The Shift PUF: Technique for Squaring the Machine Learning Complexity of Arbiter-based PUFs

Yi Tang [1] Donghang Wu [2] Yongzhi Cao [2] Marian Margraf [3]

[1]University of Michigan

[2]Peking University

[3]Freie Universität Berlin

CASES 2020 WiP

Physically Unclonable Functions (PUFs)

- Hardware cryptographic identification primitive;
- Exploiting inevitable manufacturing variations;
- Hence "physically unclonable."

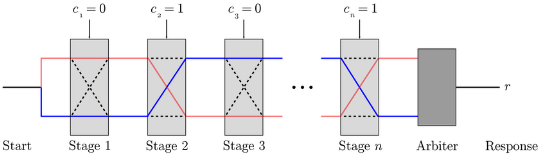
- A PUF Instance is identified by its behavior as a (probabilistic) mapping from inputs (*challenges*) to outputs (*responses*).

Physically Unclonable Functions (PUFs)

- Hardware cryptographic identification primitive;
- Exploiting inevitable manufacturing variations;
- Hence “physically unclonable.”

- A PUF Instance is identified by its behavior as a (probabilistic) mapping from inputs (*challenges*) to outputs (*responses*).

Arbiter PUFs (APUFs)

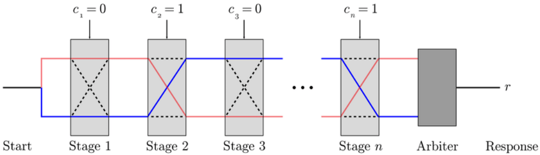


- [1] Challenge decides the paths: either parallel or crossing.
- Two signals are triggered simultaneously and propagate along the decided paths.
- *Arbiter* judges the race and yields the result as response.

- The inevitable manufacturing variations of the signal delays lead to unique challenge-response behaviors.

[1] Figure from *Georg T. Becker, "The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs"*, *2015*.

Arbiter PUFs (APUFs)



- [1] Challenge decides the paths: either parallel or crossing.
- Two signals are triggered simultaneously and propagate along the decided paths.
- *Arbiter* judges the race and yields the result as response.

- The inevitable manufacturing variations of the signal delays lead to unique challenge-response behaviors.

[1] Figure from *Georg T. Becker, "The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs", 2015.*

Desired Properties of a PUF design

- Being *lightweight*: low time and circuit complexities of the hardware;
- Being *strong*: huge challenge space, thus huge PUF instance space;
- *Reliability*: the same challenge results in the same response w.h.p.;
- *Security*: hard to predict with high accuracy the challenge-response behavior given reasonable amount of information.

Security of APUFs

- Lightweight, strong, and reliable.
- However insecure. Challenge-response behavior $r(\mathbf{c})$ of m-bit APUF:

$$r(\mathbf{c}) = [\Delta(\mathbf{c}) \geq 0] \; , \quad \Delta(\mathbf{c}) = \mathbf{w}^\top \mathbf{p} \; ,$$

  $\mathbf{w}$: $(m+1)$-d vector that is only a function of the signal delays in the APUF instance,
  $\mathbf{p}$ (*parity*): $(m+1)$-d vector that is only a function of the challenge $\mathbf{c}$.
- Linear classification model; easy to learn given reasonable number of *challenge-response pairs* (CRPs).

Security of APUFs

- Lightweight, strong, and reliable.
- However insecure. Challenge-response behavior $r(\mathbf{c})$ of m-bit APUF:

$$r(\mathbf{c}) = [\Delta(\mathbf{c}) \geq 0] , \quad \Delta(\mathbf{c}) = \mathbf{w}^\top \mathbf{p} ,$$

  w: $(m+1)$-d vector that is only a function of the signal delays in the APUF instance,
  p (*parity*): $(m+1)$-d vector that is only a function of the challenge **c**.
- Linear classification model; easy to learn given reasonable number of *challenge-response pairs* (CRPs).

Arbiter-based PUFs: XOR APUFs

k-XOR APUF:

- XOR-sum of *k* APUF instances (sharing the challenge).
- #CRPs required in learning is believed to be exponential in *k*.
- Empirically vulnerable to *reliability-based attacks*, where the *reliability* information (probability of getting the same response) is accessible besides merely the response.

