

Q Article development led by [acmqueue](https://queue.acm.org)
queue.acm.org

The use of silicon PUF circuits.

BY MENG-DAY (MANDEL) YU AND SRINIVAS DEVADAS

Pervasive, Dynamic Authentication of Physical Items

AUTHENTICATION OF PHYSICAL items is an age-old problem.³ Common approaches include the use of bar codes, QR codes, holograms, and RFID tags. Traditional RFID tags and bar codes use a *public identifier* as a means of authenticating. A public identifier, however, is *static*: it is the same each time when queried and can be easily copied by an adversary. Holograms can also be viewed as public identifiers: a knowledgeable verifier knows all the attributes to inspect visually. It is difficult to make hologram-based authentication pervasive; a casual verifier does not know all the attributes to look for. Further, to achieve pervasive authentication, it is useful for the authentication modality to be easy to integrate with modern electronic devices (for example, mobile smartphones) and to be easy for non-experts to use.

Identification is not the same as *authentication*.

A public identifier alone cannot distinguish a genuine product from a counterfeit copy, since a public identifier

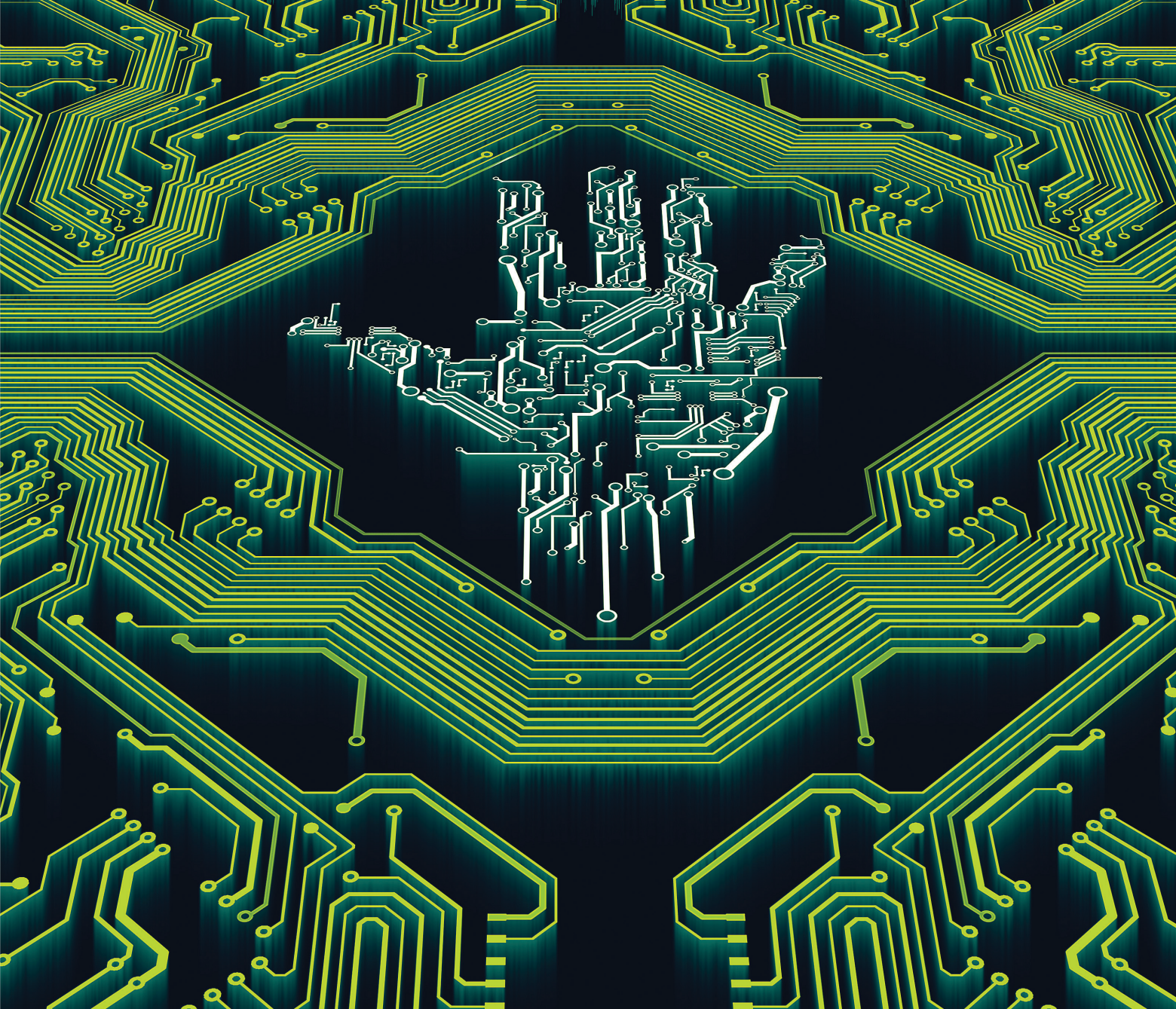
is static and can be openly queried. An adversary can “get ahead” of a legitimate authentication event by querying a genuine product ahead of time, and subsequently replaying the response or making a copy of the identifier.

