A Modeling Attack Resistant Deception Technique for Securing PUF based Authentication


Published in:
Asian Hardware Oriented Security and Trust Symposium (AsianHOST)

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Publisher rights
Copyright 2019 IEEE. This work is made available online in accordance with the publisher’s policies. Please refer to any applicable terms of use of the publisher.

General rights
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person’s rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.
A Modeling Attack Resistant Deception Technique for Securing PUF based Authentication

Chongyan Gu*, Chip Hong Chang†, Weiqiang Liu‡, Shichao Yu*, Qingqing Ma‡ and Máire O’Neill∗

*Centre for Secure Information Technologies (CSIT), ECIT
†Queen’s University Belfast, Belfast, U.K. {cg01, syu08}@qub.ac.uk, m.oneill@ecit.qub.ac.uk
‡School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore, echchang@ntu.edu.sg

Abstract—Due to practical constraints in preventing phishing through public network or insecure communication channels, simple physical unclonable function (PUF)-based authentication protocol with unrestricted queries and transparent responses is vulnerable to modeling and replay attacks. In this paper, we present a PUF-based authentication method to mitigate the practical limitations in applications where a resource-rich server authenticates a device with no strong restriction imposed on the type of PUF designs or any additional protection on the binary channel used for the authentication. Our scheme uses an active deception protocol to prevent machine learning (ML) attacks on a device. The monolithic system makes collection of challenge response pairs (CRPs) easy for model building during enrollment but prohibitively time consuming upon device deployment. A genuine server can perform a mutual authentication with the device at any time with a combined fresh challenge contributed by both the server and the device. The message exchanged in clear does not expose the authentic CRPs. The false PUF multiplexing is fortified against prediction of waiting time by doubling the time penalty for every unsuccessful authentication.

Index Terms—Physical Unclonable Function (PUF), machine learning attacks, authentication protocol.

I. INTRODUCTION

A physical unclonable function (PUF) is a security primitive that utilizes intrinsic manufacturing process variations to generate a unique digital fingerprint that is inherently difficult to clone and tamper-evident [1]. Since the initial proposal to leverage on Hamming distance (HD) threshold for lightweight PUF-based authentication [2], similar PUF-based authentication protocols have been derived to endow linearly-sized Strong PUF (SPUF) circuits with exponentially large CRP capacity for device authentication. Unfortunately, SPUFs used for device authentication with limited nonlinear mixing have been shown to be vulnerable to ML based modeling attacks. Masquerade attacks can be perpetrated through malicious nodes and unprotected communication links of ad hoc networks to efficiently collect a large number of CRPs to model the PUF.

A survey on entity authentication protocols using PUFs [3] shows that most PUF protocols [4]–[8] are heavyweight or require complicated protocol operations that add to the hardware implementation area, power and performance overheads. Only two protocols, the slender PUF [9] and noise bifurcation [10], require neither an error correction code (ECC) nor a strong cryptographic algorithm but a true random number generator (TRNG) to provide heuristic security against modeling attacks. The idea of slowing down the read out of PUF response was first raised in [11]. The SHIC PUF [11], implemented by novel semiconductor technology, does not integrate well with conventional CMOS designs [12]. The incompatibility raises constraints on the nano-micro link. As SHIC is a weak PUF, its CRP space grows only linearly as opposed to exponentially with array size. Recently, Yu et al. [12] proposed two lockdown techniques to limit the number of authentication requests. The first lockdown protocol allows only unilateral authentication while the second lockdown protocol requires a TRNG on the device side to generate a device nonce for bilateral authentication initiated by the server. The responses are sent in clear and a run time check for state segment overlap of linear-feedback shift register (LFSR) is required to prevent replay attack. Lockdown protocols support only a limited number of authentications, and rely on the hardness of XOR PUF to support model-based authentication. The hardware cost of XOR PUF increases and reliability reduces commensurately with increasing number of XORs for higher ML resistance. The most recent work from Gao et al. [13] uses a reconfigurable latent obfuscation technique to conceal and distort the relationship between CRPs. The pattern vectors for challenge and response obfuscation are selected by a random number generator (RNG), and are made latent and reconfigurable per authentication session. Nevertheless, the most recent report shows that both methods [12], [13] are vulnerable to the protocol attack [14].

