ECE 459/559
Secure & Trustworthy Computer Hardware Design

Physical Unclonable Functions
Basics

Garrett S. Rose
Spring 2017

THE UNIVERSITY OF TENNESSEE KNOXVILLE

General Security

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Authentication

What Do We Want to Achieve?

Authentication
no spoofing

Data Integrity
no data alteration

Privacy
no eavesdropping

Attacks

- Software-only protection is not enough!
- Non-volatile memory technologies vulnerable to invasive attack as secrets always exist in digital form

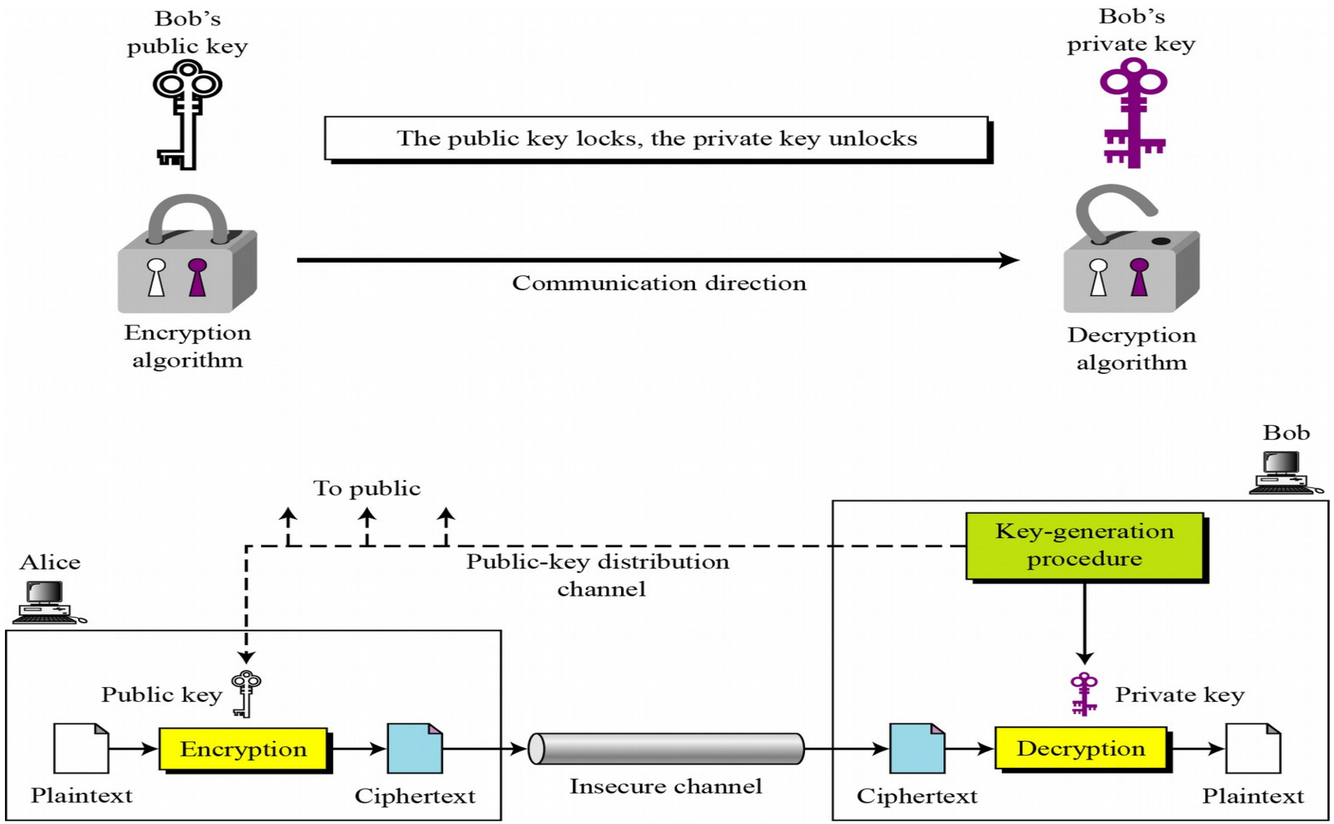
Threat Model – Attacker Goals

- Obtain crypto keys stored in RAM or ROM
- Learn the secret crypto algorithm used
- Obtain other information stored on-chip (e.g. PINs)
- Modify information on the card (e.g. calling card balance)