Attack vectors associated with the inability to distinguish the genuine from a copy are numerous. Consider these two cases:

Physical item counterfeiting. Imagine an authentication system where an authentication server detects the presence of a counterfeit item based on scans associated with its public identifier; any available geolocation and timestamp information is associated with the public identifier upon a scan, and then stored on the authentication server. A counterfeiter can produce products with bar codes or RFID tags that are programmed with a previously seen identifier of a genuine product. If the server is presented with a scan of both a genuine and a counterfeit product, it cannot distinguish one from the other, and can do no better than marking both as suspected counterfeits.

False scan injection. It may be possible, depending on the system design, for an adversary to disrupt the authentication decision ability of the server without ever building a physical counterfeit product. Continuing from the previous example, let's suppose that an adversary is able to electronically submit a scan to the authentication server, with a geolocation that has been spoofed; the scan is purely electronic and does not come from a physical product. The scan contains a public identifier obtained from a genuine product that was sitting in a store; alternatively, a list of product identifiers might have been pilfered from a distribution center. If a genuine product is scanned later, the server may regard the genuine product as a suspected counterfeit since that product identifier has apparently (from the perspective of the server) been in a different geolocation.

Indeed, the ability to distinguish a genuine from a copy is very useful,



if not central, to any effective anti-counterfeiting scheme. The magnetic-stripe-based credit cards that have been widely used in the U.S. suffer from not having this capability (they are being replaced with chip cards for this reason). Risk management analytics trigger the issuance of a new credit card number and a new credit card because the genuine one cannot be distinguished from a copy or clone. A consumer might be in possession of a genuine credit card, but they are forced to replace it because the system cannot distinguish it from a clone.

Broadly speaking, while the use of public identifiers can establish the supply-chain provenance trail, assuming all parties in the supply chain are honest, such a scheme does not pre-

vent *substitution attacks* where genuine products packaged with genuine public identifiers are replaced with counterfeit products packaged with maliciously replicated public identifiers. This can occur at supply-chain checkpoints or while products are in transit. It is desirable for an authentication system to be able to distinguish genuine from clone; address man-in-the-middle, replay, and substitution attacks; and allow multiple parties to cross-audit each other without the need to assume *all* the upstream supply-chain parties have behaved properly.

Dynamic Authentication

One of the main problems with using a public identifier to authenticate is that it is static and subject to replay attacks

when there is a man-in-the-middle adversary between the device and the authentication verification server. In the cryptographic realm, using a cryptographic primitive such as a keyed block cipher or a keyed hash function solves this problem. An authentication verification server generates a random number that can be used as a challenge, and the device's response is a function of the incoming challenge from the server and a secret key that is stored securely on the device. The authentication verification server also needs the secret key in order to verify that the incoming response from the device is correct; the response can be the cipher text of the block cipher or the digest of the keyed hash function. A public identifier (for example, a serial

number) can be used to enable the authentication verification server to look up the correct key for the device being queried. Here, the public identifier is being used for its proper purpose, to identify, and not as the primary means to authenticate. The response cannot be simply replayed to the server by a man-in-the-middle adversary because the server uses a different unpredictable challenge each time.

Cryptographic implementations of a challenge/response protocol require two items on the device:

Keyed cryptographic module. The device needs a cryptographic primitive such as a block cipher or hash function that uses a secret key.

Obfuscated secret key. The device needs a secret key that is securely stored, using, for example, secure non-volatile memory that is obfuscated and not publicly readable. Nonvolatile memory technologies are known to be subject to reverse-engineering attacks, where the secret-key bits that are stored can be recovered.⁹ Layout obfuscation is viewed as important in making key recovery more difficult.

Today, many products do not have dynamic authentication because cryptographic approaches may be too expensive or unusable in a passive circuit setting where energy to power the cryptographic circuit is harvested from an external RF field source (for example, a dedicated RFID reader or an NFC, or near-field communication-enabled, smartphone). A lower-complexity and inexpensive implementation of a challenge/response protocol would allow authentication to become more pervasive, especially if it could be integrated with a modern mobile smartphone in a manner that is easy to use.

There is an alternative method of implementing a challenge/response protocol with integrated measurement capability. The approach requires neither a keyed cryptographic module nor an obfuscated secret key on the silicon device. The idea emerged at MIT more than 10 years ago.⁶ This article discusses what the research community has learned since then in terms of using silicon PUFs (physical unclonable functions) for challenge/response authentication, how PUFs have been deployed to combat counterfeiting, and what some of the open problems are.

Silicon Physical Unclonable Function

Silicon PUF circuits generate output response bits based on a silicon device's *manufacturing variation*. The variation is difficult to control or reproduce since it is within the tolerances of the semiconductor fabrication equipment.⁶ The devices are manufactured identically from the same mask, and there is no secret-key programming to make each device respond differently even to the same challenge. When the same challenge is applied to different devices, each device outputs a different response. When the same challenge is applied repeatedly to the same device, the PUF outputs a response that is unique to the manufacturing instance of the PUF circuit, though some of the response bits may flip from query to query. This is because the response is produced based on a physical (versus a purely algorithmic) evaluation, which is subject to physical evaluation noise that depends on temperature, voltage, and other environmental effects.

