits using a single bit-by-bit replacement scheme. OKTA-1003, ¶169.

2. Legal Standard

To meet the written description requirements of 35 U.S.C. §112(a), the specification “must describe the invention sufficiently to convey to a person of skill in the art that the patentee had possession of the claimed invention at the time of the application, i.e., that the patentee invented what is claimed.” *LizardTech, Inc. v. Earth Res. Mapping, Inc.*, 424 F.3d 1336, 1345 (Fed. Cir. 2005). “This requires that the written description actually or inherently disclose the claim element.” *PowerOasis, Inc. v. T-Mobile USA, Inc.*, 522 F.3d 1299, 1306 (Fed. Cir. 2008); OKTA-1003, ¶166.

“The purpose of the written description requirement is to ensure the scope of the right to exclude, as set forth in the claims, does not overreach the scope of the inventor's contribution to the field of art as described in the patent specification.” *ICU Med., Inc. v. Alaris Med. Sys, Inc.*, 558 F.3d 1368, 1376 (Fed. Cir. 2009) (citations omitted). Thus, “[t]he question is not whether a claimed invention is an obvious variant of that which is disclosed in the specification. Rather, a prior application itself must describe an invention, and do so in sufficient detail that one

skilled in the art can clearly conclude that the inventor invented the claimed invention as of the filing date sought.” *Lockwood v. Am. Airlines, Inc.*, 107 F.3d 1565, 1572 (Fed. Cir. 1997); *see also PowerOasis*, 522 F.3d at 1306; *Transperfect Glob., Inc. v. Matal*, 703 F. App’x 953, 963 (Fed. Cir. 2017); *Rivera v. Int’l Trade Comm’n*, 857 F.3d 1315, 1320 (Fed. Cir. 2017). And likewise, a proper written description analysis “must” take each claim “as an integrated whole rather than as a collection of independent limitations.” *Novozymes A/S v. Dupont Nutrition Biosciences APS*, 723 F.3d 1336, 1349 (Fed. Cir. 2013). As such, written description support for a claim cannot be derived “from an amalgam of disclosures plucked selectively from the [underlying] application.” *Id.*; *see also Purdue Pharma L.P. v. Recro Tech., LLC*, 694 F. App’x 794, 797 (Fed. Cir. 2017) (“To the extent that Purdue contends that a person of skill in the art would isolate and combine aspects from various embodiments in the specifications...to obtain the claimed invention, Purdue relies upon the wrong test.”); OKTA-1003, ¶167.

3. Lack of Written Description Support

(a) Determining multiple erroneous response bits in a single error-detection pass

As discussed at §II.C.1, *supra*, the ’359 patent’s specification juxtaposes two alternative bit replacement schemes. First, the specification explains that “the PUF circuits may *replace one by one each bit* of the PUF response by an arbitrary value

and determine if the error in the PUF response is corrected by comparing the hash of the modified PUF response with the reference hash value.” OKTA-1001, 5:1-5. Next, addressing a distinct hypothetical where “multiple bits are erroneous in the PUF response,” the specification says that “*multiple bits may be replaced by their complement at the same time* in order to determine all of the erroneous bits of the PUF response.” *Id.*, 5:5-8; OKTA-1003, ¶170.

In short, the inventors possessed two distinct and alternative bit replacement schemes, one that replaces individual bits to discover individual bit errors and another that replaces multiple bits simultaneously to discover multiple bit errors at the same time. OKTA-1003, ¶171. There is no explicit, implicit, or inherent disclosure that the disclosed singular bit-by-bit replacement scheme—the one the claims recite—can simultaneously discover multiple bit errors in the PUF response. OKTA-1003, ¶171. Thus, to the extent Patent Owner insists on narrowing the Challenged Claims in this fashion to avoid the prior art, the Challenged Claims must be invalid for lack of written description support. Mixing and matching isolated disclosures in a manner not contemplated by the specification cannot, as a matter of law, show written description support. *Cf. Novozymes*, 723 F.3d at 1349; *Purdue Pharma*, 694 F. App’x at 797.

