Google

GPAM: Generalized power attack model

Breaking ECC and AES with a single model


Karel Král


Jean-Michel Picod

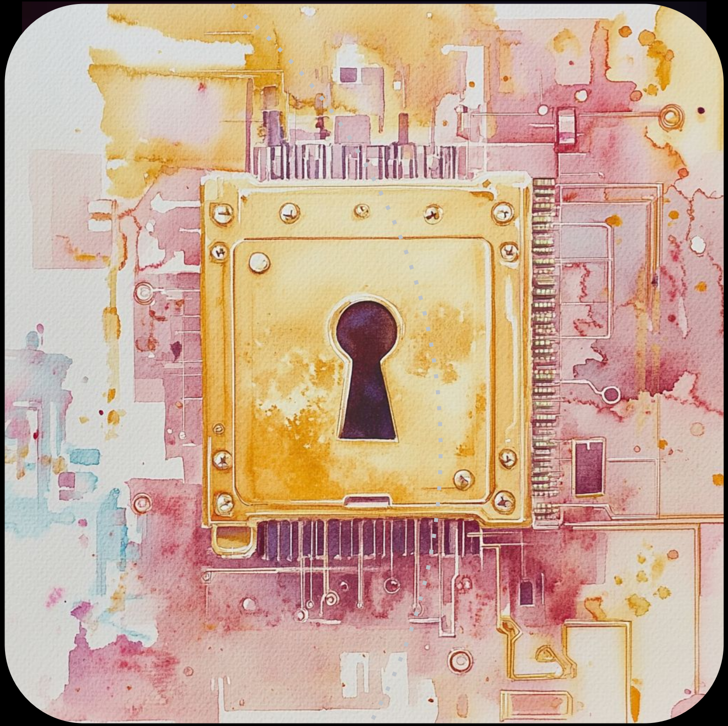

Elie Bursztein

with the help of Luca Invernizzi, Daniel Moghimi, and Marina Zhang and many Googlers

SCAN ME

Presentation slides and more available here:
https://elie.net/gpam

Side-channel attacks are human labor intensive



Scaling hardware implementation security testing prohibitively expensive

Leverage recent advance in deep-learning to create
fully generalized automated side-channel attacks

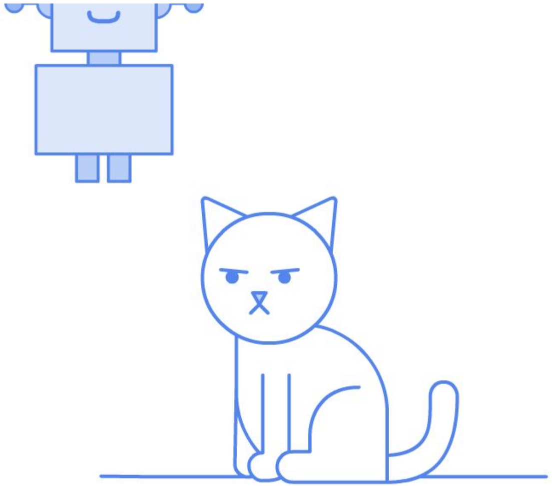
Google

Security and Privacy Research

Why generalizing?

AI Generalization benefits

Full trace w/o pre-processing	Reduce human labor
Multi-algorithms	Work on all type of algorithms without changing the model
Multi-counter-measure	Work on all type of implementations, and countermeasures
Full automated	No human intervention requires - only compute light hypertuning



Fully automated and general AI? Really?

	Dataset	Trace-size	Attack point	Accuracy	MeanRank
ECC	new	1.6M	k0	**100%**	0
Masked ECC	new	5M - **17.5M**	k0	78% - 8.6%	0.75 - 20
Masked AES	ASCAD v2	1M	c[i]	1.18%	80

Regardless of the
the algorithm, implementation protections, and trace length
GPAM is able to reliably and automatically
attack hardware implementations