Some Terminology

- Keys are rules used in algorithms to convert a document into a secret document
- Two types of keys:
  - Symmetric
  - Asymmetric
- Symmetric if same key used for encryption and decryption
- Asymmetric if different keys for encryption and decryption

Asymmetric Security



Security of Asymmetric Keys

- Asymmetry between the information (secret)
- One-way functions
  - Easy to evaluate one direction, hard to reverse in other
  - E.g., multiplying large prime numbers as opposed to factoring
- One-way hash functions
  - Maps a variable length input to a fixed length output
  - Avalanche property: changing one bit in input alters nearly half the output bits
  - Preimage resistant, collision resistant
  - Usage: digital signature, secured password storage, file identification, and message authenticated code

Challenges of Algorithmic One-Way Functions

- Technological
  - Massive number of parallel devices broke DES
  - Reverse-engineering of secure processors
- Fundamental
  - There is no proof that attacks do not exist
  - E.g., quantum computers could factor two large prime numbers in polynomial time
- Practical
  - Embedded systems applications

Solution:
Physical One-Way Function

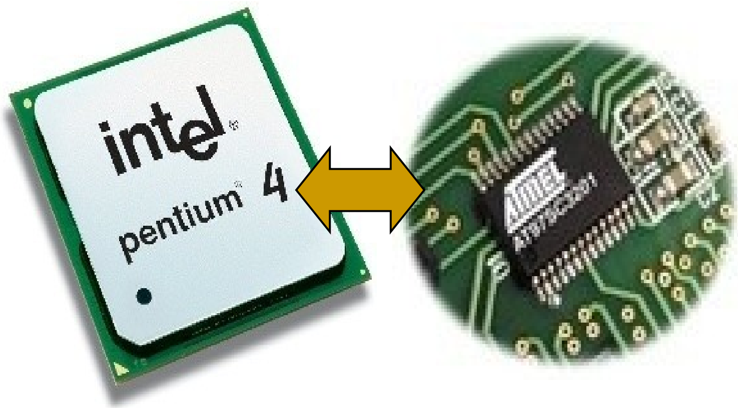
- Use the stochastic physical structures that are difficult to model instead of mathematical one-way functions
- Physical One-Way Function (POWF):
  - Inexpensive to fabricate
  - Prohibitively difficult to duplicate
  - No compact mathematical representation
  - Intrinsically tamper-resistant

Example: IBM 4758

- Goal: Store digital information such that physical attack is difficult and expensive
- IBM 4758
  - Cryptographic coprocessor with secret key generation and memory
  - Includes RNG
  - Tamper-proof package
  - Tens of sensors, resistance, temperature, voltage, etc.
  - Continually battery-powered
  - ~ $3,000 for a 99 MHz processor and 128 MB memory

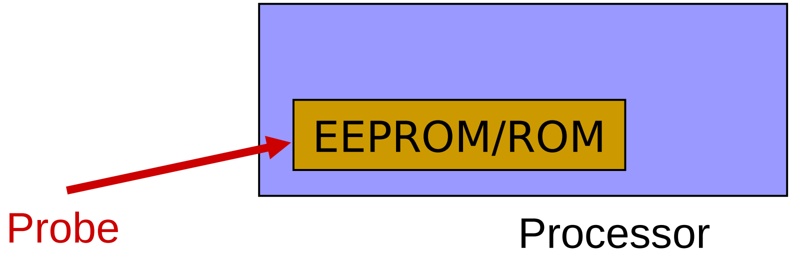
Trusted Platform Module (TPM)

