(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 September 2016 (09.09.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/141383 A1

(51) International Patent Classification:
G06F 11/30 (2006.01)

(21) International Application Number:
PCT/US2016/021264

(22) International Filing Date:
7 March 2016 (07.03.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

62/128,920	5 March 2015 (05.03.2015)	US
62/150,254	20 April 2015 (20.04.2015)	US
62/150,586	21 April 2015 (21.04.2015)	US
14/704,947	5 May 2015 (05.05.2015)	US
14/746,090	22 June 2015 (22.06.2015)	US

(71) Applicant: **SYPRIS ELECTRONICS, LLC** [US/US]; 10901 N. McKinley Drive, Tampa, Florida 33612 (US).

(72) Inventor: **WALLRABENSTEIN, John Ross**; 1281 Win Hentschel Blvd., West Lafayette, Indiana 47906 (US).

(74) Agent: **BRINDISI, Thomas**; Law Office of Thomas J. Brindisi, 1613 Sunshine Drive, Key West, Florida 33040 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

— *with amended claims (Art. 19(1))*

(54) Title: SYSTEM AND DEVICE FOR VERIFYING THE INTEGRITY OF A SYSTEM FROM ITS COMPONENTS

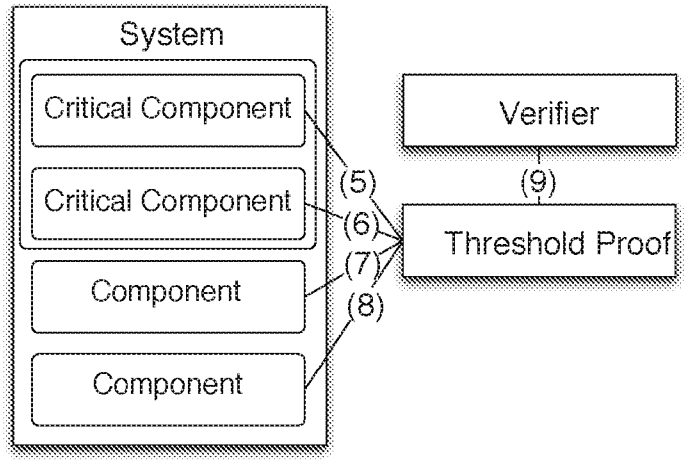


Fig. 2

(57) Abstract: A system and device for verifying the integrity of a system from its components, the system comprising a plurality of components each having a physical state, the system and the device comprising a processor that is connected to each of the components, the processor configured to verify systemic integrity by performing verification on some or all specified components. The verification may be individual (1, 1) or threshold (n, 1), and may be interactive or non-interactive.

WO 2016/141383 A1

SYSTEM AND DEVICE FOR VERIFYING THE INTEGRITY OF A SYSTEM FROM ITS COMPONENTS

FIELD OF THE INVENTION

[0001] The present invention relates to integrity verification of systems comprising electronic components.

REFERENCE TO RELATED APPLICATIONS

[0002] This international application claims the benefit of the priority of non-provisional U.S. Patent Applications S.N. 14/704,947 ("the '947 application") filed May 5, 2015 and S.N. 14/746,090 filed June 22, 2015 (which was a continuation-in-part of the '947 application), and of U.S. Provisional Patent Applications S.N. 62/150,254 filed April 20, 2015, S.N. 62/128,920 filed March 5, 2015 ("the '920 application"), and S.N. 62/150,586 filed April 21, 2015 ("the '586 application"). The contents of the '920 and '586 applications and of U.S. Provisional Patent Application S.N. 61/988,848 filed May 5, 2014 (published in the record of prosecution of U.S. Patent Application Publication No. 20150317480) are also incorporated herein by reference.

BACKGROUND OF THE INVENTION

[0003] In many applications, it can be useful to employ means for verifying the integrity of a system by interrogating the components it is composed of. For example, a weapon system may require components to be internally validated during a boot process, or a vehicle may validate critical electronic control units on startup. Prior

art typically accomplishes the verification of a component through a demonstration that it possesses a secret value, for example, through a zero knowledge proof of knowledge. This method of verification, however, may be associated with one or more constraints relating to hardware integrity or the security of private information. As to hardware integrity, existing component authentication protocols only verify that an entity possesses a private value, and typically just infer hardware integrity if the device has a physical construction designed to deter tampering (e.g., a hardware security module). Even with a tamper resistant physical construction, the integrity of the physical construction is not inextricably linked to the integrity of the device itself. As to the security of private information, existing component authentication protocols require that the component store and protect private information (typically a private key for cryptographic authentication protocols). If the private information is compromised, it may be possible for an adversary to masquerade as a valid component in the larger system.

[0004] Asim et al. ("Physical Unclonable Functions and Their Applications to Vehicle System Security," Vehicular Technology Conference, VTC Spring 2009, IEEE 69th) discusses using PUFs in vehicle components as a method for regenerating private keys, which is a well-known application. However, they fail to give an enabling construction allowing a system-wide identity to be constructed from each of the individual components.

[0005] Rigaud (editor) in "D3.1 Report on Protocol choice and implementation," Holistic Approaches for Integrity of ICT-Systems (2014) describes applying PUFs to chips as a method for authenticating a chip (the device-under-test) to the testing equipment, which could detect fake chips. However, there is no construction that would enable a system-wide identity to be constructed from each of the individual

chips.

[0006] Ibrahim et al. ("Cyber-physical security using system-level pufs," Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th Int'l, IEEE) discusses the general concept of combining PUFs from distinct system components to form a combined identity, but they fail to give an enabling construction. In their concluding remarks, the authors specifically state that they lack a realized solution.

