SECURING PHYSICALLY UNCLONABLE FUNCTIONS WITH ADDITONAL RANDOM TERNARY STATES

FIELD OF THE INVENTION

[0001] The present disclosure relates to implementations of computing systems. Specifically, the disclosure describes implementations of physically unclonable functions (PUFs) that use ternary states for implementing security systems.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The present disclosure is illustrated by way of examples, embodiments and the like and is not limited by the accompanying figures, in which like reference numbers indicate similar elements. Elements in the figures are illustrated for simplicity and clarity and have not necessarily been drawn to scale. The figures along with the detailed description are incorporated and form part of the specification and serve to further illustrate examples, embodiments and the like, and explain various principles and advantages, in accordance with the present disclosure, where:

[0003] FIG. 1 depicts a table illustrating different memory-based PUFs.

[0004] FIG. 2 is a mathematical formula for randomness in the system.

[0005] FIG. 3 is a mathematical formula for something the number of improvements possible.

[0006] FIG. 4 depicts the partitioning of cells of a memory-based PUF.

[0007] FIG. 5 depicts an equation used to limit random numbers.

[0008] FIG. 6 depicts an equation to describe possible configurations in an embodiment.

[0009] FIG. 7 depicts an equation to describe possible configurations in an embodiment.

[0010] FIG. 8 depicts an illustration of using two random numbers to blank cells.

[0011] FIG. 9 depicts an illustration of linking random numbers to blank more cells.

[0012] FIG. 10 depicts a table outlining protocol for PUF challenge generation.

[0013] FIG. 11 depicts a table outlining protocol for PUF response generation.

[0014] FIG. 12 is a block diagram of an APG architecture.

DETAILED DESCRIPTION OF PREDERRED EMBODIMENTS

[0015] PUFs generate from hardware components can be the equivalent of human DNA or finger prints to strengthen the level of security of authentication protocols, and this as part of a set of cryptographic primitives. PUFs exploit intrinsic natural manufacturing variations, which are introduced during fabrication of the devices such as local variations in critical dimensions, doping levels of semiconducting layers, and threshold voltages. These variations make each device unique and identifiable from each other. The underlying mechanism of PUF is the creation of a large number of input-output interactions, called Challenge Response Pairs (CRPs), which are unique to each device. Once deployed during the authentication cycles, the PUFs are queried with challenges.

[0016] Memory arrays of all sorts, as shown FIG 1, have been reported as excellent elements to generate PUFs. Usually PUFs need only 128 to 256 bits, while commercial secure memory arrays (SM) which are integrated within secure micro-controllers, could have memory densities in the mega-byte range.

[0017] One of the generic methods to generate CRPs is to characterize a particular parameter $\mathcal{P}$ of the cells of the array with a "built-in-self-test" (BIST) module. Each cell being different, the value of parameter $\mathcal{P}$ varies cell to cell, and follows a distribution with a median value T. For challenge and response generations, all cells with $\mathcal{P}$ below T can be then considered as "0", all others as "1".

[0018] The resulting streams of data generated by the method can be used as cryptographic primitives to authenticate the memory array, and this because they randomly vary between arrays due to manufacturing variations. As it is presented below in the present disclosure, one example embodiment of Addressable Physical unclonable function (PUF) Generators (APG) uses memory based PUFs.

[0019] Some conventional approaches describe how the quality of a memory based PUF can be enhanced with the use ternary states. Rather than testing the cells to simply sort them into traditional binary "0" and "1" states as a function of the value of parameter $\mathcal{P}$, and the transition threshold T, in an example embodiment of the current disclosure, the cells that are too close to T, shaky, or unstable may carry the ternary state "X". The remaining "0"s and "1"s are thereby more predictable when subjected to repetitive queries, which reduces the CRP error rates of the PUFs.