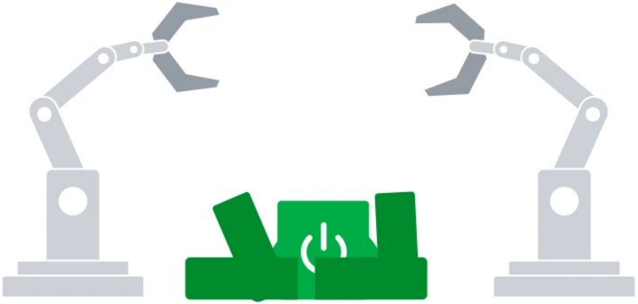
Agenda

GPAM Model architecture

(new) Datasets

Results

GPAM model architecture

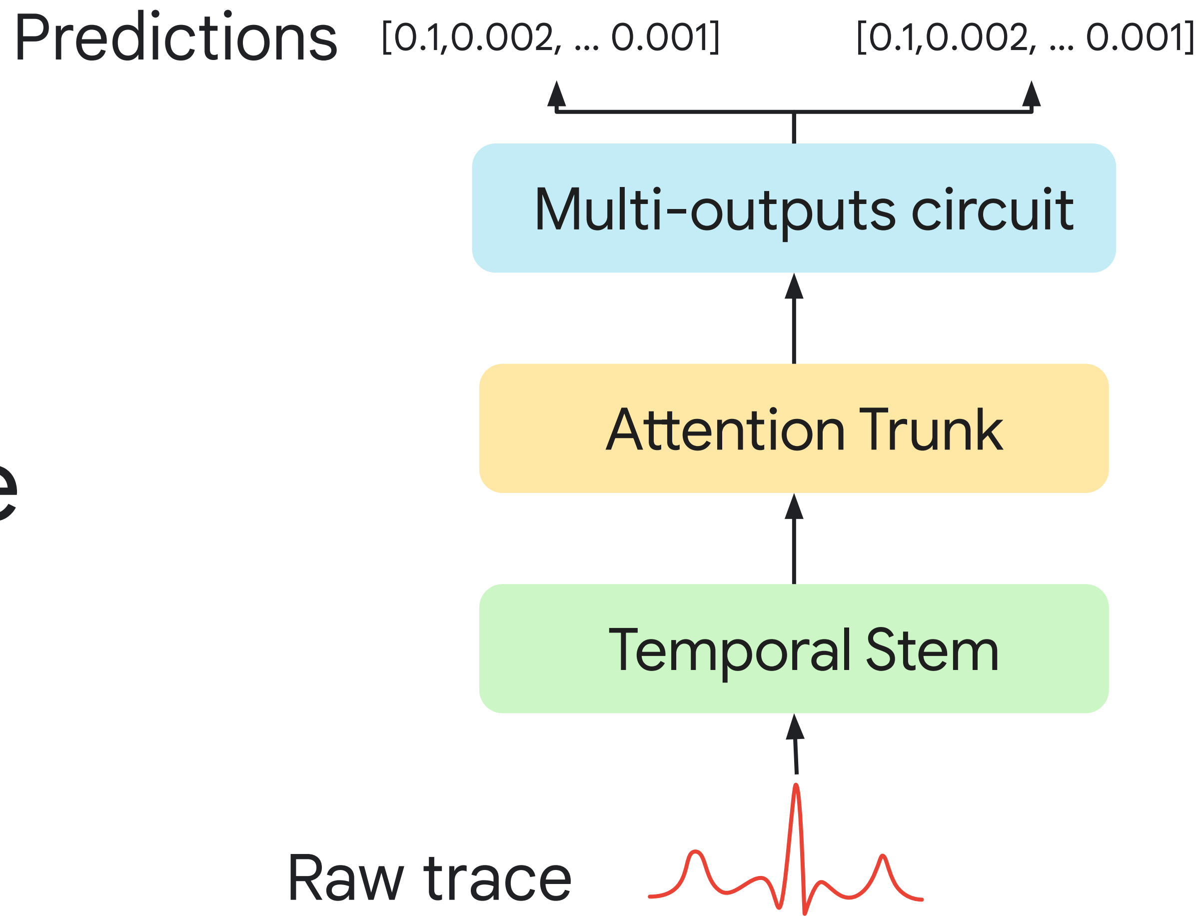


GPAM combines state of art deep-learning techniques to provide a general & efficient model that can be tuned automatically