[0007] Peeters ("Security Architecture for Things That Think," Diss. Ph. D. thesis, KU Leuven, June 2012) describes using a PUF in resource-constrained devices for regenerating a share from an external threshold system composed of a user's devices. The PUF is applied solely as a storage mechanism, eliminating the need to store the share in plaintext on the device. However, no internal threshold application is given, nor is the challenge-helper pair ever refreshed.

[0008] Krzywiecki et al. ("Coalition resistant anonymous broadcast encryption scheme based on PUF," Trust and Trustworthy Computing. Springer Berlin Heidelberg, 2011, 48-62) describe a broadcast encryption scheme where subscribers must invoke a PUF-enabled card to regenerate shares of a threshold system. The construction requires an incorruptible distributor to store and protect raw PUF output. The system is designed to allow an end device to recover a symmetric key only if it has not been revoked by the broadcaster. The PUF-enabled receiving device must construct the full symmetric key from its shares in order to decrypt the incoming transmission. No internal threshold application is given, nor is the challenge-helper pair ever refreshed.

[0009] Khoshroo et al. ("Design and Evaluation of FPGA-based Hybrid Physically Unclonable Functions," Diss. Western University London, 2013) describe a modified

secret sharing scheme, where each player's share is a challenge-helper pair generated from the dealer's PUF. The actual shares for the threshold system are recovered only given both the challenge-helper pair and access to the PUF, which regenerates the share from the challenge-helper pair. As each share is worthless without access to the PUF, an adversary can compromise all of the end devices, and yet is unable to recover the secret without access to the PUF. No cryptographic operations are possible over these pseudo-shares. The shared secret may only be recovered if all of the shares are regenerated, and the dealer is assumed to be incorruptible. The dealer's PUF is used only as a method for obfuscating the shares that are distributed to players.

SUMMARY OF THE INVENTION

[0010] Various embodiments of the invention provide for the verification of a set of components of an electronic system, such that the integrity of the system as a whole is deduced therefrom. One embodiment of the invention employs physical unclonable functions (PUFs) for detecting hardware tampering in integrated circuits, and zero knowledge proof protocols for authentication. In one embodiment this is done by individual verification of components; in another embodiment, relevant components may be verified together, with each generating a local proof of validity and collaborating to combine their local proofs into a single proof that validates the integrity of the system as a whole.

[0011] In another embodiment, which may be provided individually or in combination with one or more of the foregoing embodiments, systemic trust may be established even if the system's components themselves are untrusted, by employing a hardware root-of-trust that iteratively extends the trust boundary as each compo-

nent is verified.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] **Fig. 1** is a system diagram illustrating $(1, 1)$ integrity verification of components;

[0013] **Fig. 2** is a system diagram illustrating $(n, 1)$ integrity verification of components; and

[0014] **Fig. 3** illustrates a system establishing trust through layered security derived from a high assurance processor.

DETAILED DESCRIPTION OF EMBODIMENTS

[0015] In one embodiment, each of a system's n relevant components may be interrogated (e.g., sequentially) through an interactive or non-interactive zero-knowledge proof of knowledge. Authentication algorithms such as those disclosed in the '848 and '586 applications (elliptic curve-based) or in U.S. Patent No. 8,918,647 (discrete log-based; "the '647 patent," which is incorporated here by reference), for example, may be used to establish the hardware integrity of components having trusted means of gathering private information, such as physical unclonable functions. A PUF links the evaluation of a function with the hardware on which it is executed, such that any adversarial tampering of the hardware affects the evaluation of the function. By further linking the PUF output with the construction of the zero knowledge proof, the hardware integrity of the device can be deduced by an external verifier from its ability

to successfully complete the zero knowledge proof protocol. The PUF may also be configured to dynamically generate private information from only public information, so that components need not store and protect private information. In another embodiment, integrity of the system may be established through a single *collaborative* response from all (or a subset of) the components by constructing a *threshold* proof that requires all or some subset of the n components to be functioning correctly. In that case, rather than construct a separate proof for each of the n components, they collaboratively construct a single proof that establishes the validity of all or a subset of the n components simultaneously.

[0016] One embodiment of a component may comprise a Xilinx Artix 7 field programmable gate array (FPGA) platform, equipped, e.g., with 215,000 logic cells, 13 Megabytes of block random access memory, and 700 digital signal processing (DSP) slices. In an embodiment employing elliptic curve cryptography, for example, the hardware mathematics engine may be instantiated in the on-board DSP slices, with the PUF construction positioned within the logic cells, and a logical processing core including an input and output to the PUF and constructed to control those and the component's external input and output and to perform algorithms (sending elliptic curve and other mathematical calculations to the math engine) such as those described above. A verifier may comprise Xilinx Artix 7 FPGA as described above, or server computer having an 8-core 3.1GHz processor and 16GB of RAM, or another suitable means, connected, e.g., by wire or wirelessly to the system's components.

Component Authentication

[0017] In the individual interrogation method of verification, or "(1, 1) verification," each component interacts directly with the verifier $\mathcal{V}$. In interactive (1, 1) verification,

the verifier $\mathcal{V}$ issues a nonce as part of a two message protocol with each component. In non-interactive $(1, 1)$ verification, each component sends only a single message to the verifier $\mathcal{V}$, and includes a value equivalent to a nonce (e.g., a timestamp) that cannot be manipulated by the component. In the collaborative method of verification, or "$(n, 1)$ verification," a subset of the n components collaboratively generate a single joint proof, which convinces the verifier $\mathcal{V}$ of the integrity of the subset of n components. In interactive $(n, 1)$ verification, the verifier $\mathcal{V}$ issues a nonce as part of a two message protocol, where a subset of components act jointly and send only a single response. In non-interactive $(n, 1)$ verification, a subset of components send only a single message to the verifier $\mathcal{V}$, which includes a value equivalent to a nonce (e.g., a timestamp) that cannot be manipulated by any subset of the components.