[0020] An objective of the APG architecture is to enhance the security of the cyber physical systems (CPS), and to make attacks such as the hacking of databases of UserID-Password pairs more difficult. The novel database-free password generator architecture is based on a new component, the APG. APGs can generate passwords and authenticate a client on the network without having access to storage elements or look-up tables with UserID-Password pairs. These APG architectures use a large number of memory based PUFs, as described below, including true random number generators, and hash functions.

[0021] Using brute force attacks which assume total randomness of the PUF, the longer the PUF, the higher the entropy, i.e. the rate of randomness. For example, if N cells are used to generate a challenge, the number of possible configurations $\mathcal{E}$ of a data stream increases with the use of ternary states as shown in the mathematical equation in FIG. 2.The total improvement in the number of possible configurations due to the use of ternary states is described in FIG. 3.

[0022] The attack that may be prevented is one in which the crypto-analyst has access to the PUFs, knows how to measure parameter $\mathcal{P}$, can rank all cells based on the value of parameter $\mathcal{P}$, can find the value of the transition threshold T, and can thus approximatively find a way to correctly blank the unstable cells with an "X". For example, if the PUF has N=256 cells, then the number of possible responses for the crypto-analyst can be reduced to 256 which is undesirable for cryptography and cryptosystems.

[0023] The new method that is presented in this disclosure has the objective of eliminating the attack scenario presented above. The method, as it is shown in FIG. 3, is based on the partial random blanking of the cells of a memory based PUF.

[0024] During challenge generation, the totality of the cells of a memory based PUF, the A-cells, are tested, and sorted between the ones that have a strong "0" or "1" state, and the unstable ones that are blanked with an "X" state, as illustrated in FIG. 4. These cells are blanked due to their poor quality. Additionally an arbitrary and random number of cells of the PUF that have strong "0" and "1" states are also blanked with an "X", as represented in FIG. 4. The remaining cells that have strong "0" or "1" state are not blanked, as represented in FIG. 4. To take an example PUF with N cells, there may be n number of unstable cells that need to be blanked because their parameter $\mathcal{P}$ is too close to the transition point T. An additional number k cells may also be randomly selected to be blanked. In one embodiment, limitations may be placed on the random selection of cells according to the equation in FIG. XX.

[0025] The resulting challenge will be a stream of N ternary bits with values of either "0", "1", or "X". In an example embodiment of this method, the data stream is converted into a stream of 2N binary bits by transforming the "0s" into a (01), the "1s" into a (10), and the "Xs" into either a (00) or a (11). The challenges can also be encrypted, and may be stored in a secure server. Other possible methods to handle data streams of ternary bits may be used in other embodiments.

[0026] During response generation, only the cells of the PUF that are not blanked with an "X" are tested again. The authentication is positive if the cells that are tested yield the same data stream, and are similar to the one that was tested during challenge generation. The challenge-response-pair (CRP) error rates have to be low enough for positive authentication.

[0027] If a crypto-analyst knows the exact number of unstable cells in a PUF as well as the number of strong cells that are randomly blanked, such as depicted in FIG. 4, the total number of possible configurations $\mathcal{E}$, can be related to the entropy. In the methods of the current disclosure, the crypto-analyst may not know precisely how many unstable cells or randomly blanked cells are in a given PUF. This increases the number of possible configurations $\mathcal{E}+$ than the number that could be calculated with the known number of cells. The revised number of configurations $\mathcal{E}+$ that exist for a crypto analyst through the methods of this disclosure would be much higher.

[0028] FIG. 6 illustrates the mathematical equation that may be used to compute a number of possible configurations for the scenario where the number of unstable and randomly blanked cells is known. FIG. 7 illustrates the mathematical equation used to compute the revised number of possible configurations when neither the number of unstable or randomly blanked cells is known.

[0029] In FIG. 7 the variables **n**min to **n**max represent a possible range of unstable cells that may be reasonable for crypto-analysis. The largest number of configurations occurs when the number of unstable cells is minimized. The crypto-analyst also cannot know the precise number of randomly assigned blank cells. In one embodiment, number of randomly blanked cells k can be varied. The entropy generated within this example embodiment is then converging with a PUF configuration that approaching pure randomness of ternary states within the example N number of cells.