- Separate chip (TPM) specifically designed for security functions
- TPM has become international standard for crypto processors
- Includes cryptographic key generation and RNG
- Decrypted “secondary” keys can be read out from bus



Problems…

Storing digital information in a device in way that is resistant to physical attacks is difficult and expensive



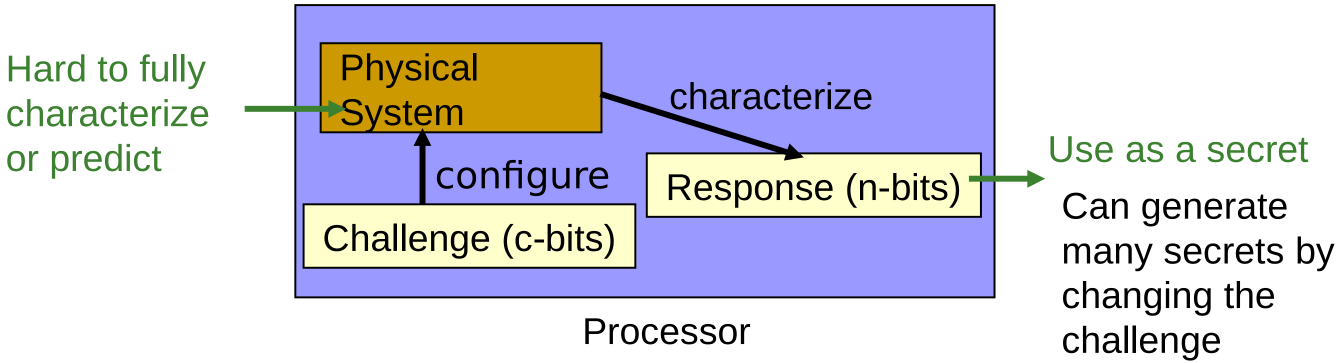
- Adversaries can physically extract secret keys from EEPROM while the processor is off
- Trusted party must embed and test secret keys in a secure location
- EEPROM adds additional complexity to manufacturing

Turn a Problem into Feature!

- Do we expect process variation (length, width, oxide thickness, etc.) in circuit and system? YES!
  - Impact circuit performance
  - Functional failure
  - Major obstacle to continued scaling of integrated circuit technology in sub-45 nm regime
- Process variations can be turned into a feature rather than a problem
  - Each IC has unique properties – exploit for security!

Physical Random Functions

- Generate keys from a complex physical system



- Security advantage:
  - Keys generated on demand → no non-volatile secrets
  - No need to program the secret
  - Can generate multiple master keys
- What can be hard to predict, but easy to measure?

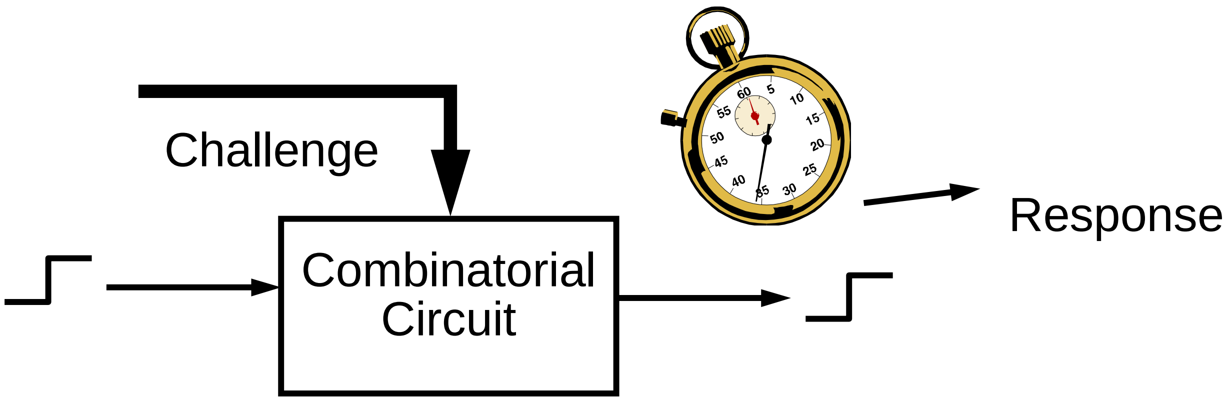
Gassend, et al., “Silicon Physical Random Functions,” CCS 2002.