[0018] For the purposes of providing a detailed description of an embodiment, the example of an elliptic curve-based construction is utilized, with E denoting an elliptic curve defined over a finite field $\mathbb{F}_p$ where G is a base point of order q. One of ordinary skill will recognize that the invention (be it $(1, 1)$, $(n, 1)$, and/or layered security) can be readily implemented using various other constructions (with just one example alternative being the '647 patent's discrete logarithm construction). Thus the invention is not limited to any particular construction, except where specifically stated in the claims.

[0019] A zero knowledge authentication protocol typically requires a unique and random *nonce* to be issued by the verifier $\mathcal{V}$ during each protocol invocation. The nonce prevents the proof from the verifier from being reused in the future (i.e., a replay attack), and the proving component must not be able to influence its value. For example, the '848 application discloses a derived token-based zero knowledge proof protocol, the teachings regarding which are incorporated here by reference,

summarized as follows:

Interactive Authentication Algorithm for an Individual Device

---

for Server s **do**

Device $d \leftarrow \{c, E, G, p, q, P, N\}$ where N is a nonce and P is the helper string

for PUF Device d **do**

$x \leftarrow Hash(c, E, G, p, q)$

$p_i^{\text{priv}} \leftarrow D(PUF(x) \oplus P)$ where $PUF(\cdot)$ is the PUF function and D is an error decoding scheme

$r \leftarrow$ random $\in \mathbb{F}_q$, a random group element

$B \leftarrow r \cdot G \mod p$

$c' \leftarrow \text{Hash}(G, B, A, N)$, (a hash, not a challenge)

$m \leftarrow r + c' \cdot p_i^{\text{priv}} \mod q$

Server $s \leftarrow \{B, m\}$

for Server s **do**

$c' \leftarrow \text{Hash}(G, B, A, N)$

$D \leftarrow m \cdot G - c' \cdot A \mod p$

$$\text{Device } d \leftarrow \begin{cases} \text{accept} & : D = B \\ \text{deny} & : D \neq B \end{cases}$$

This algorithm proceeds as follows:

- Prior to authentication, the server has issued a random challenge variable c to the device, which is used to form a PUF challenge input x. The enrollment server and device agree on an elliptic curve E defined over a finite field $\mathbb{F}_p$ where G is a base point of order q. The device d_i returns a public commitment $A = p_i^{\text{priv}} \cdot G$ to the server, which links its PUF to the challenge variable c (on which the challenge input x depends), and a public helper value P that will correct the noisy PUF output.

- When the server wishes to authenticate the device, it issues an authentication request and the tuple $\{c, E, G, p, q, P, N\}$ is sent to the device.

- The device constructs the PUF challenge input $x \leftarrow H(c, E, G, p, q)$, which links the challenge variable c with the public parameters of the elliptic curve, and passes it to the PUF yielding output O', which is $\oplus$'d with helper value P and the result decoded using an error decoding scheme D.

- As the PUF output is noisy, when queried on challenge x again in the future, the new output O' may not exactly match the previous output value O. However, it is assumed that O and O' will be t-close with respect to some distance metric (e.g. Hamming distance). Thus, an error correcting code may be applied to the PUF output such that at most t errors will still recover O. During enrollment, error correction was applied over the random group element p_i^{priv}, and then this value was blinded with the output of the PUF O, so that the final helper value $P = ECC(p_i^{\text{priv}}) \oplus O$ reveals no information about p_i^{priv}. During recovery for authentication, computing the exclusive-or of $ECC(\text{rand}) \oplus O \oplus O'$ will return p_i^{priv} whenever O and O' are t-close. This process is referred to as *fuzzy extraction*, and is detailed further in the '848 application (see "Gen Algorithm,", "Rep Algorithm," and Definition 3).

- The device chooses a random group element $r \in \mathbb{F}_q$ and computes point $B = r \cdot G$.

- The server's nonce N is linked to the proof by constructing a hash c' that also combines the base point G, the device's nonce B, and its public key A.

- The device constructs the zero knowledge proof token $(B, m = r + c' \cdot p_i^{\text{priv}} \mod p)$, and returns this tuple to the server.

- The server verifies that:

$$\begin{aligned}\{(m \cdot G) - (c' \cdot A)\} &= \{((r + c' \cdot p_i^{\mathtt{priv}}) \cdot G) - (c' \cdot p_i^{\mathtt{priv}} \cdot G)\} \\ &= \{r \cdot G +' c \cdot p_i^{\mathtt{priv}} \cdot G -' c \cdot p_i^{\mathtt{priv}} \cdot G\} \\ &= r \cdot G \\ &= B\end{aligned}$$

$(1,1)$ Verification

[0020] In $(1,1)$ verification, the verifier individually interrogates each component in order to establish the integrity of the larger system; all (or all specified) components successfully complete a zero knowledge proof with the verifier in order for the verification of the integrity of the system as a whole to succeed. Referring to **Fig. 1**, the verifier is illustrated sequentially validating each of the system's components. At first verification 1 and second verification 2, the verifier validates each critical system component. At third verification 3 and fourth verification 4, the verifier validates each non-critical system component. An interactive version of this process is set forth in Algorithm 1.