PUFs have two broad classes of applications:¹⁴

Authentication. In the authentication use case, the silicon device is deemed authentic if the response from an authentication query is close enough in Hamming distance to a reference response obtained during a provisioning process. This is similar to the false-positive and false-negative behavior found in human biometric systems, where noisy mismatching bits can be “forgiven” using a threshold-based comparison. To prevent replay attacks, challenges are not reused. Early research at MIT showed that identically manufactured circuitry could produce unique challenge/response pairs on different silicon instances of the same circuit, and it was argued that for any given device, the response is difficult to predict when subject to a random challenge.

Key generation. If, instead of a threshold-based authentication, the PUF is to serve as a secret-key generator, only a fixed number of response bits need to be generated from the PUF. These bits can serve as symmetric key bits and can be used in a secure processor.¹⁵ Since cryptographic keys are required to be bit exact, the basic PUF circuit needs to be enhanced with error-correction logic, which

increases implementation complexity. There are security considerations in the exposure of error-correcting bits that need to be addressed in any PUF error-correction scheme.⁶ These considerations have been addressed in the past ten years, leading to commercial products that use PUFs to provide key generation capability enabled through manufacturing variation. For example, a PUF was integrated into an ARM-based SoC (system-on-a-chip) from Xilinx¹⁹ for secure firmware load during the device boot process.

The focus here is on the authentication use case. The goal is to bring authentication to applications where conventional cryptographic approaches are too expensive and cumbersome, and enable pervasive dynamic challenge/response authentication of physical items.

Differential Measurements

To make silicon PUFs viable, one main obstacle to overcome is preventing changes in environmental conditions (for example, temperature, voltage) from overwhelming minuscule manufacturing-variation-induced measurements. One of the key insights in early PUF research⁶ was to use differential measurements, which was later proven to be highly effective in silicon. Before then, the characterization of manufacturing variation required expensive semiconductor test equipment and a lot of evaluation time, and it was not obvious how to do so very quickly in an *in-circuit* fashion (without expensive external equipment) and in a manner that is robust to environmental changes. To the extent that temperature, voltage, and other environmental effects impact the differential measurements equally, their effects cancel out, thereby allowing the minute manufacturing-variation-induced effects to be manifested in the PUF response measurements. This is a general principle that was successfully employed in many subsequent silicon PUF circuits. A survey of different PUF approaches as they relate to authentication was published in 2015.⁴

The first custom silicon PUF implementation from MIT was the arbiter PUF,⁷ shown in the dash-lined box in Figure 1. This is referred to as a “basic” arbiter PUF building block to distinguish it from more complex constructs to be

discussed later. The PUF output for each basic arbiter PUF is derived from a differential race condition formed by successive delay stages. Each stage consists of a crossbar switch that can be formed using 2:1 multiplexers. A challenge bit for each of the $n = 64$ stages determines whether the parallel path or the cross path is active. Collectively, n challenge bits determine which path is chosen to create the transition at the top input to the arbiter latch, and similarly for the bottom input. The arbiter latch is formed using a pair of NAND gates that is cross-coupled (this is denoted by the rectangular boxes marked A in Figure 1). The difference comparison between the two delay paths, configured by the challenge bits, determines whether the basic arbiter PUF produces a 1 or a 0 output bit. The layout of the design has to be symmetrically matched so that random manufacturing variation affects the response.

To obtain a multibit challenge, a seed challenge is applied to a challenge expansion circuit—for example, an LFSR (linear-feedback shift register), which is not shown in the figure. The LFSR is used to produce the subchallenge bits $\langle c \rangle$. The subchallenge bits are used to generate a response bit. Using multiple subchallenges and concatenating the resulting response bits together forms a multibit response.

Because of the linear and additive nature of the response evaluation, it was recognized early on that learning attacks can be employed to mathematically model a basic arbiter PUF.⁷ Various techniques, including creating multiple instances of the basic PUF and bitwise-XORing their output bits,¹⁴ served as countermeasures to make learning attacks more difficult. Figure 1 depicts four basic 64-stage PUF instances whose output can be XORed together in both a parallel and a serial fashion. An XOR PUF is formed by instantiating multiple copies and XORing the output bits.

Model Building to Aid Authentication

For many commercial applications, more than just a few challenge/response pairs are needed. With the original PUF authentication scheme, the number of challenge/response pairs grows linearly with the number of supported authentication events. If the

number of supported events is relatively large—say, 1,000 authentications or more—this would result in many challenge/response pairs needing to be collected as part of the provisioning process and then stored on the server; both the provisioning time and the server storage would grow linearly with the number of authentications supported.

