It's been a while since we posted aboutproject Trustless TEE, but there's been a lot happening on the ground.

Many conversations have been had between @gluonix (who has been driving a lot of this), @PradyumnaShome, Fabric cryptography, Dan Boneh, @socrates1024, @jonathan, researchers at UC Louvain's simple crypto group and many others.

At a high level, here's a summary of our (at least my) current thinking:

- Open source

- a lot of hardware projects that claim to be open, only release their HDL which cannot be cross-checked with the actual chip. We need GDSII files.

- a lot of hardware projects that claim to be open, only release their HDL which cannot be cross-checked with the actual chip. We need GDSII files.

- manufacturing:

- we know that PUFs are essential to get around trusting the manufacturer. There's a lot of details here that@gluonix knows more about but it seems like weak PUFs are the direction to go in the short run and that a substantial part of the design challenge is making sure the use of the PUF response (fuzzy extraction process, signing, decryption etc) doesn't like info through e.g. EM radiation. Folks at UC Louvain are working on a PUF-based signing oracle that ticks all the boxes.

- the chip should not be more vulnerable to physical attacks during manufacturing than when complete. A promising approach in this direction is by having the PUF be derived from the whole chips hardware fingerprint or even also the capsule/cover.

- we know that PUFs are essential to get around trusting the manufacturer. There's a lot of details here that@gluonix knows more about but it seems like weak PUFs are the direction to go in the short run and that a substantial part of the design challenge is making sure the use of the PUF response (fuzzy extraction process, signing, decryption etc) doesn't like info through e.g. EM radiation. Folks at UC Louvain are working on a PUF-based signing oracle that ticks all the boxes.

- the chip should not be more vulnerable to physical attacks during manufacturing than when complete. A promising approach in this direction is by having the PUF be derived from the whole chips hardware fingerprint or even also the capsule/cover.

- hardware trojans detection

: * the toughest trojan to detect that we are aware of isthis attack which simply changes dopant levels on transistors. Its possible to detect it, but this requires destroying the chip in order to use a scanning electron microscope (SEM). As suggested in this paper about infra-red imaging, a non-destructive approach could be arrived at by combining techniques like logic locking, scan chains and path delay fingerprinting to force trojans to a larger scale, making them easier to detect. Another suggestion from people at Fabric Crypto is to run transistors "near threshold" to render chips with dopant-based trojans inoperable.

- we know that non-invasive techniques are important for decentralised testing The most powerful non-invasive analysis technique is X-ray ptychometry, but this is a very slow and expensive. There is alsoIRIS, which non-destructively uses short-wave infra-red imaging, but due to limited resolution needs to be combined with other techniques as mentioned above. Its still not clear to us (me?) how robust and well tested these techniques really are.

- the toughest trojan to detect that we are aware of isthis attack which simply changes dopant levels on transistors. Its possible to detect it, but this requires destroying the chip in order to use a scanning electron microscope (SEM). As suggested in this paper about infra-red imaging, a non-destructive approach could be arrived at by combining techniques like logic locking, scan chains and path delay fingerprinting to force trojans to a larger scale, making them easier to detect. Another suggestion from people at Fabric Crypto is to run transistors "near threshold" to render chips with dopant-based trojans inoperable.

- we know that non-invasive techniques are important for decentralised testing The most powerful non-invasive analysis technique is X-ray ptychometry, but this is a very slow and expensive. There is alsoIRIS, which non-destructively uses short-wave infra-red imaging, but due to limited resolution needs to be combined with other techniques as mentioned above. Its still not clear to us (me?) how robust and well tested these techniques really are.

- Side channels and processor security

: * There are three levels in which side channels can leak information: * at the software-implementation level

. E.g. branching on secret bits

- at the microarchitecture level

. For example, speculative execution and out-of-order execution have been exploited in Spectre and Meltdown before, but there's also a whole bunch of other microarchitectural channels [1, 2, 3, 4].

- at the logic-gate level

. Even if the processor isn't doing any performance optimisation like spec. execution or prefetching, there is still some information leakage due to some channels like different gates taking slightly longer or using more power than others (see here for more). Dual-rail techniques and randomisation are relevant here (although I don't know enough yet)

- at the software-implementation level

. E.g. branching on secret bits

- at the microarchitecture level

. For example, speculative execution and out-of-order execution have been exploited in Spectre and Meltdown before, but there's also a whole bunch of other microarchitectural channels [1, 2, 3, 4].