Algorithm 1 Interactive $(1, 1)$ System Verification

for each Component Device $d_i \in \bar{D}$ **do**
  for Verifier $\mathcal{V}$ **do**
    $d_i \leftarrow \{c_i, G, p, P_i, N_i\}$ where N_i is a nonce and P_i is the helper string
  $x_i \leftarrow Hash(c_i, E, G, n)$
  $p_i^{\text{priv}} \leftarrow D(PUF(x_i) \oplus P_i)$ where $PUF(\cdot)$ is the PUF output function and D is an error decoding scheme
  $A_i = p_i^{\text{priv}} \cdot G \mod p$
  $r_i \leftarrow$ random $\in \mathbb{F}_q$, a random group element
  $B_i \leftarrow r_i \cdot G \mod p$
  $c_i' \leftarrow \text{Hash}(G, B_i, A_i, N_i)$
  $m_i \leftarrow r_i + c_i' \cdot p_i^{\text{priv}} \mod q$
  Verifier $\mathcal{V} \leftarrow \{B_i, m_i\}$
  for Verifier $\mathcal{V}$ **do**
    $A_i = p_i^{\text{priv}} \cdot G \mod p$ (stored from device enrollment)
    $c_i' \leftarrow \text{Hash}(G, B_i, A_i, N_i)$
    $D_i \leftarrow m_i \cdot G - c_i' \cdot A_i \mod p$
    Component Device $d_i \leftarrow \begin{cases} \text{accept} & : D_i = B_i \\ \text{deny} & : D_i \neq B_i \end{cases}$
for Verifier $\mathcal{V}$ **do**
  System $\leftarrow \begin{cases} \text{accept} & : \forall i, D_i = B_i \\ \text{deny} & : \exists i \text{ s.t. } D_i \neq B_i \end{cases}$

[0021] The requirement for communication from the verifier $\mathcal{V}$ in the interactive zero knowledge proof is to obtain a nonce value specific to the current proof. This prevents an eavesdropping adversary from using previous proofs from a valid component to successfully complete an authentication protocol and masquerade as a valid component. A *non-interactive* zero knowledge proof removes this communication requirement. A non-interactive version of Algorithm 1 can be made by configuring the component to generate a nonce in a manner that prevents the proving component from manipulating the proof. To accomplish this, the component device d_i constructs the nonce $N \leftarrow Hash(p_i^{\text{priv}} \cdot G \mod p || \tau)$ where τ is a timestamp and $||$ denotes concatenation. The timestamp ensures that previous proofs constructed by the proving component cannot be replayed by an adversary in the future, while the

hash function ensures that the proving component cannot manipulate the challenge in an adversarial manner. The verifier preferably checks that the timestamp is reasonably current (e.g., second granularity) and monotonically increasing to prevent replay attacks. Alternately, globally-synchronized clocks may be used rather than a timestamp, such as if network latency is not significant. A non-interactive version of $(1,1)$ verification is set forth in Algorithm 2, with each component locally choosing a current timestamp τ to construct its nonce.

Algorithm 2 Non-Interactive $(1,1)$ System Verification

for each Component Device $d_i \in \bar{D}$ **do**
- $x_i \leftarrow Hash(c_i, E, G, n)$
- $p_i^{\text{priv}} \leftarrow D(PUF(x_i) \oplus P_i)$ where $PUF(\cdot)$ is the PUF output function and D is an error decoding scheme
- $A_i = p_i^{\text{priv}} \cdot G \mod p$
- $r_i \leftarrow$ random $\in \mathbb{F}_n$, a random group element
- $B_i \leftarrow r_i \cdot G \mod p$
- $N_i \leftarrow \text{Hash}(A||\tau)$ where τ is the current timestamp
- $c_i' \leftarrow \text{Hash}(G, B_i, A_i, N_i)$
- $m_i \leftarrow r_i + c_i' \cdot p_i^{\text{priv}} \mod q$
- Verifier $\mathcal{V} \leftarrow \{B_i, m_i, \tau\}$
- **for** Verifier $\mathcal{V}$ **do**
  - $A_i = p_i^{\text{priv}} \cdot G \mod p$ (stored from device enrollment)
  - $N_i \leftarrow \text{Hash}(A||\tau)$
  - $c_i' \leftarrow \text{Hash}(G, B_i, A_i, N_i)$
  - $D_i \leftarrow m_i \cdot G - c_i' \cdot A_i \mod p$
  - Component Device $d_i \leftarrow \begin{cases} \text{accept} & : D_i = B_i \wedge \text{check}(\tau) \\ \text{deny} & : D_i \neq B_i \vee !\text{check}(\tau) \end{cases}$

for Verifier $\mathcal{V}$ **do**
- System $\leftarrow \begin{cases} \text{accept} & : \forall i, D_i = B_i \wedge \text{check}(\tau_i) \\ \text{deny} & : \exists i \text{ s.t. } D_i \neq B_i \vee !\text{check}(\tau_i) \end{cases}$

$(n,1)$ Verification

[0022] Referring to the threshold methods disclosed in the '920 application, which are incorporated by reference, for example, to satisfy the requirement that all k

critical components are functioning properly, a (k, k) sharing can be constructed such that all k components must collaborate to complete a joint proof. For a system that comprises critical components as well as non-critical or redundant components, it may be desired to verify that all critical and some non-critical or redundant components are operational. One method for verifying such a system is to generate a separate sharing for each set, whereby the verifier checks two proofs.

[0023] Alternately, both critical and non-critical components may be verified together using a single (t, n) threshold proof in which n shares are allocated so as to ensure integrity of all critical components and a specified subset of non-critical components. For example, assume there are two critical components (both of which must be operational) and two non-critical components (at least one of which must be operational). $n = 6$ shares can be distributed with each critical component receiving two shares and each non-critical component receiving one, and the minimum number of shares for verification t being 5 (i.e., a $(5, 6)$ sharing). If one of the critical components fails or both of the non-critical components fail, verification fails since the remaining operational components can only contribute a total of four shares; if both critical components and at least one non-critical component are operational, the five shares necessary for successful verification can be contributed. In general, this approach is referred to as constructing a threshold access structure, which enforces a set of rules (e.g., all critical components and half of the non-critical components must be functional for the system to authenticate) through the distribution of shares. As shown in **Fig. 2**, first threshold proof 5 and second threshold proof 6, the critical components contribute their local proofs. At third threshold proof 7 and fourth threshold proof 8, the remaining components contribute their local proofs to form a single, joint proof. At combined verification 9 the Verifier validates the joint proof

(such as by Algorithm 6) to establish the validity of the system as a whole. Similarly, a (t, n) sharing can be constructed for redundant systems, such that t of the n redundant systems must be functioning to complete the proof. Thus, rather than complete $O(n)$ proofs for n systems, the systems can jointly construct a single threshold proof to represent the system they compose.