A major development in PUF research was using the ease of model building of the basic arbiter PUF (prior to the XOR countermeasure) to create an *authentication verification model* on the server side. This authentication verification model essentially serves as a symmetric counterpart to the physical PUF circuit on the device. Therefore, instead of collecting a number of challenge/response pairs that are linear in the number of authentication events and requiring a linear amount of storage on the server side, there is now a constant-size storage per PUF device on the server side regardless of the number of authentication events.

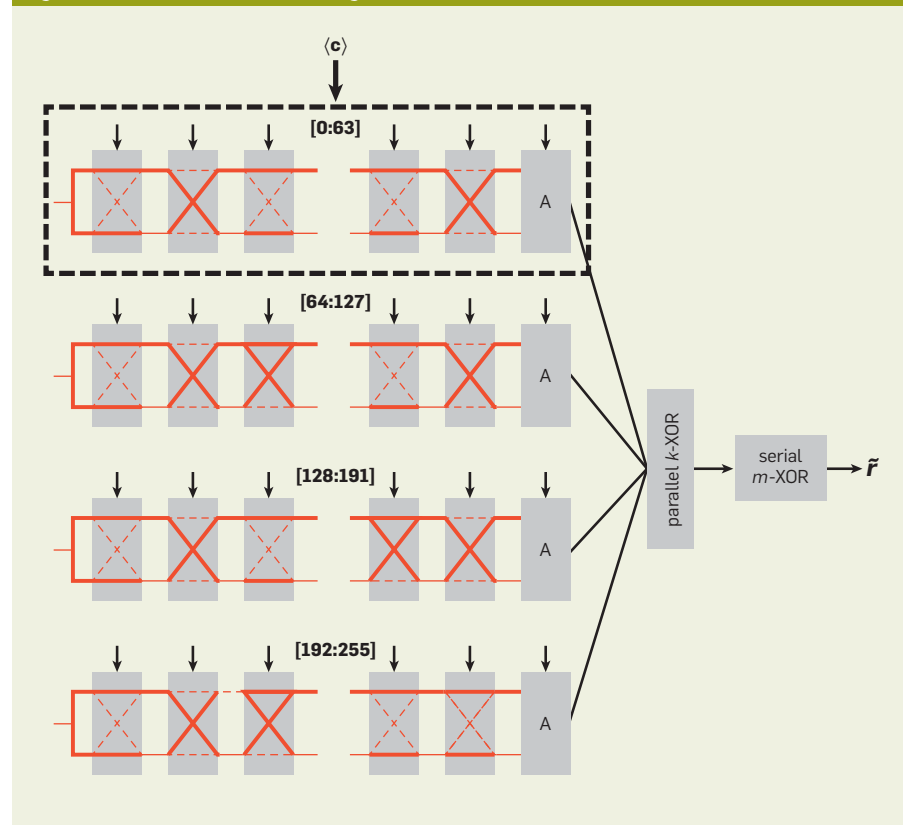
While the basic arbiter PUF can be learned with relative ease, the XORing produces output bits that are more difficult to learn. One can take advantage of this by making the easier-to-learn

variant available during the provisioning process (that is, bypassing the XORs). Here, the pre-XOR response bits for each basic arbiter PUF are obtained, and a state-of-the-art machine-learning algorithm can be used to derive the delay values on the server side. Afterward, the XORs are no longer bypassed, increasing the machine-learning difficulty for the adversary. To get the benefit of constant provisioning time and the constant storage requirement, the provisioning functionality needs to be disabled after the device leaves the manufacturing facility. For example, the publicly readable serial number of the device, once programmed, can disable the extraction of the PUF response bits prior to the XOR countermeasure. Reprogramming of the serial number is also disabled.

Machine Learning Attacks

When lightweight PUF-based authentication is performed using a threshold-based comparison as described previously, neither a cryptographic algorithm nor an obfuscated key is required on the silicon PUF device. Unfortunately, without a cryptographic algorithm and an obfuscated secret or

Figure 1. Basic arbiter PUF building block shown in the dotted line.



private key, it is difficult to derive an exponential number of challenge/response pairs from a linearly sized PUF circuit. Arbitrary logical or arithmetic post-processing cannot be applied to the silicon manufacturing variation since the physical PUF evaluation noise would be amplified. Therefore, popular PUF authentication circuits are evaluated in a mostly linear fashion, with limited nonlinear mixing (for example, using XORs as described earlier), and are therefore prone to modeling attacks.

Machine-learning attacks have been applied successfully on a variety of PUF constructs using computer-simulated PUF models¹¹ and, later, for silicon PUF implementations.¹² These attacks include the popular XOR construction of the arbiter PUF and serve as a benchmark for a PUF's machine-learning attack attributes as well as a catalyst for the development of countermeasures.