The PUF Defined

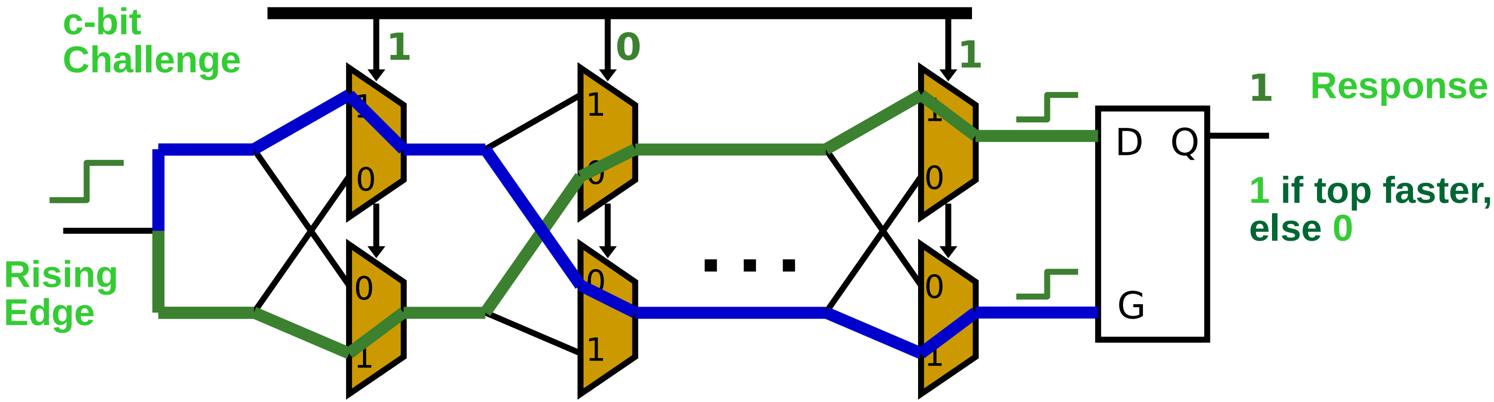
- Physical Random Function or Physical Unclonable Function (PUF) is a function that is:
  - Based on a physical system
  - Easy to evaluate *using the physical system*
  - Output looks like a random/unclonable function
  - Unpredictable even for attacker with physical access

Electronic Silicon PUF

- Due to process variations, no two integrated circuits are identical
- Experiments in which identical circuits with identical layouts are placed on different ICs/FPGAs show that path delays vary enough across ICs to use for identification



The Arbiter PUF



- Compare two paths designed for identical delay
  - Random process variations determine which path faster
  - Arbiter output is 1-bit, representing fastest path
- Path delays in an IC are statistically distributed due to random manufacturing variations