[0030] In summary, blanking all unstable cells with ternary states "X", together with the blanking of an additional arbitrary number of random cells can result in PUFs having a low CRP error rate and high entropy level. This could be used to protect from a crypto-analysis attack.

[0031] One important element of this disclosure is randomness. Particularly in one example embodiment, where random selection of an additional number of strong cells k to be blanked is chosen from the cells that are left un-blanked after the testing of all the cells of a PUF. There are several ways to do this random cell selection. A first method is to select k cells is based on the generation of a true random number, and to blank 50% the cells of the PUF corresponding to a 0. The cells of a PUF can then be blanked either because they need to be blanked for quality reasons (the cells are unstable, as parameter $\mathcal{P}$ is close to the transition T), or because the random number is a zero for these cells. As a result of such a method, the number of randomly blanked cells would be half of the strong cells on a PUF. To reduce the number of randomly blanked cells, a second true random number can be used to reverse some of the randomly blanked strong cells. A third iteration could be applied to increase the number of blanked cells. A different true random number could be used to blank half of the remaining cells. With successive iterations, the number of randomly blanked cells k can be adjusted up or down between 0 and the maximum number of strong cells on a PUF.

[0032] A second method to randomly select k cells to blank is based on two random numbers TRN-1 and TRN-2 that can be applied in parallel to the non-random cells. This is illustrated in FIG. 8. An output of two XOR gates can be further XORed to generate an output. This method increases the randomness of k.

[0033] A third method to generate randomly blanked k cells is based on a scheme where the number of additional cells (k) can be varied by choosing a particular number of stages. This scheme is shown in FIG. 9. In FIG.9, it is shown how to blank a variable number of cells. For this purpose, a stage compression scheme can be used. Each stage divides the number of cells to be blanked by a factor of 2.

[0034] The protocol described above to generate challenges is shown in FIG. 10. Step 4.0 in this protocol serves to increase arbitrarily the number of blanked "X" cells. As part of Step 4.0, one possible embodiment of the insertion of arbitrary "X" cells is to use random numbers, as described by the methods above. Using this protocol, a third party should not be able to

differentiate a C" PUF challenge generated with this method from the challenges generated from a memory array with ternary states.

[0035] The protocol for response generation and authentication described above is shown in FIG 11. With this protocol a third party should not be able to differentiate the generation of R' PUF responses generated with this method from the responses generated from a memory array with ternary states. The challenges contain all information related to finding of the location of the B-cells (the non-blanked cells as depicted in FIG. 4). These are the ones needed to be tested for PUF responses. The authentication is positive if the error rates during CRP matching is low enough. A low error rate can only be realized if the respective position of the blanked "X" cells is accurate, and if the measurement of parameter $\mathcal{P}$ is reproducible between challenges and responses.

[0036] An example of block diagram representing a hardware implementation of APGs, is shown FIG. 12.

[0037] The hash function converts the UserID of a particular client or terminal into an address where a PUF is located. In order to generate a challenge from a PUF in question, a random number TRN1 is also generated to blank the additional arbitrary k cells from the PUF. Without this information, a crypto-analyst cannot extract a correct response from the PUF.

[0038] The challenges generated using the method discussed above may use a randomization process for the blanking of the additional k "X" cells that can be based on a random number TRNa. After authentication, and the generation of matching responses, the protocol can generate different challenges based on a different random number TRNb that can be kept for the following authentication. Thereby the method reduces its exposure to a crypto-analyst's ability to intercept challenges which are used only once.

[0039] Example embodiments of the method described above may implement PUFs that are based on memory arrays of any type, including but not to be limited to, SRAM, DRAM, Flash, EEPROM, PROM, one-time programmable arrays, Resistive RAM (ReRAM), MRAM, Phase change memory, conductive bridge, carbon nanotubes, and others. The examples shown FIG. 1 are all based on memory arrays that can be arbitrary blanked with "X states as described above.