Arbiter-based PUFs: XOR APUFs

k-XOR APUF:

- XOR-sum of k APUF instances (sharing the challenge).
- #CRPs required in learning is believed to be exponential in k.
- Empirically vulnerable to *reliability-based attacks*, where the *reliability* information (probability of getting the same response) is accessible besides merely the response.

Arbiter-based PUFs: XOR APUFs

k-XOR APUF:

- XOR-sum of k APUF instances (sharing the challenge).
- #CRPs required in learning is believed to be exponential in k.
- Empirically vulnerable to *reliability-based attacks*, where the *reliability* information (probability of getting the same response) is accessible besides merely the response.

Arbiter-based PUFs: Interpose PUFs (iPUFs)

(x, y)-iPUF:

- Use two XOR APUFs as follows:

$$r(\mathbf{c}) = r_2(\mathbf{c}_1, r_1(\mathbf{c}), \mathbf{c}_2) \ ,$$

  r_1 and r_2: respectively an x-XOR APUF and a y-XOR APUF, $(\mathbf{c}_1, \mathbf{c}_2)$: some fixed split of $\mathbf{c}$.

- As secure as $(x/2 + y)$-XOR APUF, while moreover resilient to reliability-based attacks.

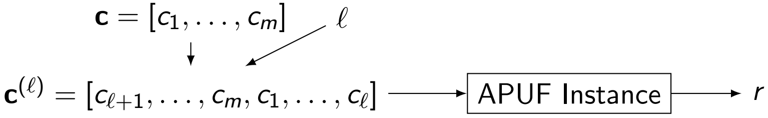
Arbiter-based PUFs: Interpose PUFs (iPUFs)

(x, y)-iPUF:

- Use two XOR APUFs as follows:

$$r(\mathbf{c}) = r_2(\mathbf{c}_1, r_1(\mathbf{c}), \mathbf{c}_2) \ ,$$

  r_1 and r_2: respectively an x-XOR APUF and a y-XOR APUF, $(\mathbf{c}_1, \mathbf{c}_2)$: some fixed split of $\mathbf{c}$.
- As secure as $(x/2 + y)$-XOR APUF, while moreover resilient to reliability-based attacks.



- Prepend to APUF a (w.l.o.g., left) *circular shift* operation.
- Model of challenge-response behavior $r(\mathbf{c})$ becomes:

$$r(\mathbf{c}) = [\Delta_{\text{Shift}}(\mathbf{c}) \geq 0] \ , \quad \Delta_{\text{Shift}}(\mathbf{c}) = \Delta(\mathbf{c}^{(\ell)}) \ ,$$

$\mathbf{c}^{(\ell)}$: the circular shift of $\mathbf{c}$ by ℓ bits.

Shift Displacement ℓ in Shift PUFs

- If ℓ is known by attacker, then the attacker could easily preprocess out the effect of the circular shift.
- Recall: securely generating some secret ℓ is exactly among the applications of PUFs.
- Could use *PUF-based key generation* with the underlying APUF instance to securely generate ℓ;
- ℓ is only $\log m$ bits long; will not harm the efficiency badly.

Shift Displacement ℓ in Shift PUFs

- If ℓ is known by attacker, then the attacker could easily preprocess out the effect of the circular shift.
- Recall: securely generating some secret ℓ is exactly among the applications of PUFs.
- Could use *PUF-based key generation* with the underlying APUF instance to securely generate ℓ;
- ℓ is only $\log m$ bits long; will not harm the efficiency badly.

Shift Displacement ℓ in Shift PUFs

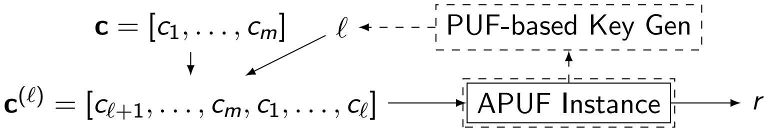
- If ℓ is known by attacker, then the attacker could easily preprocess out the effect of the circular shift.
- Recall: securely generating some secret ℓ is exactly among the applications of PUFs.
- Could use *PUF-based key generation* with the underlying APUF instance to securely generate ℓ;
- ℓ is only $\log m$ bits long; will not harm the efficiency badly.