Another breakthrough attack was presented in 2015, where PUF response "reliability" information, which can be viewed as noise side-channel information, is used essentially to bypass the nonlinear mixing effects of the XORs, making even a PUF with a very high (for example, 20-plus) number of XORs relatively easy to learn (for example, using a few hundred thousand response bits).¹

Probabilistic Authentication

To thwart the attacks described in the previous section, machine-learning attack countermeasures were developed, taking advantage of the model-building concept. With it, challenges can now be determined *at runtime*. Recall that the authentication verification model that is stored on the server side can be used to synthesize *any* challenge/response pairs; the server is no longer restricted to the challenge/response pairs collected during the provisioning process. The device can now produce part of the challenge at runtime, so neither the server nor the device alone can fully determine the exact subchallenges used. This also makes the authentication process probabilistic: even if a (malicious) server repeatedly issues the same challenge C_s to the device, any output response R would be a function of both C_s and C_d . Here, C_d represents the device-generated part

of the challenge, which is passed back to the server. Instead of $R = \text{PUF}_{id}(C_s)$, now it is $R = \text{PUF}_{id}(C_s || C_d)$. As a result, the PUF cannot be trivially distinguished from random by repeating the same challenge C_s , analogous to what happens when probabilistic encryption is used in the cryptographic realm.

The use of device-generated challenges^{8,21} can be viewed as a countermeasure to prevent repeated challenges, which addresses the reliability-based machine-learning attacks that take advantage of noise-side-channel information.¹ This also addresses the noise-filtering approach using majority voting employed to attack silicon PUFs.¹²

Theoretical Learning Difficulty

Recently published research⁵ establishes the theoretical difficulty of PUF learning—in particular, for the popular XOR PUF construct. These recent results used the celebrated PAC (probably-approximately correct) framework,¹⁶ which links learning with complexity theory, and applied that framework to determine which kinds of PUFs are polynomially learnable and which would require an exponential attack resource (for example, attack runtime, number of response bits) with respect to the circuit size. The authors of this research⁵ not only declared certain XOR PUF constructs to be exponentially difficult to learn under the PAC framework, but also questioned whether such PUFs can be realized in practice. A new protocol-level countermeasure²⁰ allows exponentially difficult-to-learn PUFs based on the PAC results⁵ to be instantiated in practice, with silicon results to demonstrate practical feasibility. The main idea is to run the authentication protocol in *both* directions; only after the PUF device has authenticated the authentication verification server does the PUF device release new response bits to a potential man-in-the-middle adversary. The device is effectively locked down, with the exposure of *new* response bits implicitly controlled by the server at the protocol level.

To address noise-side-channel attacks, a device-side challenge^{8,21} can be added. The challenge/response behavior of the device is locked down at the protocol level, and the adversary is faced with machine learning using

limited data; meanwhile, the theoretical learning results from the previously mentioned research⁵ imply that both an exponential number of response bits and an exponential attack time are required. The response-data exposure can be throttled dynamically by the server as knowledge of new attacks emerges, even against an adaptive chosen-challenge adversary with uninterrupted interface access to the device, a result of the use of mutual authentication.

So Where Are We?

While a lot of strides have been made in terms of silicon PUF constructs and an understanding of their behavior in terms of practical and theoretical learnability when used in a challenge/response context, some questions remain. While the recent theoretical learning results declared that certain PUF constructs are exponentially difficult to PAC-learn, there is still a reliance on heuristic results for what the exponential learning-difficulty curve looks like for the "best attacks" available. Although there have been several years of machine-learning attacks on PUFs by multiple research groups thus far, there is still work to be done in this area to affirm a practical and safe heuristic limit in terms of number of response bits that can be exposed for different XOR PUF constructs that ensures security. Fortunately, the lockdown approach²⁰ allows the server to manage the response-bits exposure at the protocol level, thereby allowing a degree of dynamic adjustment to emerging attack results.

Additionally, side-channel attacks coupled with machine learning represent a relatively new research area. While the lockdown protocol²⁰ addressed various side-channel attacks that have been published, including noise-side-channel attacks, noise-filtering attacks, and backside photonics imaging attacks, new attacks may emerge that could bring forth research into new countermeasures and new PUF constructs.

PUF NFC IC and Tags

For a silicon PUF to be useful for authentication, it must be integrated into a form that can be easily authenticated. With the recent emergence of NFC-enabled smartphones, custom RFID/

NFC readers are no longer necessary for many use cases. An Android smartphone with a downloaded app would put the power of authentication in the hands of the consumer, or allow a store owner, distributor, or others in the supply chain to perform authentication of products. A product-authentication kiosk (similar to the barcode-scanning kiosks found in many major U.S. retailers) can also be used to provide PUF NFC authentication results as well as pulling information from a cloud server about a particular product instance's origin, source of manufacturing, freshness/expiration, provenance, and other information.

Major progress has been made in terms of PUF packaging and form factor. The first silicon PUF circuit was a relatively large research lab prototype and required wired connections to a computer for authentication, as shown in Figure 2.

After a decade of iteration and refinement, PUF NFC tags appeared in commercial products. Figure 3 shows a PUF NFC tag on a Canon camera package sold in Asia and an Android off-the-shelf NFC device that can be used to authenticate the tag.

Figure 4 is a close-up of a PUF-NFC IC encapsulated in a tag inlay. The area is taken up mostly by the antenna, and the actual IC area is extremely small (shown by the arrow). The antenna size affects the read range. The tag shown has a read range of about five centimeters. A small read range is useful for applications where privacy is an issue or for item-level tagging applications where it is desirable to know that a particular item is being interrogated using an NFC scan, which can be done with a modern NFC-enabled smartphone.