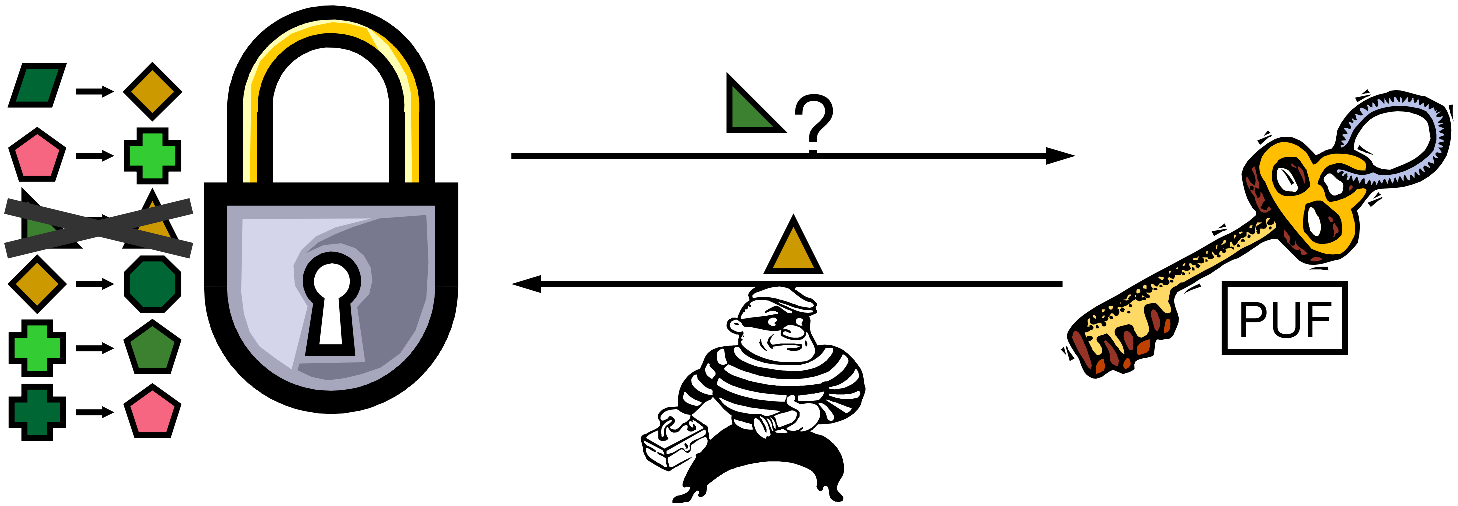
Suh et al., “Physical Unclonable Functions,” DAC 2007.

Physical Attacks

- PUF delays should depend only on physical variations
  - Must minimize any “bias by design”
    (e.g. wire length, transistor sizing, symmetry)
- Invasive attack (e.g., package removal) changes PUF delays and destroys PUF
- Non-invasive attacks are still possible
  - To find wire delays we need precise relative timing of transient signals as opposed to looking for 0's and 1's
  - Wire delay not a number but function of challenge bits and adjacent wire voltages

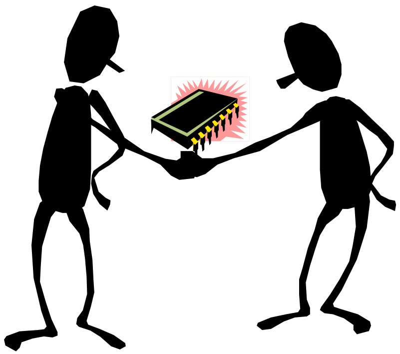
PUF as Unclonable Key

- The lock has database of challenge-response pairs
- To open the lock, key has to show that it knows the response to one or more challeges