Optimized to be easy to be tuned and trained on commodity hardware

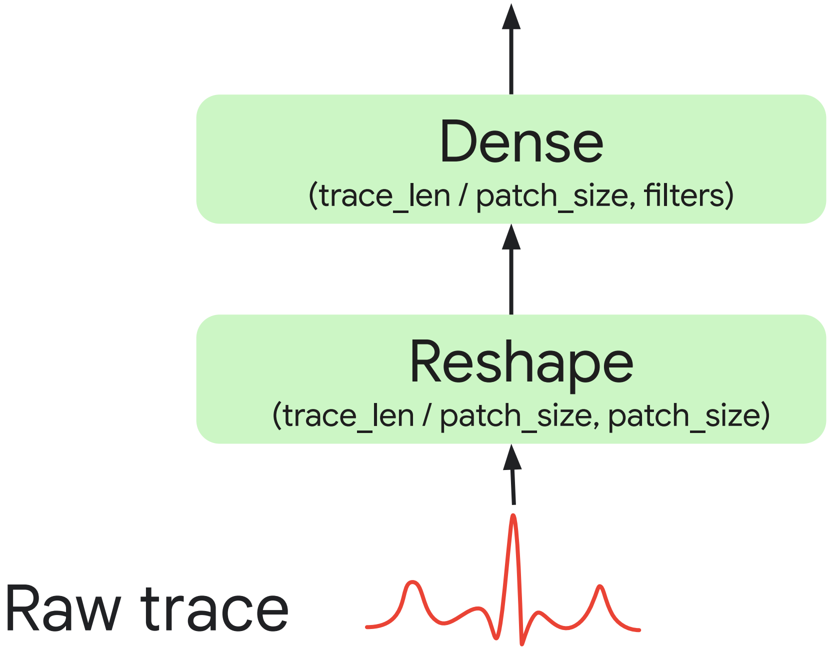
Attacking a new Implementation requiring ~700 GPU Hours

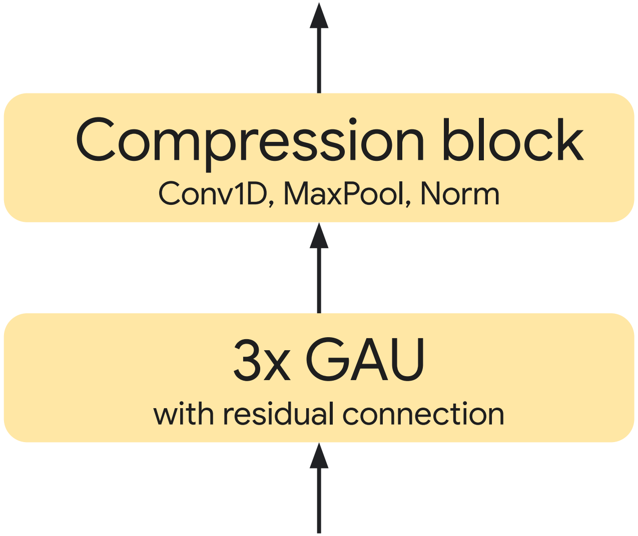
GPAM architecture overview



Temporal Stem

Create a **learned compressed representation suitable for long range prediction** by performing packing and patchification which is critical to modern model performance [ConvNext]



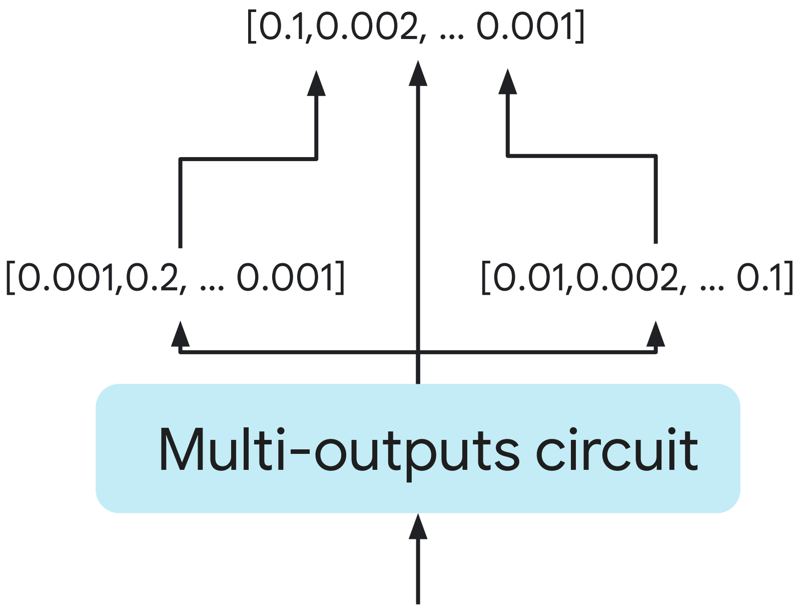


Attention Trunk

Combine state of art **transformer decoding blocks for long range leakage interaction understanding** and a **compression block for efficient features extraction**

Multi-outputs circuit

Novel technique that **interconnects the model heads as a DAG** to **encode algorithm leakage points understanding** into the model for better performance



Notes from the architect

Large trace and patchification

Doing the non-overlapping convolution feature extraction on the trace allows to compress its representation which is **critical for perf and scale to very long traces** – make sure to include this as best practice

GAU vs Transformer block

GAU is significantly faster than regular transformer decoder block while providing better perf. Using SOTA relative positional encoding is critical for performance - RoPE seems good

Training regime matters

Adafactor or ADAM optimizer with a careful learning rate schedule is critical to training stability and performance.