[0024] Algorithm 3 illustrates an example of a subset of component devices $\bar{D} \subseteq D, |\bar{D}| = m \leq n$ constructing a joint threshold proof for the verifier $\mathcal{V}$. Although in this example the verifier combines partial proofs (thus, implying $O(n)$ work for $\mathcal{V}$ as the number of partial proofs is n), a secretary could instead combine the partial shares and forward the result to the verifier. As another alternative, the components could form a ring, and pass their partial shares to the next component, which combines their own partial proof before forwarding on to the next component. The Enrollment Algorithm, Distributed Key Generation Algorithm, and PUF-Retrieve are set forth in the '920 application.

Algorithm 3 Interactive Threshold Proof Construction

Goal: Perform threshold zero knowledge proof at time τ

One-Time Setup Stage

for each System Device $d_i \in \bar{\mathcal{D}}$ **do**

Run Enrollment Algorithm

Run Distributed Key Generation Algorithm

Evaluation Stage

for Verifier $\mathcal{V}$ **do**

Broadcast ephemeral nonce N to all $d_i \in \bar{\mathcal{D}}$

for all Participants $p_i \in \mathcal{P}$ **do**

Recover share $r_i \leftarrow$PUF-Retrieve(c_i, helper_i)

Choose a random $y_i \in \mathbb{F}_q$

Compute and broadcast $B_i = y_i \cdot G \mod p$ to all $d_i \in \bar{\mathcal{D}}$

Compute

$$B = y \cdot G = \sum_{i=1}^{t} B_i \mod p$$

$$e = Hash(G, B, \mathcal{P}^{\text{pub}}, N)$$

$$M_i = y_i + r_i e \left(\prod_{\substack{j=1 \\ j \neq i}}^{t} \frac{-j}{i-j} \right) \mod q$$

Send (B_i, M_i) to verifier $\mathcal{V}$

[0025] Similarly, Algorithm 3 can be performed non-interactively. This is accomplished by replacing the verifier's nonce N with a timestamp τ generated by the components, as illustrated in Algorithm 4. The timestamp serves as a replacement for the server's randomness N, and prevents replay attacks by adding a temporal requirement to the proof. That is, the timestamp is monotonically increasing, and the verifier simply checks that the timestamp used in the proof is reasonably (e.g., second granularity) current.

Algorithm 4 Non-Interactive Threshold Proof Construction

Goal: Perform non-interactive threshold zero knowledge proof at time τ

One-Time Setup Stage

for each System Device $d_i \in \bar{\mathcal{D}}$ **do**

Run Enrollment Algorithm

Run Distributed Key Generation Algorithm

Evaluation Stage

for all Participants $p_i \in \mathcal{P}$ **do**

Fix current timestamp τ

Recover share $r_i \leftarrow$PUF-Retrieve(c_i, helper_i)

Choose a random $y_i \in \mathbb{F}_q$

Compute and broadcast $B_i = y_i \cdot G \mod p$ to all $d_i \in \bar{\mathcal{D}}$

Compute

$$B = y \cdot G = \sum_{i=1}^{t} B_i \mod p$$

$$e = Hash(G, B, \mathcal{P}^{\text{pub}}, \tau)$$

$$M_i = y_i + r_i e \left(\prod_{\substack{j=1 \\ j \neq i}}^{t} \frac{-j}{i-j} \right) \mod q$$

Send (B_i, M_i, τ) to verifier $\mathcal{V}$

[0026] Algorithm 5 illustrates a further refinement of Algorithm 3 that incorporates updating the challenge-helper pair and share after each operation. The PUF-Share-Update and PUF-Store algorithms are set forth in the '920 application.

Algorithm 5 Interactive Threshold Proof Construction with Refreshing

Goal: Perform threshold zero knowledge proof at time τ

One-Time Setup Stage

for each System Device $d_i \in \bar{\mathcal{D}}$ **do**

Run Enrollment Algorithm

Run Distributed Key Generation Algorithm

Evaluation Stage

for Verifier $\mathcal{V}$ **do**

Broadcast ephemeral nonce N to all $d_i \in \bar{\mathcal{D}}$

for all Participants $p_i \in \mathcal{P}$ **do**

Recover share $r_i^{(\tau)} \leftarrow$PUF-Retrieve$(c_i^{(\tau)}, \text{helper}_i^{(\tau)})$

Choose a random $y_i \in \mathbb{F}_q$

Compute and broadcast $B_i = y_i \cdot G \mod p$ to all $d_i \in \bar{\mathcal{D}}$

Compute

$$B = y \cdot G = \sum_{i=1}^{t} B_i \mod p$$

$$e = Hash(G, B, \mathcal{P}^{\text{pub}}, N)$$

$$M_i = y_i + r_i^{(\tau)} e \left(\prod_{\substack{j=1 \\ j \neq i}}^{t} \frac{-j}{i-j} \right) \mod q$$

Send (B_i, M_i) to verifier $\mathcal{V}$

Update share

$$r_i^{(\tau+1)} \leftarrow \text{PUF-Share-Update}(r_i^{(\tau)})$$

Store $r_i^{(\tau+1)}$ and update PUF challenge:

$$\{c_i^{(\tau+1)}, \text{helper}_i^{(\tau+1)}\} \leftarrow \text{PUF-Store}(r_i^{(\tau+1)})$$