[0040] The method of this disclosure can be extended beyond memory based PUFs. In some embodiments, the PUFs can be used with memories that have known patterns stored such as

cryptographic keys, biometric prints, or passwords. A challenge can be generated by extracting such patterns and blanking them randomly and arbitrarily. Crypto-analysts, able to read the memory are then prevented from finding a response matching the challenge. This method may be easier to implement on legacy systems.

[0041] The method described in this disclosure is directly applicable to an array of any number of PUFs. These arrays may be based on methods such as arrays of ring oscillators, arrays of gate delays, unclonable images, PUF sensors, or similar. In such a case the challenges may either be streams of N ternary bits (0, 1, X), or streams of 2N binary bits (0, 1).

[0042] The method described in this disclosure to add arbitrary and random "X" cells may not have to be combined with the use of ternary states to blank the unstable cells. The method may be used with binary PUFs, or with PUFs having more than three states such as quaternary state PUFs.

[0043] A cryptographic protocol may be based on reference pattern generation that is made of data streams, challenges, and subsequent pattern generation also made of data streams, and responses, which are used to authenticate components when the challenge-response-pairs are similar. In this protocol, random and arbitrary portions of the data streams which are part of the challenges are blanked with an X-state, while the remaining portions of the data stream consists of the stream of binary bits, 0 and 1 previously generated. A correct authentication needs a response protocol in which the positions of the portions of the data streams that are blanked with an X-state are correctly identified, and in which the remaining data steam is similar than the one generated during challenge generation. The data streams may be the result of reading the content of memory arrays, which may be based on cryptographic keys, biometric prints, passwords, unclonable images, or other data bases. The data streams may be the result of challenges generated by physical unclonable functions (PUF),that are based on ring oscillators, gate delays, sensor PUFs, or memory based arrays. The memory arrays may be made of SRAMs, DRAMs, Flash, EEPROMs, PROMs, one-time programable arrays, Resistive RAMs, CB-RAMs, MRAM, PC-RAM, or carbon nanotube based memories. The PUFs may generate binary, ternary, quaternary, or multi-states data streams. The approach may be used to convert a random and arbitrary portion of the data streams that are part of the challenges blanked X-state, based on random numbers, pseudo random numbers, or true random numbers. The random numbers may be integrated to the positions of non-blanked cells to select k cells. These k cell can be additional

to the list of blanked cells. The protocols can be based on random numbers that change as much as every time after authentication in order to generate new challenges for subsequent authentications. The PUFs can be organized as arrays of addressable PUF generators (APG). In such an implementation, the method can then be inserted as part of the circuitry of the APG to protect the array of PUFs. The method to convert a data stream of bits with ternary states into a data stream of bits with binary states, or the reverse, can be based on the association of 0 state with (01), 1 state with (10), and X state with either (00) or (11). The X states can alternate from (00) to (11) with a randomization methods that can use random numbers.

[0044] This disclosure is intended to explain how to fashion and use various embodiments in accordance with the invention rather than to limit the true, intended, and fair scope and spirit thereof. The foregoing description is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications or variations are possible in light of the above teachings. The embodiment(s) was chosen and described to provide the best illustration of the principles of the invention and its practical application, and to enable one of ordinary skill in the art to utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated. All such modifications and variations are within the scope of the invention as determined by the appended claims, as may be amended during the pendency of this application for patent, and all equivalents thereof, when interpreted in accordance with the breadth to which they are fairly, legally, and equitably entitled.

CLAIMS

What is claimed is:

1. A method, comprising:

receiving a first data stream, the data stream encoding binary bits and X-values; and

authenticating the first data stream using a physically unclonable function (PUF) by determining whether a location of the binary bits and X-values within the first data stream corresponds to locations of binary bits and X-values in a second data stream generated using the PUF.