(New) datasets

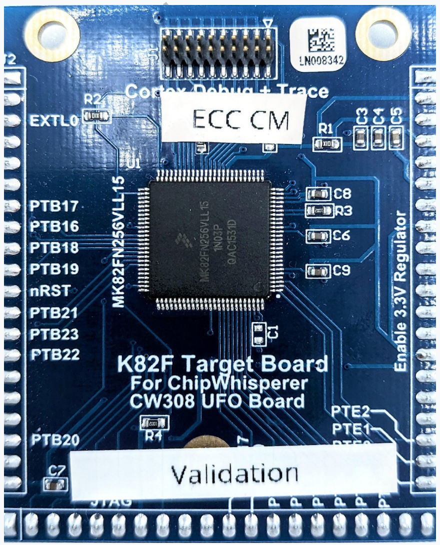
Dual evaluation strategy reusing SOTA AES datasets and creating extensive ECC datasets

Existing datasets

Name	Algorithm	Protection	Target	Train Size	Test size	Disk size
ASCADv2 [1]	AES	shuffle & affine mask	STM32F3	640,000	80,000	880GB
REASSURE [2]	ECC	arithm. swap & randomization	STM32F4	1.2M	153,000	7GB
SMAesH S6 [3]	AES	hardware private circuits	Spartan-6	17M	17M	250GB
SMAesH A7 [3]	AES	HPC	Artix-7	17M	17M	220GB
ASCADv1 [4]	AES	masked	ATmega8515	180,000	100,000	52GB

[1] Loïc Masure and Rémi Strullu. Side-channel analysis against ANSSI's protected AES implementation on ARM: end-to-end attacks with multi-task learning
[2] Łukasz Chmielewski. Reassure (h2020 731591) ECC dataset
[3] Gaëtan Cassiers, Charles Momin and François-Xavier Standaert, SMAesH Challenge
[4] Ryad Benadjila, et.al., Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database

Creating high quality ECC reference datasets



Targeting LTC in K82F chips which provides constant time Mul and Add operations

Capture using LeCroy Wavepro 404HD-MS at 50MS/s

Ensuring realistic settings by using a different MCU to capture testing data

ECC countermeasures implemented

Unprotected HW (CM0)	**k * G** k <- Rand(256) *(random 256-bit multiplier k for each example)*
Additive masking (CM1)	**r * G + (k - r) * G** k <- Rand(256), r <- Rand(256)

Multiplicative masking (CM2)	**r * ((k // r) * G) + (k % r) * G**
	k <- Rand(256), r <- Rand(128)

Combined masking (CM3)	**(r1 * G mul masked by r2) + ((k - r1) * G mul masked by r3)** k <- Rand(256), r1 <- Rand(256), r2 <- Rand(128), r3 <- Rand(128)

New datasets

	Trace length	Number of traces	Disk usage
ECC CM0	1’600’000	73’000	200GB
ECC CM1	5’000’000	208’000	1.5TB
ECC CM2	10’000’000	138’000	2.1TB
ECC CM3	17’500’000	138’000	3.7TB

Results

Whitebox attacks results

	LSTM (CHES21)	CNN (VGG-16)	ConvNeXt	GPAM
ECC CM0	91.4%	100%	100%	100%
ECC CM1	random	random	74.5%	78.8%
ECC CM2	-	-	14%	66.2%
ECC CM3	-	-	random	8.6%

Blackbox attack results

	LSTM (CHES21)	CNN (VGG-16)	ConvNeXt	GPAM
ECC CM0	91.4%	100%	100%	100%
ECC CM1	-	-	random	random
ECC CM2	-	-	3.5%	22.8%
ECC CM3	-	-	-	random

AES results

	SoTA	GPAM
ASCADv2	60 traces to recover key [MS23]	80 traces to recover key, full trace
ASCADv1	Multitrace DL attacks [LZC+21], [HCM24]; single trace [BCS21]	96% acc byte 3 of SBOX input
SMAesH S6	290k traces, GE < 2^60 (of the whole key)	GE between 2^70 and 2^90 (of the whole key)
SMAesH A7	900k traces, GE < 2^60 (of the whole key)	GE around 2^90 (of the whole key)

While GPAM doesn't reach SoTA performance like hand-crafted models it deliver strong performance against all implementations

Takeaways

GPAM allows fully automated side-channel attacks testing

Scaling up benchmarking to more algorithms and modality is a priority

More research needed on automating leakage origin pinpointing

Thank you

get the paper and slides
at https://elie.net/gpam