In this paper, a deception based authentication scheme is proposed to allow a server to authenticate a delayed device at any time without specifically limiting the number of authentications. A monolithic architecture is built around a lightweight machine learnable SPUF at the device side to enable a constant size SPUF model to be built during enrollment and kept at the server side for server initiated bilateral authentication. Upon deployment, the device tracks the time between two successive challenge applications and uses this to deceive or excessively delay the adversary attempting to collect the CRPs by brute-
force queries made on the device or protocol interface. A fake SPUF is used to generate false responses to the adversary who uses the enrollment interface to query the device within the waiting period. The fake SPUF is reused for nonce generation at the device side in response to the authentication request made by the server who has the knowledge of its unused CRPs. No response is transmitted in clear in the proposed bilateral protocol to prevent eavesdropping attacks. The waiting time to adversary consists of a static and a dynamic component. The dynamic component escalates on each fake response generation, which rapidly blows up the waiting time on adversarial queries. The false PUF multiplexing is also complemented by the random adjustment of the dynamic waiting time component for each successful mutual authentication by the genuine server to further delude the attacker into using an incorrectly predicted waiting time for model building attack.

II. The Proposed Deception Protocol

To support machine learn resistant model-based authentication, the genuine SPUF (SPUF\textsuperscript{G}) is monolithically integrated with a fake SPUF (SPUF\textsuperscript{F}), a pseudo random number generator (PRNG), a \(L\)-bit counter, a \(L\)-bit register, a comparator and a simple controller as shown in Fig. 1. When an \(n\)-bit input challenge \(c\) is applied to the device, \(m\) sub-challenges are derived from the PRNG using \(c\) as a seed. If the select enable strobe \(S\) of the multiplexer is 0, these sub-challenges are input to SPUF\textsuperscript{G} to generate an \(m\)-bit response. Otherwise, the response will either be generated from SPUF\textsuperscript{G} or SPUF\textsuperscript{F} depending on the comparator output \(Z\). The comparator compares the output of \(Q\) with that of \(P\). \(Z = 1\) if they are not equal and \(Z = 0\) otherwise. \(Q\) is cleared upon power-up reset or when \(c\) is applied to it, \(Z = 0\). A simple controller is used to manage the inputs and outputs to the external world when the device is queried.

A. Enrollment

During enrollment, sufficiently large number of CRPs are extracted from SPUF\textsuperscript{G} to train an authentication verification model SPUF\textsuperscript{G}\_model by machine learning algorithm for later reproduction of any CRPs of SPUF\textsuperscript{G} with a high accuracy. As shown in Fig. 2, after SPUF\textsuperscript{G}\_model has been successfully built, one bit of register \(P\) is forced to stick at logic one permanently by an irreversible antifuse or an one-time programmable (OTP) non-volatile memory (NVM) bitcell. Upon device deployment, if the same protocol is used by the adversary to query the system, the response will only be output from SPUF\textsuperscript{G} if the challenge is input after a duration of \(T_{\text{min}}\), where \(T_{\text{min}} = 2^k \times T_{\text{clk}}\).\(k\) is the stuck-at-one bit of \(P\). To prevent the adversary from modelling SPUF\textsuperscript{G} after device deployment, \(k\) and \(T_{\text{clk}}\) are designed such that it is impossible for an adversary to collect a minimum number of CRPs from SPUF\textsuperscript{G} to accurately model SPUF\textsuperscript{G} in a practical amount of time. \(T_{\text{min}}\) can be adapted to the machine-learn resistance of the chosen SPUF\textsuperscript{G} and the use-case. Even with the same number of CRPs required for a successful attack on a selected SPUF, \(T_{\text{min}}\) can be set differently according to the application risk due to the different service span and diminishing profitability of a successful attack with time.