Shift Displacement ℓ in Shift PUFs

- If ℓ is known by attacker, then the attacker could easily preprocess out the effect of the circular shift.
- Recall: securely generating some secret ℓ is exactly among the applications of PUFs.
- Could use *PUF-based key generation* with the underlying APUF instance to securely generate ℓ;
- ℓ is only $\log m$ bits long; will not harm the efficiency badly.

Shift Displacement ℓ in Shift PUFs

- If ℓ is known by attacker, then the attacker could easily preprocess out the effect of the circular shift.
- Recall: securely generating some secret ℓ is exactly among the applications of PUFs.
- Could use *PUF-based key generation* with the underlying APUF instance to securely generate ℓ;
- ℓ is only $\log m$ bits long; will not harm the efficiency badly.



Security of Shift PUFs

- Linear classification model w.r.t. $\mathbf{p}^{(\ell)}$, the parity of $\mathbf{c}^{(\ell)}$, instead of $\mathbf{p}$.
- However ℓ is unknown to the attacker; cannot apply linear methods.
- Eliminate the effect of ℓ by enumerating all $\ell = 0, 1, \ldots, m-1$.
- Natural and general approach: a $\Theta(m^2)$-d linear classification model.

$$\begin{bmatrix} p_1 & p_1 & p_1 & \cdots & p_1 & p_1 \\ p_2 & p_1 p_2 p_3 & p_1 p_3 p_4 & \cdots & p_1 p_{m-1} p_m & p_m p_1 \\ p_3 & p_1 p_2 p_4 & p_1 p_3 p_5 & \iddots & p_{m-1} p_1 & p_m p_2 \\ \vdots & \vdots & \iddots & \iddots & \vdots & \vdots \\ p_{m-1} & p_1 p_2 p_m & p_3 p_1 & \cdots & p_{m-1} p_{m-3} & p_m p_{m-2} \\ p_m & p_2 p_1 & p_3 p_2 & \cdots & p_{m-1} p_{m-2} & p_m p_{m-1} \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

- Conjecture: the attacker cannot do better than this approach.

Security of Shift PUFs

- Linear classification model w.r.t. $\mathbf{p}^{(\ell)}$, the parity of $\mathbf{c}^{(\ell)}$, instead of $\mathbf{p}$.
- However ℓ is unknown to the attacker; cannot apply linear methods.
- Eliminate the effect of ℓ by enumerating all $\ell = 0, 1, \ldots, m-1$.
- Natural and general approach: a $\Theta(m^2)$-d linear classification model.

$$\begin{bmatrix} p_1 & p_1 & p_1 & \cdots & p_1 & p_1 \\ p_2 & p_1 p_2 p_3 & p_1 p_3 p_4 & \cdots & p_1 p_{m-1} p_m & p_m p_1 \\ p_3 & p_1 p_2 p_4 & p_1 p_3 p_5 & \iddots & p_{m-1} p_1 & p_m p_2 \\ \vdots & \vdots & \iddots & \iddots & \vdots & \vdots \\ p_{m-1} & p_1 p_2 p_m & p_3 p_1 & \cdots & p_{m-1} p_{m-3} & p_m p_{m-2} \\ p_m & p_2 p_1 & p_3 p_2 & \cdots & p_{m-1} p_{m-2} & p_m p_{m-1} \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

- Conjecture: the attacker cannot do better than this approach.