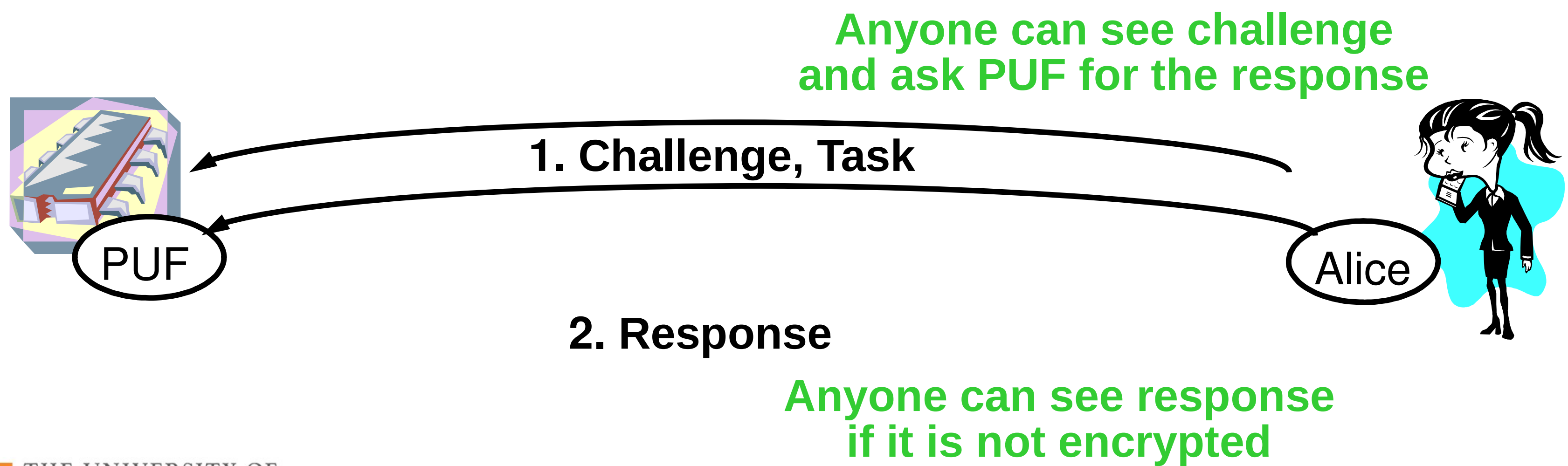
Applications

- Anonymous Computation
  - Alice wants to run computations on Bob's computer & wants to make sure she is getting correct results
  - Certificate returned with her results to show they were correctly executed
- Software Licensing
  - Alice wants to sell Bob a program which will only run on Bob's chip (identified by a PUF)
  - Program is copy-protected so it will not run on any other chip
- Can enable above applications by trusting only single-chip processor that contains a PUF



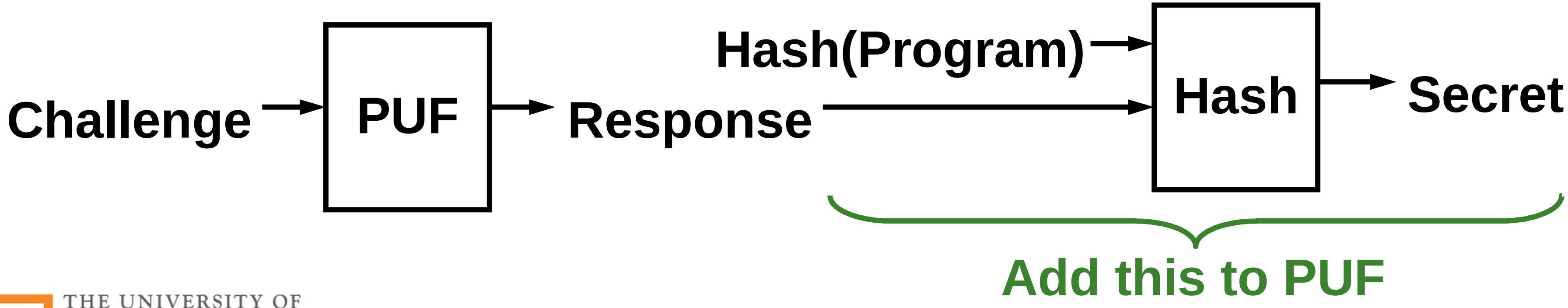
Sharing Secrets using a PUF

- Suppose Alice wishes to share secret using a PUF
- She has a challenge-response pair that no one else knows, which can authenticate the PUF
- She asks the PUF for the response to a challenge



Restricting PUF Access

- To prevent attack, man in the middle must be prevented from finding the response
- Alice's program can establish a shared secret with the PUF, attacker's program must not be able to get secret
  - Combine PUF response with hash of program
- PUF can only be accessed via the GetSecret function:



Gassend et al., “Controlled Physical Random Functions,” ACSAC 2002.