B. Deception-based Technique Against Model-building Attack

Fig. 2 shows the devices action when the enrollment protocol is utilized by the adversary to collect the CRPs for model building. When the device is queried by a challenge \(c\), the device sets \(S = 1\) and reads the comparator output \(Z\). If \(Z = 0\), the time lapse \(T\) of the current query from the last output response is at least \(T_{\text{min}}\), this frequency of authentication events is perceived to be normal with enough safety margin against model building attack. The controller will apply the sub-challenges \(\langle c \rangle\) derived from \(c\) to SPUF\textsuperscript{G} to produce a response \(\tilde{f}\). When \(T = Q \times T_{\text{clk}} < T_w\), where \(T_w = P \times T_{\text{clk}}\), then \(Z \neq 0\) signifies that the authentication events are unusually frequent. In this case, a response \(\tilde{f}\) from a fake SPUF (SPUF\textsuperscript{F}) will be output by applying \(\langle c \rangle\) to it, followed by clearing \(Q\) with \(\tilde{c}\tilde{f} = 0\) and doubling \(P_{os}\). \(P_{os}\) is a nonzero integer that can be loaded into \(P\) by the server to extend \(T_w\) beyond \(T_{\text{min}}\), through the proposed bilateral authentication protocol described in the next subsection.

As each consecutive query within the waiting time \(T_w\) will double \(P_{os}\), the waiting time will extend rapidly. Hence, an adversary who try to brute-force attack the SPUF system using the enrollment interface will receive fake responses from SPUF\textsuperscript{F}. Using the incorrect CRPs to train the model will result in either non-convergence or convergence with highly inaccurate prediction results. Unknowingly using such an incorrect SPUF model to mount an attack on the target device will easily expose the adversary. Comparing with output
nothing or a fixed response to alert the attackers, the false PUF responses waste the attacker’s time and resources to build an incorrect model. This will increase their opportunity cost to attack another target or switch attack strategy for a better chance of success.

The server with SPUF^G model can still successfully authenticate the deployed device as frequently as necessary using the proposed mutual authentication protocol presented in the next subsection.

C. Mutual Authentication

The genuine server can use the mutual authentication protocol shown in Fig. 3 to authenticate the device at any time, and change the current waiting time T_w by changing P_os. The minimum waiting time T_min is determined by T_clk × 2^k, where k ∈ [0, L − 1] is the bit position of P that has been forced into the stuck-at-one state after enrollment and T_clk is the clock period of Q. Therefore, the range of waiting time T_w to apply the next challenge to SPUF^G can be varied from T_min to ≈ 2L−k × T_min by loading the remaining L − 1 bits of the register with a non-zero integer P_os. The maximum offset time T_os that can be added to T_min ranges from 2L−1 × T_clk to 1/2 T_min for k = L − 1 (i.e., the MSB of the register) to (2^L−2) × T_clk = (2^L−2) × T_min if k = 0 (i.e., the LSB of the register). By adjusting T_os along with this bilateral authentication, it becomes more difficult for an adversary to estimate the minimum T_w to collect enough valid responses by brute-force query as the waiting time will escalate with every incorrect attempt.

To initiate a bilateral authentication, the server randomly selects an unused half-length challenge c_s and sends it to the device. Upon receiving c_s, the device uses it as a seed to its PRNG to generate n/2 random sub-challenges (c). The device then clears the counter to ensure that Z = 1 so that SPUF^G or SPUF^F can be selected by S to receive (c). By setting S = 1, these sub-challenges are applied to SPUF^F to produce a half-length challenge c_d to the server. The process is aborted by the server if c_d has been used. Otherwise, the server concatenates c_s and c_d to complete the full-length challenge c_s || c_d and uses it as a seed to the same PRNG as the device to generate 3m sub-challenges (c) to produce three m-bit responses, r_1, r_2 and r_3 from SPUF^G model. Two helper data h_1 = P_os ⊕ r_1 and h_3 = r_1 ⊕ r_3 are computed, where P_os = T_os/T_clk is a (L − 1)-bit positive integer. If m ≥ L, the m-bit response is truncated to match the length of P_os before the xor operation. h_1 and h_3 are concatenated into h_1 || h_3 and sent to the device.