(b) Replacing each and every single bit of the PUF response in a single error-detection pass

As discussed at §II.C.1, *supra*, the '359 patent's specification fails to disclose replacing the value of each and every single bit of the PUF response in a single error-detection pass. *See, e.g.*, OKTA-1001, 6:29-36, FIG. 4; OKTA-1003, ¶172. To the contrary, the specification makes clear that the disclosed error-detection process concludes—i.e., makes no further bit replacements—when the erroneous bit or bits are identified and matched with a corresponding additional output bit. *Id.* Indeed, an exhaustive bit-flipping sweep is inconsistent with the '359 patent's stated goals to “*minimize the computations* required to find the erroneous bits of the PUF response” and to “*reduc[e] the number* of response bits to be replaced.” OKTA-1001, 2:51-52, 3:1-3. What's more, even if replacing each and every single bit were an obvious variant of the '359 patent's disclosure (which it is not), that still would not be enough for written description support. *Cf. Lockwood*, 107 F.3d at 1572.

J. [GROUND 4] – If Narrowly Interpreted to Avoid the Prior Art, Claims 1-14 Are Indefinite

1. Background

Independent claims 1, 6, 12, and 14 each require “replacing one by one...each bit of the PUF response by an arbitrary value and determining if the error...is corrected.” As discussed, under the ordinary and customary meaning in view of the specification, these limitations do *not* require, in a single error-detection pass,

replacing the value of each and every single bit of the PUF response. *See* §II.C.1, *supra*; OKTA-1003, ¶174; OKTA-1001, 2:51-52, 3:1-3, 5:1-8, 6:29-36, FIG. 4. Thus, the prior art combinations in Grounds 1A–2D render this limitation obvious through a PUF repair architecture that improves Wu’s evaluation-time workflow by implementing error correction using Muelich’s iterative bitflip procedure that evaluates candidate single-bit replacements. *See* §§III.A–D, *supra*. What’s more, any alleged alternative interpretation narrow enough to distinguish this limitation from the prior art renders the Challenged Claims unpatentable due to indefiniteness. Specifically, because the specification supports Petitioner’s proposed interpretation at least as strongly as any narrower alternative, a skilled artisan is left with no objective basis to confidently choose between the competing definitions.

2. Legal Standard

A claim is indefinite if its language, read in light of the specification and prosecution history, “fail[s] to inform, with reasonable certainty, those skilled in the art about the scope of the invention.” *Nautilus, Inc. v. Biosig Instruments, Inc.*, 572 U.S. 898, 901 (2014). Claim scope is not reasonably certain, for example, when the “language ‘might mean several different things and no informed and confident choice is available among the contending definitions.’” *Media Rights Techs., Inc. v. Capital One Fin. Corp.*, 800 F.3d 1366, 1371 (Fed. Cir. 2015) (quoting *Nautilus*, 572 U.S., at 911 n.8); *accord HZNP Meds. LLC v. Actavis Labs. UT, Inc.*, 940 F.3d

680, 698 (Fed. Cir. 2019) (affirming a district court’s indefiniteness conclusion regarding the property “better drying time,” where the two tests disclosed in the specification to evaluate this parameter produced inconsistent results); *Akamai Techs., Inc. v. MediaPointe, Inc.*, 159 F.4th 1370, 1377-79 (Fed. Cir. 2025) (affirming a district court’s indefiniteness conclusion regarding the term “optimal routes,” where the specification identified different measures for determining an optimal route but “no guidance about which measure, or what combinations of measures, to rely on when they diverge”) OKTA-1003, ¶173.

3. Indefiniteness

The specification explicitly defines the inventors’ purpose as “minimiz[ing] the computations required to find the erroneous bits” and “reducing the number of response bits to be replaced.” OKTA-1001, 2:51-52, 3:1-3. Consistent with Petitioner’s interpretation, this expressly stated purpose of promoting efficiency is served by an iterative process that stops when the erroneous PUF response bit is found. OKTA-1003, ¶175.

By contrast, any alternative interpretation that requires indiscriminately flipping each and every single bit of the PUF response demands a process that is deliberately inefficient. It requires the system to continue testing every remaining bit even after the error has been resolved. To adopt this narrow view, the Board would have to conclude that the specification’s core objective of efficiency does not

apply to the claims. Narrowly construing the Challenged Claims to require flipping each and every single bit of the PUF response would also require dismissing the '359 patent's preferred embodiment. As discussed, the only provided flowchart and associated description make clear that the error-detection process concludes when the erroneous bit is identified and matched with a corresponding additional output bit. *See* §II.C.1, *supra* (discussing OKTA-1001, 6:29-36, FIG. 4); OKTA-1003, ¶176.