Algorithm 6 Interactive Threshold Proof Verification

for Verifier $\mathcal{V}$ **do**

Upon receipt of $(B_i, M_i)_{1 \leq i \leq t}$, compute:

$$B = \sum_{i=1}^{t} B_i \mod p$$
$$M = \sum_{i=1}^{t} M_i \mod q$$
$$e = h(G, B, \mathcal{P}^{\text{pub}}, N)$$

Verify the proof against the group's public key $\mathcal{P}^{\text{pub}} = rG$:

$$\begin{aligned} B &\stackrel{?}{=} M \cdot G - e \cdot \mathcal{P}^{\text{pub}} \mod p \\ &= (y + re) \cdot G - e \cdot (rG) \\ &= yG + reG - reG \\ &= yG \end{aligned}$$

$$\text{Decision} \leftarrow \begin{cases} \text{accept} & : B = yG \\ \text{deny} & : B \neq yG \end{cases}$$

Layered Security

[0027] When the components themselves are unable to generate a proof of correctness, the integrity of the system as a whole must be derived from a *root-of-trust.* An additional embodiment of the invention is a system achieving a layered security approach across all computing levels by deriving a hardware root-of-trust from a *high assurance processor.* The high assurance processor is used to validate all layers in a computing architecture, providing secure boot control, change detection, alarm indicators and audit functions. **Fig. 3** illustrates the high assurance processor in an exemplary computing architecture.

[0028] Secure computing architectures create a layered security approach, where

the trusted boundary is iteratively extended from a core root-of-trust. For example, a *trusted boot procedure* assumes a minimal trust boundary (e.g., a root-of-trust, such as a trusted platform module (TPM)) and iteratively extends the trust boundary by validating each component of the system as it boots. This mitigates risk from components more susceptible to adversarial modification, such as the operating system or applications. The root-of-trust is used to detect modification to system components, and will only complete the boot sequence if all components are validated as correct. However, existing trusted boot systems typically rely on roots-of-trust that are assigned (rather than intrinsic) to the device. For example, TPMs hold a private key in protected memory that represents the identity of the system. Thus, an adversary that extracts the assigned identity is able to masquerade as the system. Further, existing systems do not provide intrinsic tamper detection, and rely on tamper detecting hardware enclosures for security. Existing roots-of-trust are illustrated in **Fig. 3** at the root of trust layer 14, which is situated above the hardware layer.

[0029] One embodiment of the invention employs a high assurance processor based on a PUF that captures intrinsic and unique properties of the hardware and preferably provides intrinsic hardware tamper detection. As the PUF mapping is a function of the physical properties of the hardware, it can be used to generate a hardware-intrinsic identity that represents the physical state of the system. Referring to **Fig. 3**, high assurance processor 10, which is at the hardware layer, is established as the root-of-trust for the system and forms a layered security architecture interaction with application layer 11, operating system layer 12, network layer 13, root of trust layer 14, and hardware layer 15. The high assurance processor 10 addresses NIST SP 800-53 Rev. 4 ("Security and Privacy Controls for Federal Information Systems and Organizations") *Security Capability*, where trust is derived from interactions among system

components. The high assurance processor 10 may be used in mutual reinforcement controls within the system, where the high assurance processor 10 may validate an existing root-of-trust and vice versa.

[0030] The high assurance processor 10 is preferably designed to interact with the system through common commercial standard interfaces (e.g., USB, Ethernet) to enable interaction with commercial-off-the-shelf devices without hardware modification, and integration and continued support may be achieved through firmware and/or software upgrades. At root of trust layer 14 the high assurance processor 10 may be used to extend and/or interact with existing roots-of-trust (e.g., TPM, ARM TrustZone). This enables a system with an existing trusted boot process to remain essentially unchanged, as the high assurance processor 10 can first validate the existing root-of-trust (which can subsequently complete the existing trusted boot process). At application layer 11 the high assurance processor 10 may be used to validate applications prior to execution, for example by storing a cryptographic hash of the application code or binary executable when it is first installed from a trusted source. The high assurance processor 10 signs the cryptographic hash, which may be stored on the system. Before an application may be executed by the system, the high assurance processor 10 first computes a cryptographic hash of the current application code or binary executable, validates its signature on the stored cryptographic hash, and validates that the two hash outputs match. If any of these checks fail, the high assurance processor 10 preferably halts execution of the application and issues an alarm.

What is claimed is:

1. A device for verifying the integrity of a system comprising a plurality of components each having a physical state and a physical unclonable function ('PUF') including a PUF input and a PUF output and constructed to generate, in response to the input of a specific challenge, an output value that is characteristic to: i) the PUF, ii) the component's physical state, and iii) the specific challenge; the device comprising a memory, and a processor connected to each of the PUF-containing components and configured to perform threshold verification by computing threshold cryptographic operations over multiple shares each associated with a specific component and a response from that component's PUF and combining those threshold cryptographic operations and checking whether a sufficient set of PUF-containing components represented thereby is valid.
2. The device of claim 1, wherein the processor is configured to perform interactive verification.
3. The device of claim 1, wherein the processor is configured to perform non-interactive verification.
4. The device of claim 1, wherein the processor is configured to perform verification of one or more specified groups of components.
5. The device of claim 4, wherein the system includes critical components and the processor is configured to perform verification on k critical components through a (k,k) sharing construction.
6. The device of claim 1, wherein the system includes a total of n critical and non-critical components and the processor is configured to perform verification through a (t,n) sharing construction.

7. The device of any of claims 1, 2, 3, 4, 5, or 6, wherein the processor is configured to perform zero knowledge proof authentication.

8. The device of any of claims 1, 2, 3, 4, 5, or 6, wherein the processor is further configured to perform elliptic curve cryptography.

