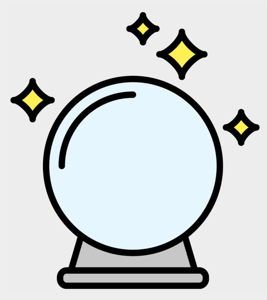
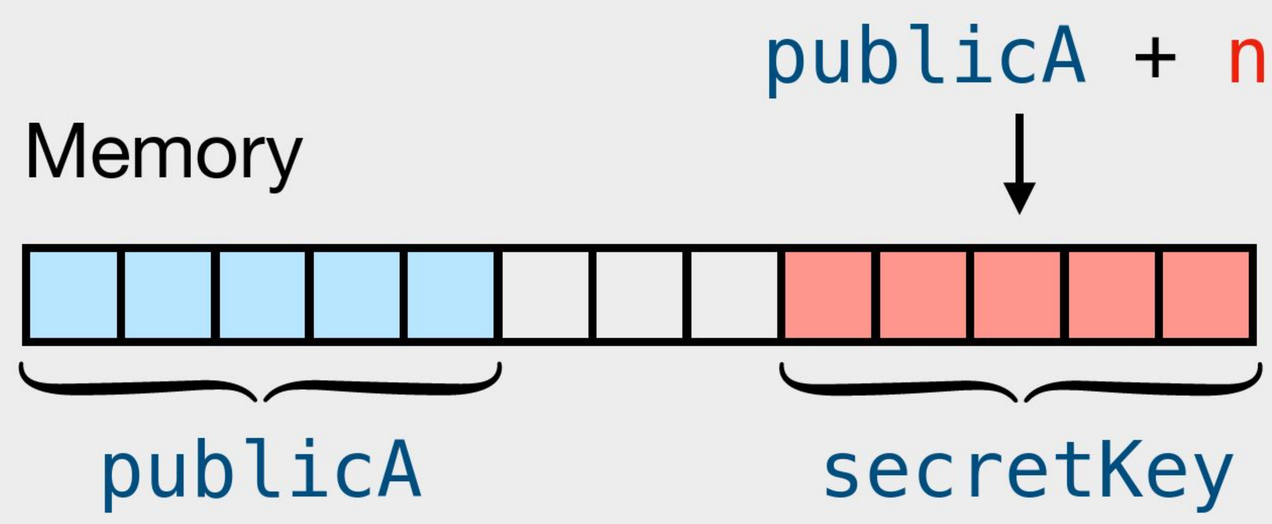
Constant-Time Foundations for the New Spectre Era

Sunjay Cauligi, Craig Disselkoen, Klaus v. Gleissenthall, Dean Tullsen, Deian Stefan, Tamara Rezk, Gilles Barthe

University of California, San Diego
INRIA Sophia Antipolis
IMDEA Software Institute
MPI for Security and Privacy



```
if (n < publicLen) {
  x = publicA[n]; // n is normally safe, but is OoB when misspeculated!
  y = publicB[x]; → Secret memory access!
}
```



Spectre attacks can **break secure code.**

The only robust way to prevent leaking secrets in cryptographic code is to use *constant-time programming*.

Unfortunately, Spectre attacks reveal that even *securely written* cryptographic code may unintentionally leak secret information as a result of *misspeculation* in the processor.

The example on the left is constant-time. But if the processor *misspeculates* into the branch, it can still leak bytes of `secretKey` via the cache!

Existing Spectre defenses are **ad hoc** and **miss attacks.**

Existing defenses are generally unsound (Microsoft's `/Qspectre` compiler flag) or far too heavy-handed (Intel's SSBD feature) — **we need defenses rooted in formal methods**. To that end, we define **Speculative Constant-Time** (SCT), the *first formal notion of security for cryptographic code*. Code that is SCT is secure even when the attacker has *complete control* over the branch predictor or other hardware features!