2. The method of claim 1, wherein the second data stream is generated by reading a content of a memory array of the PUF.

3. The method of claim 1, wherein the PUF includes at least one of a ring oscillator, gate delay, sensor, and memory based array.

4. The method of claim 1, including reading the first data stream from a memory array.

5. The method of claim 4, wherein the memory array includes at least one of an SRAM, DRAM, Flash, EEPROM, PROM, one-time programmable array, Resistive RAM, CB-RAM, MRAM, PC-RAM, or carbon nanotube based memory.

6. The method of claim 1, wherein the PUF is contained in an array of addressable PUF generators.

ABSTRACT

The present disclosure relates to implementations of computing systems. Specifically, the disclosure describes implementations of physically unclonable functions (PUFs) that use ternary states for implementing security systems.

MEMORY	Parameter for PUF Generation	Comments
SRAM	*Random Flip of the 6T cell:* start as a "0" or a "1" after power up	*Mainstream but not always secure*
DRAM	*Discharge the capacitors, then measure voltage: Get a "0" or a "1"*	*Need constant refresh*
Flash	*Partial programming, then measure threshold:* Get a "0" or a "1"	*Slow programming*
ReRAM	*Variations of the value of the Vset:* Define a "0" or a "1"	*Quite novel*
MRAM	*Variations of the Rmax's after programming:* Define a "0" or a "1"	*Quite novel*

Table 1: Memory based PUFs

FIG. 1

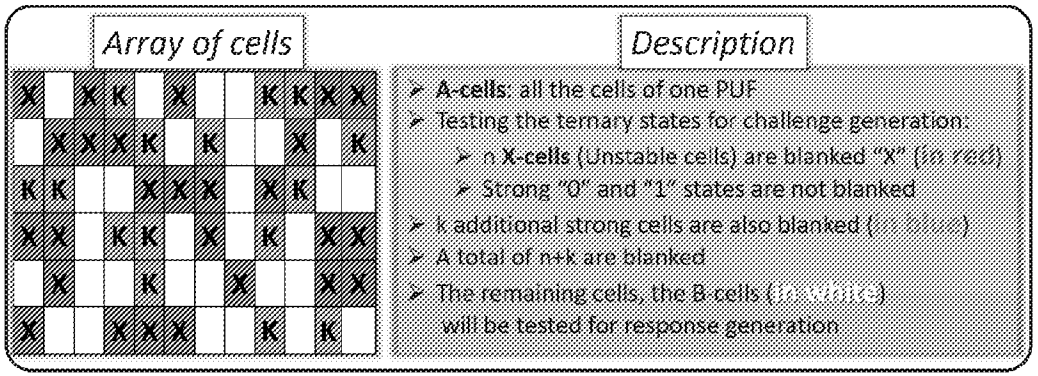
If $\mathbf{N=256}$, then $\mathscr{E} = 2^N \rightarrow \mathscr{E} = 3^N = 2^N \bullet 1.5^N$ Eq. 1

FIG. 2

$$\Delta \mathscr{E} = 1.5^{N} = 1.2\ 10^{45} \approx 2^{150} \qquad \text{Eq 2}$$

The entropy is then increased from **256 to 256+150=406**.

FIG. 3



Partitioning of the cells of a memory based PUF.

FIG. 4

$\forall \mathbf{k},\ \mathbf{n}+\mathbf{k} < \mathbf{N}$ $\mathbf{N}$, $\mathbf{n}$, and $\mathbf{k}$ are integers Eq.3

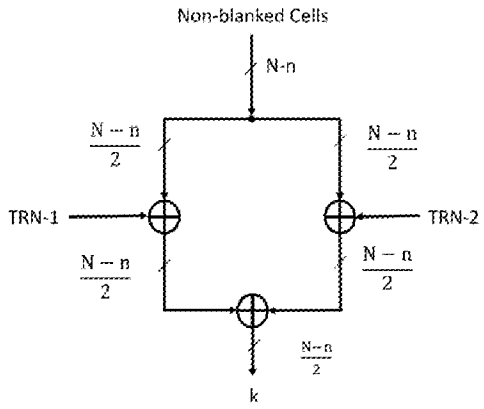
FIG. 5

$$\mathscr{E} = \binom{N-n}{k} = \frac{(N-n)!}{k!\,(N-(n+k))!} \qquad \text{Eq 4}$$