9. A system of components configured to attest integrity of the system to a verifying device, each component having a physical state and comprising:

a) a physical unclonable function ('PUF') including a PUF input and a PUF output and constructed to generate, in response to the input of a specific challenge, an output value that is characteristic to i) the PUF, ii) the component's physical state, and iii) the specific challenge; and

b) a processor connected to the PUF and configured to, in response to a verification request from the verifying device, provide an input to the PUF input and receive a response from the PUF output, and compute and convey to the verifying device a share for that PUF-containing component of a joint threshold proof associated with a specified set of PUF-containing components.

10. The device of claim 9, wherein the system includes a total of n critical and non-critical components and the zero knowledge proof is based upon a (t,n) sharing construction.

11. The device of claim 9, wherein the zero knowledge proof is based upon a (k,k) sharing construction.

12. The device of claim 9, wherein the zero knowledge proof is interactive.

13. The device of claim 9, wherein the zero knowledge proof is non-interactive.

14. The device of claims 9, 10, 11, 12, or 13, wherein the zero knowledge proof is based upon an elliptic curve mathematical framework.

15. The device of any of claims 9, 10, 11, 12, or 13, wherein the joint threshold proof is a zero knowledge proof.

AMENDED CLAIMS
received by the International Bureau on 29 June 2016 (29.06.2016)

1. A device for verifying the integrity of a system comprising a plurality of components each having a physical state and a physical unclonable function ('PUF') including a PUF input and a PUF output and constructed to generate, in response to the input of a specific challenge, an output value that is characteristic to the PUF, the component's physical state, and the specific challenge; the device comprising a memory, and a processor connected to each of the PUF-containing components and configured to perform threshold verification by:

a) computing threshold cryptographic operations over multiple shares, wherein each one of the multiple shares is associated with a specific component and a response from that component's PUF;

b) combining the computed threshold cryptographic operations; and

c) checking whether a sufficient set of multiple PUF-containing components represented by the multiple shares is valid.

2. The device of claim 1, wherein the processor is configured to perform interactive verification.

3. The device of claim 1, wherein the processor is configured to perform non-interactive verification.

4. The device of claim 1, wherein the processor is configured to perform verification of one or more specified groups of components.

5. The device of claim 4, wherein the system includes critical components and the processor is configured to perform verification on *k* critical components through a (*k*,*k*) sharing construction.

6. The device of claim 1, wherein the system includes a total of *n* critical and non-critical components and the processor is configured to perform verification through a (*t*,*n*) sharing construction.

AMENDED SHEET (ARTICLE 19)

7. The device of any of claims 1, 2, 3, 4, 5, or 6, wherein the processor is configured to perform zero knowledge proof authentication.

8. The device of any of claims 1, 2, 3, 4, 5, or 6, wherein the processor is further configured to perform elliptic curve cryptography.

9. A system of components configured to attest integrity of the system to a verifying device, each component having a physical state and comprising:

a) a physical unclonable function ('PUF') including a PUF input and a PUF output and constructed to generate, in response to the input of a specific challenge, an output value that is characteristic to i) the PUF, ii) the component's physical state, and iii) the specific challenge; and

b) a processor connected to the PUF and configured to, in response to a verification request from the verifying device, provide an input to the PUF input and receive a response from the PUF output, and compute and convey to the verifying device a share for that PUF-containing component of a joint threshold proof that requires shares from multiple PUF-containing components.

10. The system of claim 9, wherein the system includes a total of *n* critical and non-critical components and the joint threshold proof is based upon a (*t*,*n*) sharing construction.

11. The system of claim 9, wherein the joint threshold proof is based upon a (*k*,*k*) sharing construction.

12. The system of claim 9, wherein the joint threshold proof is interactive.

13. The system of claim 9, wherein the joint threshold proof is non-interactive.

14. The system of claims 9, 10, 11, 12, or 13, wherein the joint threshold proof is based upon an elliptic curve mathematical framework.

15. The system of any of claims 9, 10, 11, 12, or 13, wherein the joint threshold proof is a zero knowledge proof.

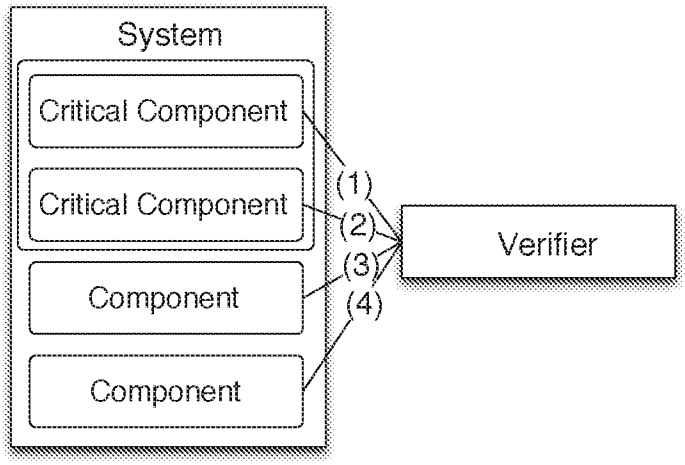

System
Critical Component
Critical Component
Component
Component
(1)
(2)
(3)
(4)
Verifier