If the specification's guidance is dismissed in this fashion, a POSITA is left with only the raw claim text. In this interpretive vacuum, the phrase "replacing... each bit" is susceptible to two mutually exclusive interpretations, with no objective basis to choose between them:

- **Petitioner's Interpretation:** Replace each bit in the PUF response until the error in the PUF response is corrected.
- **Alternative Narrower Interpretation:** Replace each and every single bit in the PUF response, regardless of when the PUF response error is corrected.

Without the specification's context, "no informed and confident choice is available among the contending definitions," and the Challenged Claims are indefinite. *Media Rights Techs.*, 800 F.3d at 1371 (quotes omitted); OKTA-1003, ¶177.

IV. DISCRETIONARY DENIAL IS NOT APPROPRIATE

Petitioner believes that discretionary denial is unwarranted, and yet, Petitioner intends to utilize the bifurcated briefing process contemplated by the March 26, 2025, Stewart Memorandum to rebut contentions if offered by Patent Owner to the contrary. OKTA-1014.

V. CONCLUSION

The Challenged Claims are unpatentable. Petitioner authorizes charge of fees to Deposit Account No. 06-1050.

VI. PAYMENT OF FEES—37 C.F.R. §42.103

Petitioner authorizes the Patent and Trademark Office to charge any fees to Deposit Account No. 06-1050.

VII. MANDATORY NOTICES

A. Real Party-In-Interest Under 37 C.F.R. § 42.8(b)(1)

Petitioner, Okta, Inc., and Okta GmbH are the real parties-in-interest.

B. Related Matters Under 37 C.F.R. § 42.8(b)(2)

Petitioner is not aware of any disclaimers, reexamination certificates or petitions for *inter partes* review for the '359 patent. Petitioner is not aware of any civil actions concerning the '359 patent.

C. Lead And Back-Up Counsel Under 37 C.F.R. § 42.8(b)(3)

Petitioner provides the following designation of counsel.

Lead Counsel	Backup Counsel
Kenneth Darby, Reg. No. 65,068 Fish & Richardson P.C. 60 South Sixth Street Suite 3200 Minneapolis, MN 55402 Tel: 202-783-5070 Fax: 877-769-7945 Email: PGR56407-0020IP1@fr.com	Ryan Chowdhury, Reg. No. 74,466 Chris A. von Gunten, Reg. No. 83,726 Fish & Richardson P.C. 60 South Sixth Street Suite 3200 Minneapolis, MN 55402 T: 202-783-5070 / F: 877-769-7945 Email: PGR56407-0020IP1@fr.com

D. Service Information

Please address all correspondence and service to the address listed above.

Petitioner consents to electronic service by email at PGR56407-0020IP1@fr.com
(referencing No. 56407-0020IP1).

Respectfully submitted,

Dated May 7, 2026

/Ryan Chowdhury/
Kenneth Darby, Reg. No. 65,068
Ryan Chowdhury, Reg. No. 74,466
Chris A. von Gunten, Reg. No. 83,726
Fish & Richardson P.C.
60 South Sixth Street
Minneapolis, MN 55402
T: 202-783-5070
F: 877-769-7945

(Control No. PGR2026-00045)

Attorneys for Petitioner

CERTIFICATION UNDER 37 CFR § 42.24

Under the provisions of 37 CFR § 42.24(d), the undersigned hereby certifies that the word count for the foregoing Petition for Post Grant Review totals 17,399 words, which is less than the 18,700 allowed under 37 CFR § 42.24.

Dated May 7, 2026

/Ryan Chowdhury/
Kenneth Darby, Reg. No. 65,068
Ryan Chowdhury, Reg. No. 74,466
Chris A. von Gunten, Reg. No. 83,726
Fish & Richardson P.C.
60 South Sixth Street, Suite 3200
Minneapolis, MN 55402
T: 202-783-5070
F: 877-769-7945

Attorneys for Petitioner

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. §§ 42.6(e)(4)(i) *et seq.* and 42.105(b), the undersigned certifies that on May 7, 2026, a complete and entire copy of this Petition for Post Grant Review, Power of Attorney, and all supporting exhibits were provided via Federal Express, to the Patent Owner by serving the correspondence address of record as follows:

Thales Dis Cpl USA Inc.
Intellectual Property
900 South Pine Island Road, Suite 710
Plantation, Florida 33324
UNITED STATES

/Esther Goldschlager/
Esther Goldschlager
Fish & Richardson P.C.
60 South Sixth Street, Suite 3200
Minneapolis, MN 55402
goldschlager@fr.com