- at the logic-gate level

. Even if the processor isn't doing any performance optimisation like spec. execution or prefetching, there is still some information leakage due to some channels like different gates taking slightly longer or using more power than others (see here for more). Dual-rail techniques and randomisation are relevant here (although I don't know enough yet)

- we can't address the software issue at the hardware level so we look at the other two. In order to mitigate microarchitectural challenges, we will need to prune away a lot of performance enhancing features.

- There are three levels in which side channels can leak information:

- at the software-implementation level

. E.g. branching on secret bits

- at the microarchitecture level

. For example, speculative execution and out-of-order execution have been exploited in Spectre and Meltdown before, but there's also a whole bunch of other microarchitectural channels [1, 2, 3, 4].

- at the logic-gate level

. Even if the processor isn't doing any performance optimisation like spec. execution or prefetching, there is still some information leakage due to some channels like different gates taking slightly longer or using more power than others (see here for more). Dual-rail techniques and randomisation are relevant here (although I don't know enough yet)

- at the software-implementation level

. E.g. branching on secret bits

- at the microarchitecture level

. For example, speculative execution and out-of-order execution have been exploited in Spectre and Meltdown before, but there's also a whole bunch of other microarchitectural channels [1, 2, 3, 4].

- at the logic-gate level

. Even if the processor isn't doing any performance optimisation like spec. execution or prefetching, there is still some information leakage due to some channels like different gates taking slightly longer or using more power than others (see here for more). Dual-rail techniques and randomisation are relevant here (although I don't know enough yet)

- we can't address the software issue at the hardware level so we look at the other two. In order to mitigate microarchitectural challenges, we will need to prune away a lot of performance enhancing features.

Microarchitectural vulnerabilities indicate that we may need to build the TEE on a completely redesigned processor that does away with many of the performance enhancements we benefit from today and can be open sourced fully. We are still figuring out what we can keep, but we are facing a serious slowdown. Based off some numbers that haven't been checked

- speculative/OoO execution (10x)

- cache (200x)

- the best open source fabs like Google Skywater have 90nm of precision (in contrast with the state of the art 4m) which would lead to 100-1000x slowdown

These numbers are still very rough, but in the worst case we could be looking at a secure confidential processor that is up to

6 or seven orders of magnitude slower than the top range CPU. Naturally figuring out how slow this would actually be is the first thing to do. If we do see it being quite slow (or too hard to make secure) then we have an alternative: leverage a secure coprocessor for only the essentials and then use accelerated ZK/FHE/MPC or a CPU-based TEE with only integrity and no confidentiality guarantees for the rest.

This is the main direction we're exploring in the near future.

Now the question is what kind of functionality would we need such a coprocessor to support? This question is relevant because we may end up deciding to focus on specialised functionality exposed through a narrow API as a milestone and, even if we don't and go straight for a general-purpose coprocessor, understanding what actually has to run on the secure processor gives a better idea of the performance impact. Some ideas for what the secure processor should support:

- signing oracle to keep keys secure (this is what simple crypto is working on)
- signing oracle with a non-volatile counter for anti-equivocation use cases
- provable key deletion would give us one shot signatures and provide defense against long range attacks
- key refresh for forward secrecy
- randomness oracle
- some reencryption or arithmetic operations (1, 2) can be paired well with FHE/MPC. This is a performance optimisation as security would still depend on the TEE.
- some kind of TLS primitive
- the minimal functionality required to stop early decryption/unauthorised signing. For example, we could have the secure processor verify a succinct proof of a blockchains state, triggering a one-time decryption of data.
- verification of proofs of correctness: if we offload compute to a faster TEE-with-only-integrity or to an ASIC, we need to verify the correctness of the computation. For the TEE, verification is just some signature checking and the harder part is getting integrity in the transparent TEE (e.g. we need to address rowhammer attacks). For the ASIC compute verification we can try to use a SNARK-like proving scheme.
- ?

We could try to implement (some) of the above through a limited API in a special purpose hardware component or in a slower general purpose processor. Naturally, if we could get a secure processor.

As a meta problem: we are trying to figure out how best to communicate our ideas. The initial plan was to write a high-level position paper, but it seems the ideal hardware security venues need more concrete "scientific contributions" so we could do an SoK, choose a different venue or something else.