The lightweight nature of the PUF-NFC implementation also brings dynamic authentication capabilities to new product types—for example, secure paper, as shown in Figure 5. This is a useful means of tracking the processing of official documents (for example, when submitted by a private citizen to a government office for processing) and of authenticating them. An NFC scan made by a government employee authenticates the document; the authentication verification server can also record the geolocation and timestamp associated with the authen-

ticated document. This allows a citizen or a government audit agency to more easily track which processing step the document is undergoing as well as the whereabouts of the document.

When applied to an ID card, the PUF NFC approach not only allows a public employee to authenticate the identity of a private citizen, but also allows a private citizen to make sure a person who claims to be a public employee is not a fraud (for example, a public employee visiting a private citizen's house to perform inspection and possible repairs). An NFC scan with a smartphone would authenticate the employee's ID card, and an image of the card could be accessed on the homeowner's smartphone to make sure the picture and other vital information on the card have not been altered. A work order associated with the task that the visiting public employee is authorized to perform can also be displayed. Pervasive authentication by both the public employee and the private citizen would promote better public-sector accountability.

Previously, RFID scans required dedicated readers, which may be feasible to distribute to public employees or in other enterprise settings, but would be

Figure 2. MIT silicon PUF prototype, 2002.

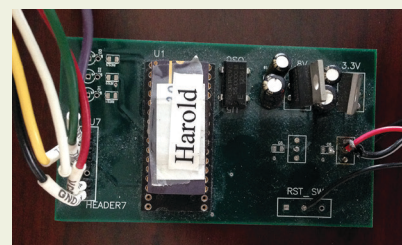


Figure 3. Commercial deployment, 2014.



cumbersome if not cost-prohibitive to distribute to private citizens. A modern NFC-enabled smartphone, when used with a PUF NFC tag, democratizes authentication, putting the power of authentication in the hands of a private

Figure 4. PUF NFC tag.

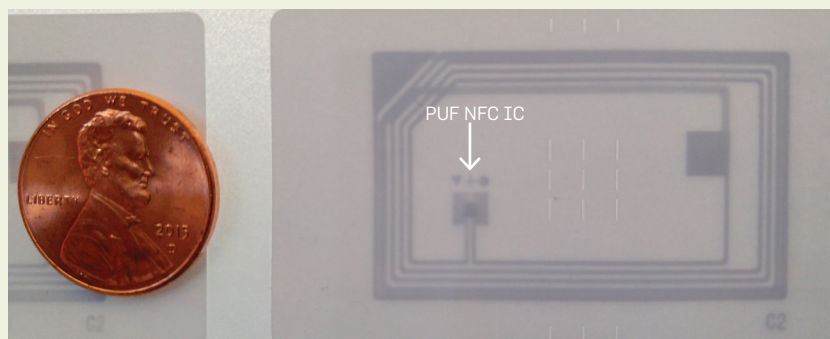
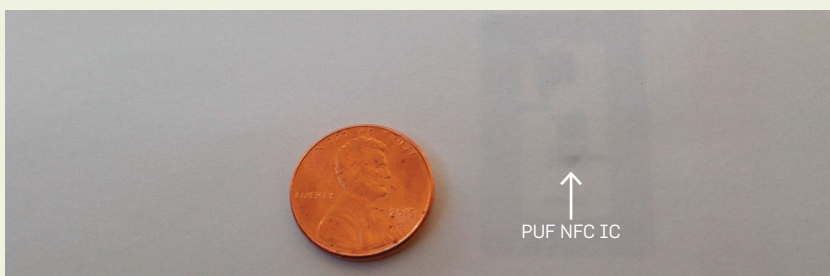


Figure 5. PUF embedded in secure paper.



individual without the burden of specialized reader hardware distribution.


Identification, Authentication, Authorization

In 2004, Bruce Schneier wrote about the importance of distinguishing three inter-related security services: identification, authentication, and authorization.¹³ While we have discussed PUFs mostly in the context of item-level authentication, they can also be used to provide each of the three security services.


Identification. As described by Schneier, an identifier needs to distinguish one member of a population from another member. Schneier also stated that conventional human biometric measurements such as fingerprint scans or iris scans cannot be used for identification; a separate identifier is needed so the biometric reading can be matched against a single reference biometric template vs. all the templates for the population. This is because, for a human biometric reading, if a match is performed across all templates in a population, the collision probability is too high (for example, on the order of 1 in 10,000 or 1 in 100,000^{10,18}). This means that out of a reasonably sized population larger than a small city, there is a high probability that two biometric readings would be regarded as coming from the same individual if a separate identifier were not used. Human biometrics can be used for authentication *if a person has already been identified through other means.*

When a silicon PUF is used, the collision probability can be made well below those for human biometrics—for example, it can be made below 1 in 1 trillion *without* the use of a separate public identifier. A silicon PUF implementation can scale the uniqueness information content better than a human biometric scheme, allowing the former to be used for identification.