Upon receiving the (m + L)-bit string, using the same sub-challenges (c) with c_s || c_d from its PRNG, the device generates three m-bit responses, r_1, r_2 and r_3, by setting S = 0 to select SPUF^G for the application of (c). Then it computes h_3 = r_1 ⊕ r_3. If h_3 matches the received h_3 within acceptable fractional Hamming distance (FHD) tolerance, the server is authenticated. The device then recovers P_os from the received h_1 by XORing it with r_1.

The recovered P_os is loaded into the register to change the waiting time to T_w = T_min + T_os. The device will acknowledge the successful update by sending h_2 to the server, where h_2 = P_os ⊕ r_2. Otherwise, if the FHD between r_3 and r_3 exceeds the acceptable tolerance, the device sets S = 1 to generate h_2 = SPUF^F((c)) and sends it to the server. Q is cleared by setting clr before transmission of h_2. The server can verify the authenticity of the device by checking the received h_2 against h_2 = P_os ⊕ r_2. If they are equal within an acceptable FHD tolerance, the device is authenticated successfully. Otherwise, the authentication fails and the process is aborted.

It should be noted that the adversary cannot issue an unseen packet h_1 || h_3 that has not been issued by the server to obtain the corresponding response packet h_2. Consequently, the uniqueness of every authentication is assured by the fresh challenge. Because r_1, r_2 and r_3 are generated from the same fresh starting challenge and locked to each other by design, replaying either half challenge will not produce new response. Even if r_1 ≠ r_1 due to the reliability of SPUF^G, the minimum waiting time is still guaranteed by T_min. There is no need for the server to keep track of the waiting time. As long as the error due to the recovered P_os and the response r_3 generated by SPUF^G does not cause the FHD between h_2 against h_2 to exceed τ, the outcome of the authentication will not be affected. Since we do not limit the number of challenges like [12], and there is no waiting time for a genuine server, the server can authenticate the same device again with a fresh challenge before rejecting the device.

III. RESULTS AND ANALYSES

Both covariance matrix adaptation evolution strategies (CMA-ES) and linear regression (LR) attacks are used to evaluate the proposed protocol. We follow the approaches in [15]–[19] to simulate the attacks on Arbitrator PUF (APUF). To test the impact of noise on the proposed deception protocol, in all our experiments, a random variable ∼ N(0, σ^2_noise) is inserted into each delay path, Δ(n), where σ^2_noise = 0.5 is derived from the practical noise level obtained in [20].
A. Effect of False PUF Multiplexing on Modeling Attacks

The fake response generator is implemented by a SPUF design with a different circuit architecture from the genuine SPUF to produce vastly different CRPs. A RNG function from MATLAB is also used to generate random responses for comparison. The resistance of the deception protocol, SPUF\(^{G}\) + SPUF\(^{F}\) are evaluated by both LR and CMA-ES attacks and compared against that of SPUF\(^{G}\) + RNG in this section.

1) Percentage of Valid/Invalid Responses on LR Attack:

Two conventional APUFs, one for SPUF\(^{G}\) and the other for SPUF\(^{F}\), are employed to produce the CRPs for training. Three different sizes of CRP sets, \(N_{\text{CRP}} = 640, N_{\text{CRP}} = 24,000\) (similar to [15], [16]), and \(N_{\text{CRP}} = 240,000\) are used for training. Depending on the percentage of fake information, \(x_f \in \{0, 100\}\), a group of mixed response bits from both SPUF\(^{G}\) and SPUF\(^{F}\)/RNG is collected for the LR attack.