Cryptographic Hash Function

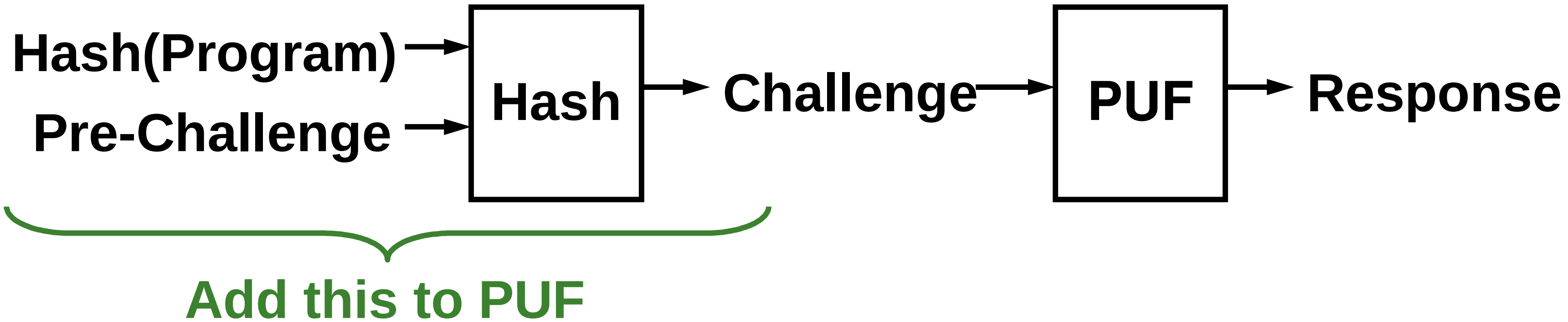
- Crypto hash function h(x) must provide:
  - Compression – output length is small
  - Efficiency – h(x) easy to compute for any x
  - One-way – given value y it is infeasible to find an x such that h(x) = y
  - Weak collision resistance – given x and h(x), infeasible to find y ≠ x such that h(x) = h(y)
  - Strong collision resistance – infeasible to find and x and y, with y ≠ x such that h(x) = h(y)

Getting a Challenge-Response Pair

- Now Alice can use a Challenge-Response pair to generate shared secret with PUF equipped device
- But Alice can't get a Challenge-Response pair in the first place since the PUF does not directly release responses
  - Only releasing “Secret” or hash of program + response
  - Extra function that can return responses needed