Fig. 1

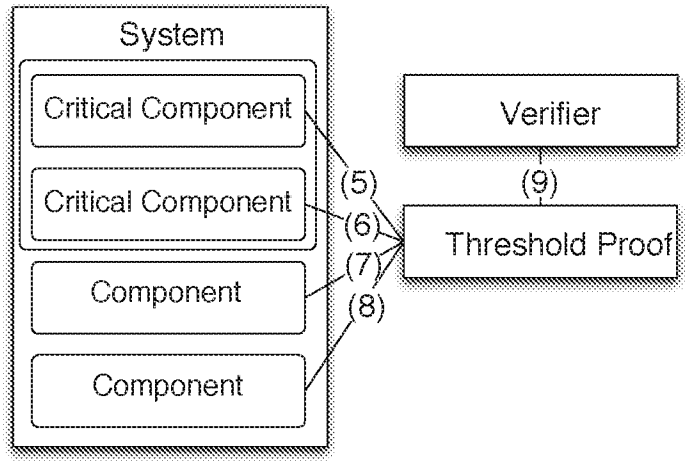

System
Critical Component
Critical Component
Component
Component
(5)
(6)
(7)
(8)
Verifier
(9)
Threshold Proof

Fig. 2

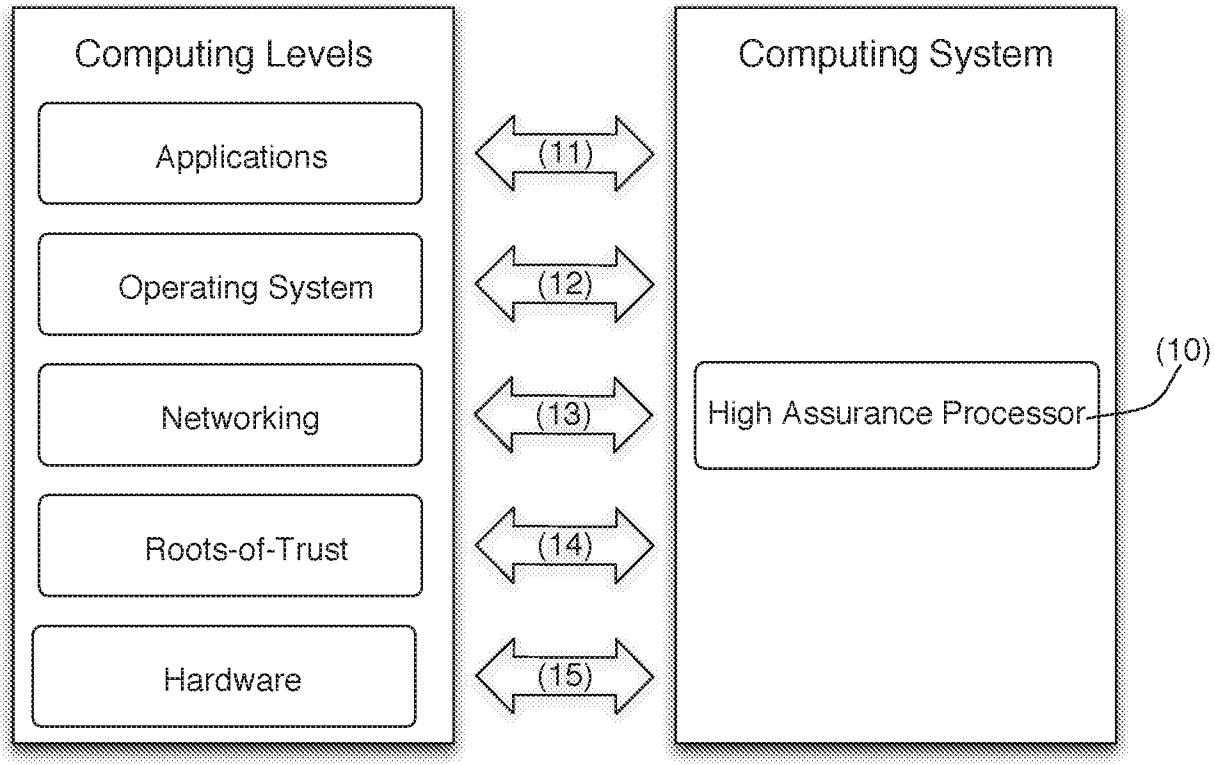

Computing Levels
Applications
Operating System
Networking
Roots-of-Trust
Hardware
(11)
(12)
(13)
(14)
(15)
Computing System
High Assurance Processor
(10)

Fig. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 16/21264

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 11/30 (2016.01)
CPC - G06F21/10, G11B20/00086, G11B20/0021, G06F2221/2107, H04L9/08
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC(8): G06F 11/30 (2016.01); CPC: G06F21/10, G11B20/00086, G11B20/0021, G06F2221/2107, H04L9/08

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC: 713/193, 713/194, 713/189

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PatBase, ProQuest Dialog, Google Web, Google Patents (Search terms: physical unclonable function, PUF, threshold, verification, share, authenticate, (1, n), (t, n), (k, k), threshold scheme, interactive, non-interactive, elliptic curve cryptography, joint threshold proof, zero knowledge proof, etc.)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	US 2011/0191837 A1 (Guajardo Merchan et al.) 04 August 2011 (04.08.2011), para. [0005], [0034], [0042], [0044]-[0047], [0053]-[0054], [0059]-[0065], and [0067], and Figs. 6a-6b, and Table 1, and claim 6.	1, 4-6, 9-11 ---------- 2-3, 7-8, 12-15
Y	US 8,918,647 B1 (Wallrabenstein) 23 December 2014 (23.12.2014), col. 5, ln. 56-60, and col. 13, ln. 6-9, and Fig. 1.	2, 7, 8/2, 12, 14/12, 15
Y	US 2010/0037056 A1 (Follis et al.) 11 February 2010 (11.02.2010), para. [0566] and [0580]-[0581].	3, 7/3, 8/3, 13, 14/13, 15/13
Y	US 2013/0051552 A1 (Handschuh et al.) 28 February 2013 (28.02.2013), para. [0104], [0143], and [0186].	8, 14

☐ Further documents are listed in the continuation of Box C. ☐

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
06 May 2016 (06.05.2016)	03 JUN 2016
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (January 2015)