Fig. 4 depicts the LR attack results of the proposed deception protocol by applying different challenge bit lengths, 64-bit and 128-bit, as well as utilizing different fake response generators (RNG and XOR PUF). The number of training samples used for this experiment, \(N_{\text{CRP}} = 4,000\), is the same as that used in [18].

Fig. 5 shows the CMA-ES attack results on the proposed deception protocol by applying different challenge bit lengths, 64-bit and 128-bit, as well as utilizing different fake response generators (RNG and XOR PUF). The number of training samples used for this experiment, \(N_{\text{CRP}} = 4,000\), is the same as that used in [18].

B. Effect of Dynamic and Static Waiting Time

For every authentication attempt made by the adversary before the waiting time \(T_{w}\), the offset time \(T_{os}\) will be doubled.
Hence, the waiting time after applying $N_f$ challenges with waiting time shorter than the current $T_w$ is given by:

$$T_w = (P_{min} + P_{os} + 2P_{os} + \cdots + 2^{N_f}P_{os}) T_{clk} = \{2^{k} + P_{os}(2^{N_f} - 1)\} T_{clk}$$  \hspace{1cm} (1)$$

For $T_{clk} = 1ms$, $L = 32$, $k = 21$ and $P_{os} = 1000$, $T_{min}$ is less than an hour. After only $N_f = 25$, the waiting time $T_w$ exceeds 388 days. If $P_{os}$ is updated by the server to $2^{25}$, it takes only $N_f = 10$ adversarial attempts for the waiting time $T_w$ to exceed 397. The more the attacker attempts to close in their estimate of $T_w$, the faster $T_w$ will blow up to years. The attacker can derive a full set of valid training samples by always waiting for a time longer than $T_w$ before sending the next challenge, provided that the genuine server did not authenticate the device to change the next challenge, provided that the genuine server did not decide to change the waiting time during this period. If the attacker sends a challenge every $t$, where $t > T_w$, the time, $T$, to obtain the training set can be calculated by (2), which includes the time to derive $N_{CRP}$ of training data, the training time $T_{train}$, and the trial-and-error time $T_{tke}$, for determining $T_w$.

$$T = N_{CRP} \times T_w + T_{train} + T_{tke}$$  \hspace{1cm} (2)$$

Omitting $T_{tke}$ and the penalty due to the dynamic component $T_{os}$ (i.e., assume $T_{os} = 0$), Fig. 6 shows that the efficiency of modeling attacks on existing unfortified ($T_w = 0$ hr) 64-stage APUF reduces with different $T_{min}$. Even if we assume the attacker knows $T_{min}$, the overall attack time $T$ without $T_{os}$ can still be made impractically long by fixing $T_{min}$ according to the known number of training samples $N_{CRP}$ for a reasonable accuracy of prediction for modeling the chosen genuine SPUF. For example, a 64-stage APUF design can be predicted with 95% accuracy in approximately 0.01 seconds (the time for sending and reading one CRP is neglected) with a training sample size of $N_{CRP} = 640$ using the above LR based experiment. Then using the proposed deception protocol with $T_{min}$ of 1 hour, it will require 25.2 days for the same $N_{CRP}$ to achieve the same prediction rate. Similarly, the overall attack time $T$ is also longer when more training samples $N_{CRP}$ are needed if a larger APUF with more number of stages or a more complex XOR-PUF is used. The original attack time increases to 0.13 seconds and the prediction rate hits 99% when $N_{CRP}$ is increased to 2,555. By using the same static $T_{min}$ of 1 hour with our protocol, without compromising the success rate of prediction, the overall attack time $T$ will increase to 109 days. If $T_{min}$ is set to 24 hours, the attacker will require approximately 7.1 years to achieve a prediction rate of 99%.