Although the NFC PUF IC implementation described earlier uses a serial number to identify the device, if a challenge/response PUF authentication occurs, it is feasible to use the PUF to *identify* the device. A preselected challenge, possibly hardwired into the chip, can have the corresponding preselected response designated as an identifier. When different devices are queried, each device outputs a unique preselected response that is possibly noisy. The



A modern NFC-enabled smartphone, when used with a PUF NFC tag, democratizes authentication, putting the power of authentication in the hands of a private individual without the burden of specialized reader hardware distribution.



server performs a fuzzy match across a reference set of preselected responses (collected during provisioning) for the entire population to determine which device is being accessed. Then the server can fetch the corresponding authentication verification model for that device, to be used for the authentication phase. The nonvolatile storage bits on the chip that would otherwise be used to store the serial number to identify the device can be eliminated.

In certain use cases, where the RFID/NFC is used for identification only, and no on-chip data storage is needed to store ancillary data associated with a tagged product, it may be possible to eliminate all nonvolatile storage from an RFID/NFC device by using a PUF to provide the identification. Eliminating on-chip nonvolatile storage is a potential source for savings in terms of silicon area and manufacturing cost.

Authentication. Item-level authentication is an obvious use case for the challenge/response silicon PUF. This is a case of *server-to-device entity authentication*. As mentioned previously, it is also possible to run the entity authentication protocol in the reverse direction, for *device-to-server entity authentication*. In the lockdown protocol scenario,²⁰ running the entity authentication in both directions is used to limit response-bits exposure.

It is possible to extend the aforementioned mutual *entity* authentication functionality to perform *data* authentication and, in particular, to authenticate a relatively small number of data bytes. This protocol extension provides *server-to-device data authentication* as well as *device-to-server data authentication*. For example, a read/write interface can be implemented between the server and the device, so only authenticated read/write commands from the server are acted upon by the device. If the server read/write commands are modified in transit, the device could detect the bit modifications.

This can be achieved by incorporating the data bytes as a part of the challenge. The server first receives the serial number (id) from the device as well as a device-side challenge C_d . Then it sends to the device a server-side challenge C_s , command bytes B , and a response R , where $R = \text{PUF}_{id}(C_s \parallel C_d \parallel B)$ emulated using the server-side model. The

device validates the response; since the challenge now incorporates the command bytes B , the read/write command is also authenticated. The device authenticates the server and the incoming command before executing the latter. These might be commands that allow certain configuration bits to be written into the device's on-chip memory, or certain data to be read. The device can send back data in a similar manner. The server can then authenticate both the source of the data (entity authentication) and that the data from the device hasn't been modified in transit (data authentication).

Authorization. In many applications, the verifier is a public employee who obtains access to a private person's database entry from a cloud server in order to perform a more comprehensive authentication of an individual. The sensitive information may include security questions or other personal information that can be verbally validated. A private person's PUF-NFC ID card can be used to limit database access by a public employee so that such an employee cannot arbitrarily pilfer sensitive private data. The employee is *authorized* to obtain database access to certain sensitive and personal information only when a particular PUF-NFC ID card from a private person is physically present and produces a proper response to a server's challenge.

In many of today's RFID use cases, it is also common to store certain information associated with a tagged product on the RFID device itself, which incurs a silicon area overhead and an increase in manufacturing cost associated with larger on-chip nonvolatile storage. Since a conventional RFID device does not offer dynamic authentication (it emits only a static public identifier), the locally stored information cannot be safely moved into the cloud; this is because another RFID that is programmed with the same serial number would be associated with that data record in the cloud. For example, the data record can be the maintenance trail of an airplane part or the supply-chain provenance trail of a pharmaceutical product. If the RFID/NFC device, however, is used to offer authorization (for reading or both read/write) to access a particular database entry in the cloud, then the data that otherwise would be

stored locally on the RFID device can be more safely moved into the cloud. This minimizes the need for large storage local to the RFID/NFC device. An individual on the ground is authorized to access the cloud data record only when the PUF-NFC device is physically present. This assumes that reader devices are cloud-connected, which is increasingly becoming the trend.

Also, in the age of big data and cloud computing, data is worth more when it is aggregated in the cloud vs. stored separately in each tag. In the former case, analytics can be performed to uncover unauthorized activities or to gather other forms of business intelligence information. The PUF serves to bind the tag to a particular database record in the cloud by providing access authorization in a manner that a static public identifier cannot.

Conclusion

For more than a decade, silicon PUFs have gathered an enormous amount of interest in applications ranging from product authentication to secure processors. There have been commercial deployments and complete integration with authentication servers and consumer-grade, off-the-shelf smartphones to give the power of authentication to the ordinary person.

People know much more about PUFs and how to use them, including vulnerabilities and countermeasures, than they did a few years ago. As the PUF field becomes well established, more attacks and countermeasures are expected to be published to further vet the security properties of PUFs. Such a cycle has also been seen in the cryptographic world—for example, the AES-ECB algorithm was subject to the “Penguin” attack,¹⁷ and the plain RSA algorithm is subject to existential forgery,² both of which can be addressed by using the fundamental primitives in a different fashion. **C**

Related articles on queue.acm.org

A Threat Analysis of RFID Passports

Alan Ramos et al.