Security of Shift PUFs

- Linear classification model w.r.t. $\mathbf{p}^{(\ell)}$, the parity of $\mathbf{c}^{(\ell)}$, instead of $\mathbf{p}$.
- However ℓ is unknown to the attacker; cannot apply linear methods.
- Eliminate the effect of ℓ by enumerating all $\ell = 0, 1, \ldots, m-1$.
- Natural and general approach: a $\Theta(m^2)$-d linear classification model.

$$\begin{bmatrix} p_1 & p_1 & p_1 & \cdots & p_1 & p_1 \\ p_2 & p_1 p_2 p_3 & p_1 p_3 p_4 & \cdots & p_1 p_{m-1} p_m & p_m p_1 \\ p_3 & p_1 p_2 p_4 & p_1 p_3 p_5 & \iddots & p_{m-1} p_1 & p_m p_2 \\ \vdots & \vdots & \iddots & \iddots & \vdots & \vdots \\ p_{m-1} & p_1 p_2 p_m & p_3 p_1 & \cdots & p_{m-1} p_{m-3} & p_m p_{m-2} \\ p_m & p_2 p_1 & p_3 p_2 & \cdots & p_{m-1} p_{m-2} & p_m p_{m-1} \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

- Conjecture: the attacker cannot do better than this approach.

"Squaring" Arbiter-based PUFs

By substituting APUFs with shift PUFs, all arbiter-based PUF designs might benefit from the $\Theta(m^2)$ enhancement.

- Any machine learning complexity $T(m)$ becomes $T(m^2)$.[2]
- Remark: not for turning insecure design into secure design.
- Substantially adding difficulties to the attacks.
- Alternatively, reducing time and/or circuit complexities of the hardware while preserving the machine learning complexity.

[2] Strictly speaking it is $\Theta(T(m^2))$, as long as T is polynomial in m; also note that $\Theta(T(m^2)) = \Theta(T(m)^2)$ in such case.

"Squaring" Arbiter-based PUFs

By substituting APUFs with shift PUFs, all arbiter-based PUF designs might benefit from the $\Theta(m^2)$ enhancement.

- Any machine learning complexity $T(m)$ becomes $T(m^2)$.[2]
- Remark: not for turning insecure design into secure design.
- Substantially adding difficulties to the attacks.
- Alternatively, reducing time and/or circuit complexities of the hardware while preserving the machine learning complexity.

[2] Strictly speaking it is $\Theta(T(m^2))$, as long as T is polynomial in m; also note that $\Theta(T(m^2)) = \Theta(T(m)^2)$ in such case.

"Squaring" Arbiter-based PUFs

By substituting APUFs with shift PUFs, all arbiter-based PUF designs might benefit from the $\Theta(m^2)$ enhancement.

- Any machine learning complexity $T(m)$ becomes $T(m^2)$.[2]
- Remark: not for turning insecure design into secure design.
- Substantially adding difficulties to the attacks.
- Alternatively, reducing time and/or circuit complexities of the hardware while preserving the machine learning complexity.

[2] Strictly speaking it is $\Theta(T(m^2))$, as long as T is polynomial in m; also note that $\Theta(T(m^2)) = \Theta(T(m)^2)$ in such case.

"Squaring" Arbiter-based PUFs

By substituting APUFs with shift PUFs, all arbiter-based PUF designs might benefit from the $\Theta(m^2)$ enhancement.

- Any machine learning complexity $T(m)$ becomes $T(m^2)$.[2]
- Remark: not for turning insecure design into secure design.
- Substantially adding difficulties to the attacks.
- Alternatively, reducing time and/or circuit complexities of the hardware while preserving the machine learning complexity.

[2] Strictly speaking it is $\Theta(T(m^2))$, as long as T is polynomial in m; also note that $\Theta(T(m^2)) = \Theta(T(m)^2)$ in such case.

"Squaring" Arbiter-based PUFs

By substituting APUFs with shift PUFs, all arbiter-based PUF designs might benefit from the $\Theta(m^2)$ enhancement.

- Any machine learning complexity $T(m)$ becomes $T(m^2)$.[2]
- Remark: not for turning insecure design into secure design.
- Substantially adding difficulties to the attacks.
- Alternatively, reducing time and/or circuit complexities of the hardware while preserving the machine learning complexity.

[2] Strictly speaking it is $\Theta(T(m^2))$, as long as T is polynomial in m; also note that $\Theta(T(m^2)) = \Theta(T(m)^2)$ in such case.

Next Step

To empirically verify the conjecture: $T(m)$ becomes $T(m^2)$.

- Various kinds of arbiter-based PUFs.
- Various commonly used attacks.

Thank you.