FIG. 6

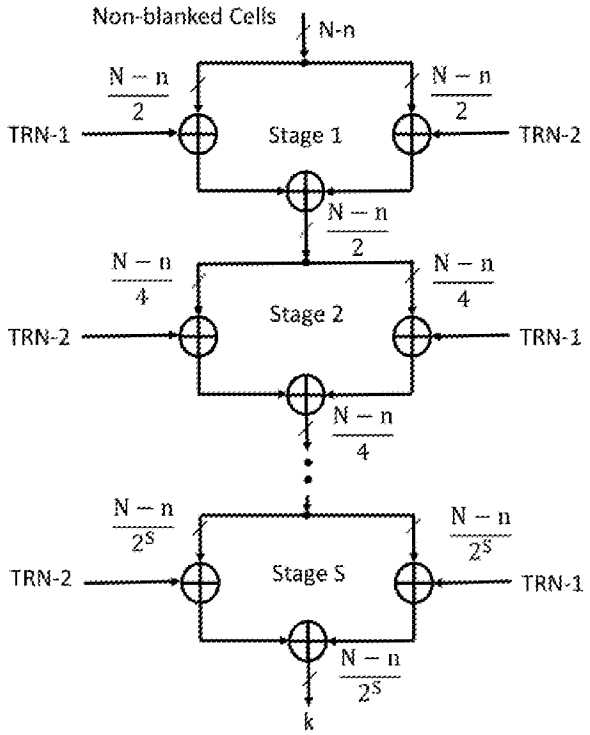
$$\mathscr{C}^{+} = \sum_{n=n_{min}}^{n=n_{max}} \frac{(N-n)!}{k!(N-(n+k))!}$$

FIG. 7



*Using two random numbers to blank **k** more cells*

FIG. 8



Using two random numbers to blank $\boldsymbol{k = \frac{N-n}{2^S}}$ *more cells*

FIG. 9

Step	PUF challenge Generation: Description of the instructions	Data stream/information	Where
1.0	Identify the N cells "**A**" in the PUF for challenge generation	{**A**1, **A**2, ..., **A**N}	**PUF or Server**
2.0	Measure parameter $\mathcal{P}$ for every "**A**" cells	{**A**1, **A**2, ..., **A**N} → {$\mathcal{P}$1, $\mathcal{P}$2, ..., $\mathcal{P}$N}	**PUF**
3.0	Generate ternary data stream **C** ϵ {0, 1, X} n cells are blanked "X" because $\mathcal{P}$ is close to **T**	{$\mathcal{P}$1, $\mathcal{P}$2, ..., $\mathcal{P}$N} → {**C**1, **C**2, ..., **C**N}	**PUF**
4.0	Randomly convert an additional k cells to "X" (out of the N-n cells that are not blanked)	{**C**1, **C**2, ..., **C**N} → {**C'**1, **C'**2, ..., **C'**N} n "X" cells → (n+k) "X" cells	**PUF**
5.0	Convert ternary **C'** into a binary stream of data **C"** 0 → (01) 1 → (10) X → (11) or (00)	{**C'**1, **C'**2, ..., **C'**N} → {**C"**11, **C"**12, **C"**21, **C"**22, ..., **C"**N1, **C"**N2}	**PUF or Sever**
6.0	**C"** is the PUF challenge in its final form Encrypt **C"** to generate the cipher **M**	**M** = E(**C"**)	**PUF or Server**
7.0	Communicate **M** to the secure server Store in the secure memory	**M**	**Server**