<http://queue.acm.org/detail.cfm?id=1626175>

The NSA and Snowden: Securing the All-Seeing Eye

Bob Toxen

<http://queue.acm.org/detail.cfm?id=2612261>

References

1. Becker, G. The gap between promise and reality: On the insecurity of XOR arbiter PUFs. *International Workshop on Cryptographic Hardware and Embedded Systems* (2015), 535–555.
2. Boneh, D., Joux, A. and Nguyen, P. Why textbook elgamal and RSA encryption are insecure. *Advances in Cryptology* (2000), 30–43.
3. Counterfeiting and piracy: stamping it out. *The Economist*. April 23, 2016.
4. Delvaux, J., Peeters, R., Gu, D. and Verbauwhe, I. A survey on entity authentication with strong PUFs. *ACM Computing Surveys* 48, 2 (2015), 26:1–26:42.
5. Ganji, F., Tajik, S. and Seifert, J.-P. Why attackers win: on the learnability of XOR arbiter PUFs. *International Conference on Trust and Trustworthy Computing* (2015), 22–39.
6. Gassend, B., Clarke, D., van Dijk, M. and Devadas, S. Silicon physical random functions. *ACM Conference on Computer and Communication Security* (2002).
7. Lim, D. Extracting secret keys from integrated circuits. Master's thesis, MIT, 2004.
8. Majzoubi, M., Rostami, M., Koushanfar, F., Wallach, D. and Devadas, S. SlenderPUF: A lightweight, robust and secure strong PUF by substrating matching. *IEEE International Workshop on Trustworthy Embedded Devices* (2012).
9. Quadir, S. E., Chen, J., Forte, D., Asadizanjani, N., Shahbazmohamadi, S., Wang, L., Chandy, J. and Tehranipoor, M. A survey on chip-to-system reverse engineering. *ACM Journal on Emerging Technologies in Computing Systems* 13, 1 (2016).
10. Quinn, G. and Grother, P. IREX III: Supplement I: Failure Analysis. NIST Interagency Report 7853 (2012).
11. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S. and Schmidhuber, J. Modeling attacks on physical unclonable functions. *ACM Conference on Computer and Communication Security* (2010).
12. Rührmair, U., Sölter, J., Sehnke, F., Xu, X., Mahmoud, A., Stoyanova, V., Dror, G., Schmidhuber, J., Burleson, W. and Devadas, S. PUF modeling attacks on simulated and silicon data. *IEEE Transactions on Information Forensics and Security* 8, 11 (2013), 1876–1891.
13. Schneider, B. Sensible authentication. *ACM Queue* 1, 10 (2004): 74–78.
14. Suh, G.E. and Devadas, S. Physical unclonable functions for device authentication and secret key generation. *Design Automation Conference* (2007), 9–14.
15. Suh, G.E. AEGIS: A single-chip secure processor. Ph.D. thesis. Electrical Engineering and Computer Science Dept., MIT, 2005.
16. Valiant, L. A theory of the learnable. *Commun. ACM* 27, 11 (1984), 1134–1142.
17. Valsorda, F. The ECB penguin, 2013; <https://blog.filippo.io/the-ecb-penguin/>.
18. Wilson, C., Hicklin, R., Bone, M., Korves, H., Grother, P., Ulery, B., Micheals, R., Zoepfl, M., Otto, S. and Watson, C. Fingerprint vendor technology evaluation 2003: summary of results and analysis report. NIST Internal Report 7123 (2004).
19. Xilinx Inc. Xilinx addresses rigorous security demands at 5th Annual Working Group for Broad Range of Applications, 2016; <http://www.prnewswire.com/news-releases/xilinx-addresses-rigorous-security-demands-at-fifth-annual-working-group-for-broad-range-of-applications-300351291.html>.
20. Yu, M., Hiller, M., Delvaux, J., Sowell, R., Devadas, S. and Verbauwhe, I. A lockdown technique to prevent machine learning on PUFs for lightweight authentication. *IEEE Transactions on Multi-Scale Computing Systems* 2, 3 (2016): 146–159.
21. Yu, M., M'Raihi, D., Verbauwhe, I. and Devadas, S. A noise bifurcation architecture for linear additive physical functions. *IEEE International Symposium on Hardware Oriented Security and Trust* (2014), 124–129.

Meng-Day (Mandel) Yu is the chief scientist at Verayo Inc., a research affiliate for CSAAIL/MIT, and is pursuing a Ph.D. based on a research career with COSIC/KU Leuven. He was manager of R&D engineering at TSI and developed a secure digital baseband radio.

Srinivas Devadas is the Webster professor of electrical engineering and computer science at MIT, where he has been since 1988. He served as associate head of EECS from 2005 to 2011. He is a Fellow of the ACM and IEEE.

Copyright held by owner(s)/authors.
Publication rights licensed to ACM. \$15.00.