Getting a Challenge-Response Pair

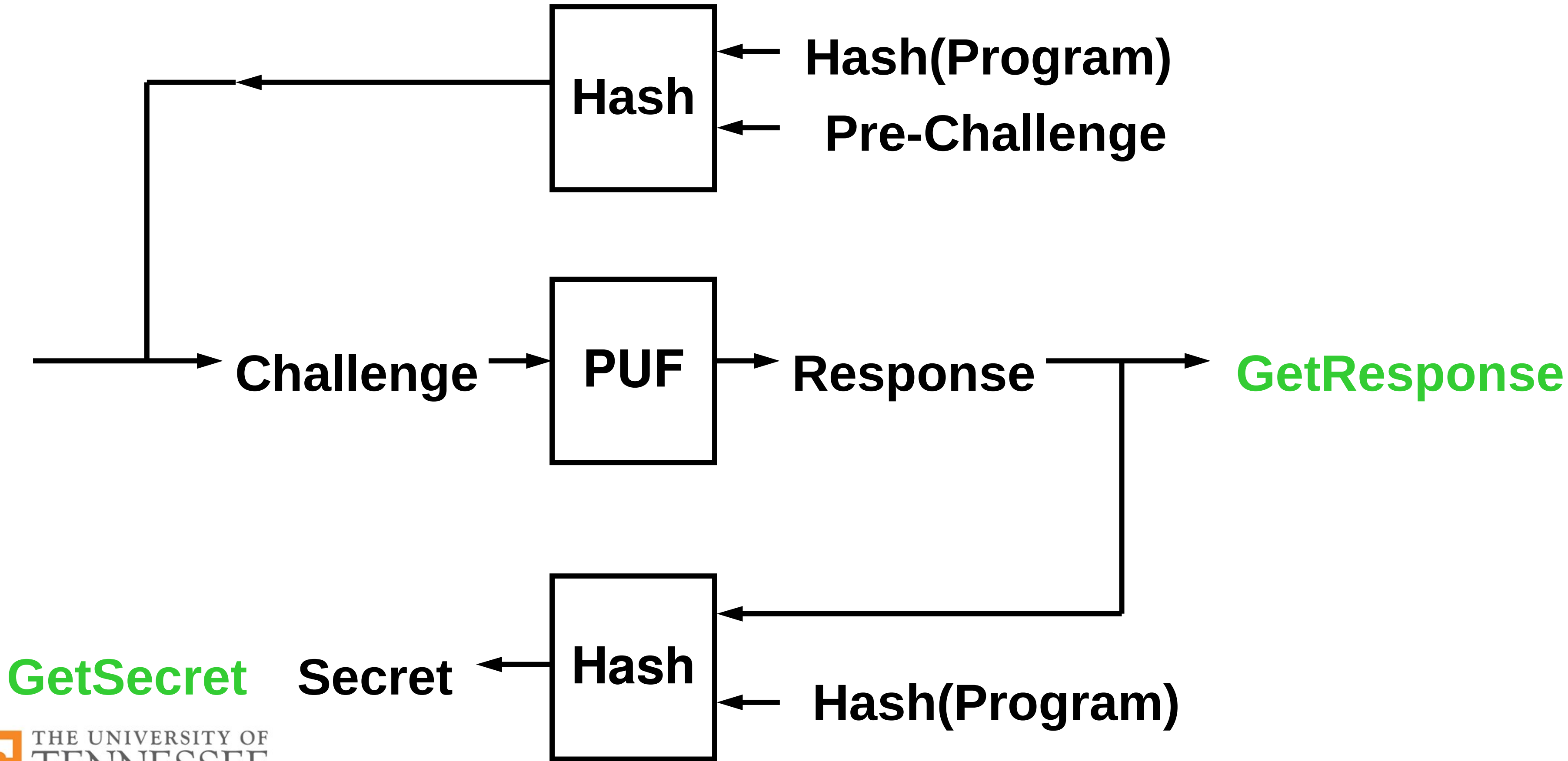
- Let Alice use a Pre-Challenge
- Use program hash to prevent eavesdroppers from using the pre-challenge
- PUF has GetResponse Function



THE UNIVERSITY OF TENNESSEE KNOXVILLE

Gassend et al., “Controlled Physical Random Functions,” ACSAC 2002.

Controlled PUF Implementation



THE UNIVERSITY OF TENNESSEE KNOXVILLE

Gassend et al., “Controlled Physical Random Functions,” ACSAC 2002.

Challenge-Response Pair Management: Bootstrapping

- When controlled PUF (CPUF) has just been produced, manufacturer wants to generate challenge-response pair
  - Manufacturer provided PreChallenge & Program
  - CPUF produces Reponse
  - Manufacturer gets Challenge by computing Hash(Hash(Program), PreChallenge)
  - Manufacturer has (Challenge, Response) pair where Challenge, Program, and Hash(Program) are public but Response not known since PreChallenge thrown away

Gassend et al., "Controlled Physical Random Functions," ACSAC 2002.

Summary

- PUFs provide secret “key” and CPUFs enable sharing a secret with a hardware device
- CPUFs are not susceptible to model-building attack if we assume physical attacks cannot discover PUF response
  - Control protects PUF by obfuscating response, PUF protects control by “covering up” control logic
  - Shared secrets are volatile