IV. PROTOCOL COMPARISON

The survey in [3] divides the PUF-based authentication protocols into two groups, heavyweight and lightweight. Most of the heavyweight protocols require either a strong cryptographic algorithm for privacy amplification and an ECC for response reconciliation [4]–[8]. For example, the controlled PUF [21] applies hashing to obfuscate its CRPs and requires ECC to correct the noisy responses. A detailed review and comparison of the heavyweight authentication protocols have already been done in [3]. In Table I, we compare the proposed deception protocol against the baseline, three other protocols, slender PUF [9], noise bifurcation [10] and system-of-PUF protocols [22] in the lightweight group, as well as two recently published authentication protocols, lockdown [12] and latent obfuscation [13].

![Figure 6. The overall time, $T$, taken by modeling attacks to predict a 64-bit APUF with respect to the minimum number of training samples $N_{CRP}$ and static $T_{min}$ of the proposed deception protocol. The typical training time, $T_{train}$, is assumed to be 1 second.](image)

<table>
<thead>
<tr>
<th>Protocol</th>
<th>PUF-independent</th>
<th>Modeling resistance</th>
<th>Controlled no. of auth.</th>
<th>TRNG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic authentication [2]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Slender PUF [9]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Noise bifur. [10]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Lockdown Ia [12]</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Lockdown Ib [12]</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Lockdown Iia [12]</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Latent obfus. [13]</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>System-of-PUF [22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Proposed protocol</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

X denotes ‘no’. The symbol ✓ denotes ‘yes’. – denotes result not available.

The evaluation metrics used in the comparison are:
- **PUF-independent** relates to whether or not the protocol requires a specific PUF design.
- **Modeling resistance** refers to the protocol assisted resistance to model building attacks.
- **Controlled no. of auth.** refers to a tight bound on quantity of CRPs that can be authenticated.
- **TRNG** refers to any TRNG component used in these protocols.

As analyzed earlier, the proposed deception protocol demonstrates good robustness to different ML based attacks. Any type of SPUF can be used as the genuine SPUF or fake SPUF. Among the previous works, only lockdown protocols have no restriction on the PUF design but its model-based authentication scheme is designed based on system-
level instantiation of hard-to-learn XOR-PUF. In contrast, our scheme enables secure model-based authentication with generic machine-learnable SPUF. Except lock-down, latent obfuscation and our proposed deception protocols, other authentication protocols listed in Table I are vulnerable to modeling attacks. Typically, a TRNG is used to generate a random substring to hide the response, e.g., in slender PUF [9]. In our work, a temporal control is used to delay the attacker from collecting enough correct CRPs from the real SPUF within a practical time. Overall, the proposed deception protocol has achieved all the desirable properties. There is no tight control on the number of authentications except a minimum time limit is imposed between two legit challenges to an attacker who has no knowledge about the unused CRPs. This time limit is otherwise unrestrictive for the genuine server.

V. CONCLUSION

Deception as a defense strategy provides greater delay, confusion and disruption than rejecting sessions to the attacker’s onslaught. It can drive preventive countermeasures to delay an attack, causing the adversary economic harm to figure out what is real and what is not, and hesitant to proceed. In this paper, we propose a novel deception authentication protocol by deceiving the adversary to use a training set dominated by fake/invalid responses for ML. This will prevent their PUF clone from correctly predicting the response to the unknown challenge. The rapid drop of prediction accuracy with increasing fraction of fake responses is demonstrated using two of the most widely known modeling attack techniques, LR and CMA-ES. The enrolled software model of real SPUF is used by the server to authenticate the device at any time through the proposed deception-based protocol. Each successful mutual authentication initiated by the server also randomly changes the dynamic component of the waiting time between two successive challenges, making it even more disruptive for the adversary to estimate the waiting time.

ACKNOWLEDGMENT

This work was partly supported by the Singapore MOE Tier 2 Grant No. MOE2015-T2-2-013, by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korean government(MSIT) (No. 2016-0-00399, Study on secure key hiding technology for IoT devices [KeyHAS Project]), by the Engineering and Physical Sciences Research Council (EPSRC) (EP/N508664/1-CSIT2), by National Natural Science Foundation China (61771239) and by Nature Science Foundation of Jiangsu Province (BK20151477).

REFERENCES