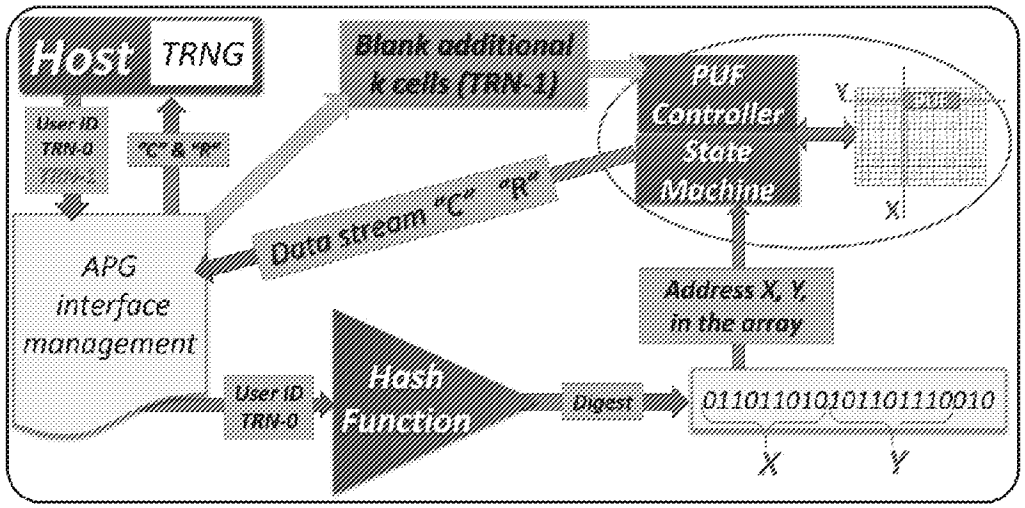
Table 2: protocol for PUF challenge generation.

FIG. 10

Steps		PUF Response Generation & authentication: Description of the instructions	Data stream/information	Where
1.0		Extract **M** from the secure memory	**M**	Server
2.0		Decrypt **M** to generate the cipher **C"**	**C"** = D(**M**)	PUF or Server
3.0		Convert binary **C"** into a ternary stream of data **C'** (01) → 0 (10) → 1 (11) or (00) → X	{**C"**11, **C"**12, **C"**21, **C"**22, ..., **C"**N1, **C"**N2} → {**C'**1, **C'**2, ..., **C'**N}	PUF or server
4.0	4.1	Identify the N-(n+k) non-blanked B-cells in the PUF	{**B**1, **B**2, ..., **B**N-(n+k)}	PUF or Server
	4.2	Keep track of the position of the "**B**" cells within **C'**. Generate **D** → **D**i = 0 when a "X" is present in C'i → **D**i = 1 when a B-cell is present in C'i	{**C'**1, **C'**2, ..., **C'**N} → {**D**1, **D**2, ..., **D**N)}	PUF or Server
5.0		Measure parameter $\mathcal{P}$ for every "**B**" cells	{**B**1, **B**2, ..., **B**N-(n+k)} → {$\mathcal{P}$1, $\mathcal{P}$2, ..., $\mathcal{P}$N-(n+K)}	PUF
6.0		Generate binary data stream of responses **R** ∈ {0, 1}	{$\mathcal{P}$1, $\mathcal{P}$2, ..., $\mathcal{P}$N-(n+K)} → {**R**1, **R**2, ..., **R**N-(n+k)}	PUF
7.0		Convert the binary stream of responses **R** into the ternary data stream **R'** with {**D**1, **D**2, ..., **D**N)}	{**R**1, **R**2, ..., **R**N-(n+k)} → {**R'**1, **R'**2, ..., **R'**N} N-(n+k) binary cells → N ternary cells	PUF or Server
8.0		Compare challenges **C'** with responses **R'** for authentication	{**C'**1, **C'**2, ..., **C'**N} ←→ {**R'**1, **R'**2, ..., **R'**N} C'i R'i Pair matching	PUF or Server

Table 3: Protocol for PUF response generation

FIG. 11



Block diagram of an APG.

FIG. 12