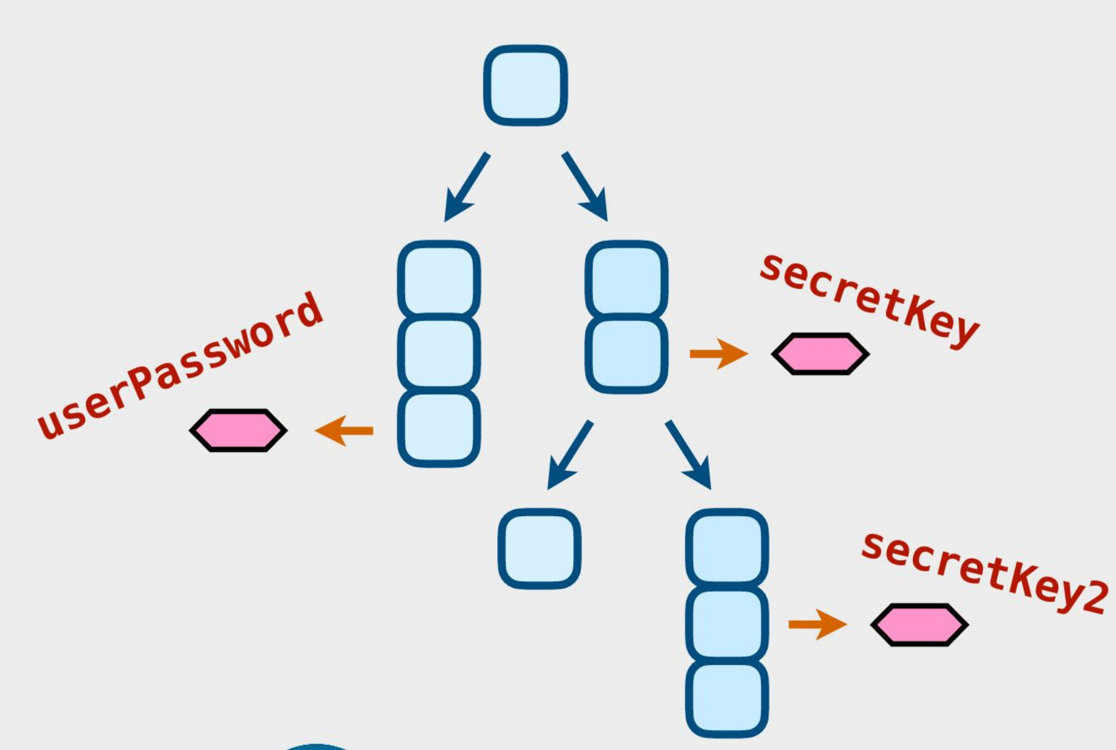
SCT is backed by our *execution semantics*, which is powerful enough to capture every known variant of Spectre, *including future ones* — we predicted and modeled the flaw in AMD's *“Predictive Store Forwarding”* feature, **which wasn't even in processors until after our paper!**

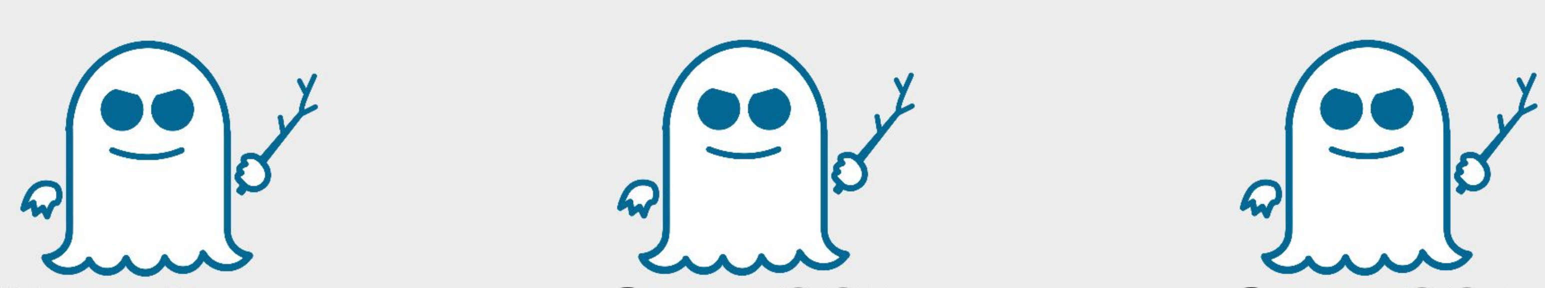
We can adapt **new hardware features** and model **future Spectre variants.**

Pitchfork reveals Spectre gadgets in **real code.**

Our semantics is also the basis for *Pitchfork*, our prototype analysis tool. Pitchfork explores every *speculative* execution path in a binary and detects whether secrets can be leaked.

We used Pitchfork to find Spectre gadgets in the libsodium and OpenSSL libraries, in code that was previously verified to be *constant-time*. In fact, we found that *compilers themselves* can insert Spectre gadgets: The vulnerable code in libsodium was part of Clang's stack-smashing defense!





libsodium secretbox

OpenSSL ssl3 validate

OpenSSL MEE-CBC

UCSD CSE
Computer Science and Engineering