Side-channel attacks

Pecha Kucha Presentation

Benjamin Drisch
Eviden Digital Identity

1996: Paul Kocher's timing attack

Derives RSA key by timing about 1000 operations

Timing Attack is a Side-Channel Attack

Side-Channel Attack: Deriving a secret key using leaked information

Side-Channel Attacks: efficient, but …

Attack implementation, not algorithm

Work well on isolated modules, but difficult on PCs, smart phones, …

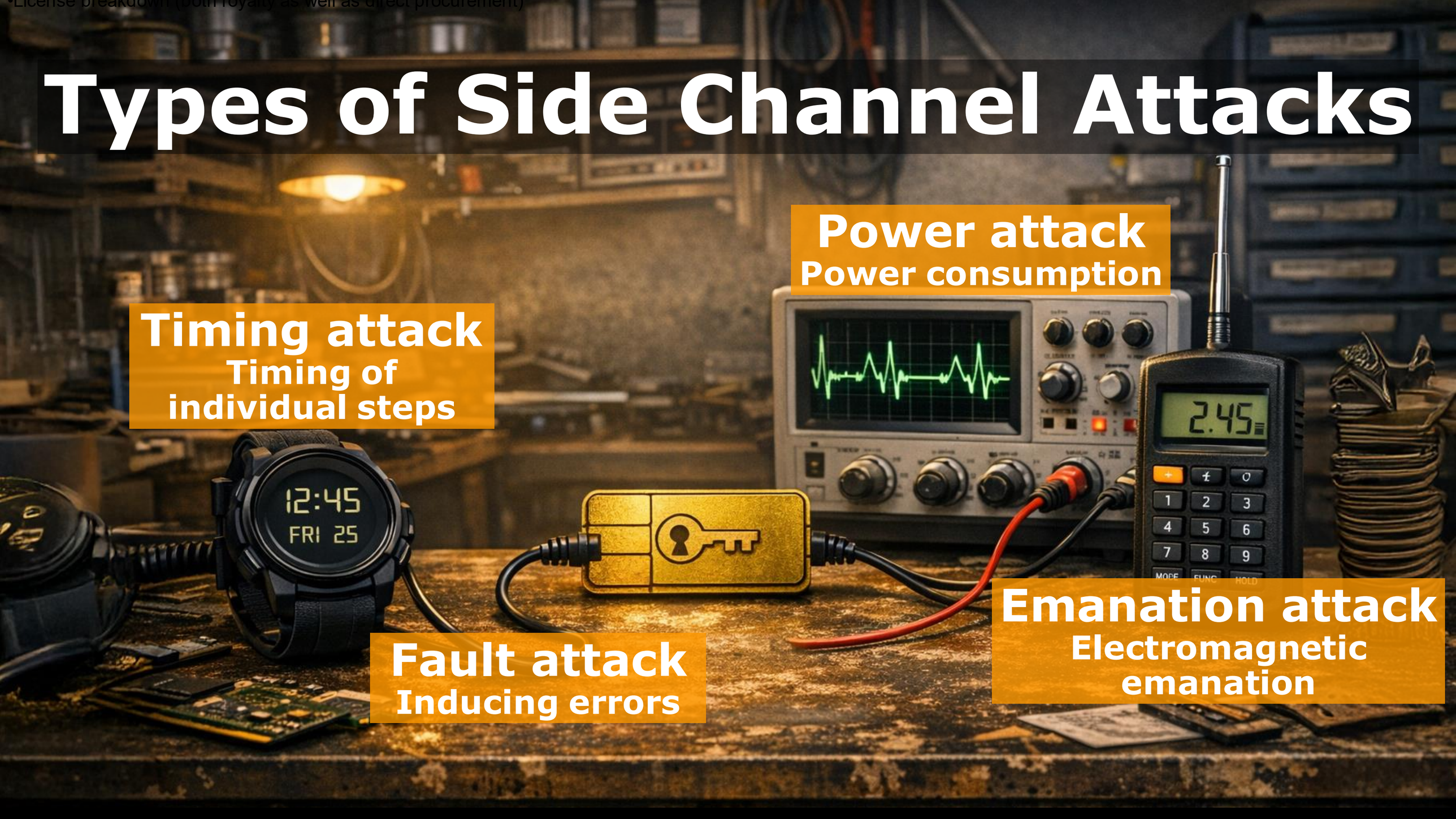
Where do these attacks apply best?

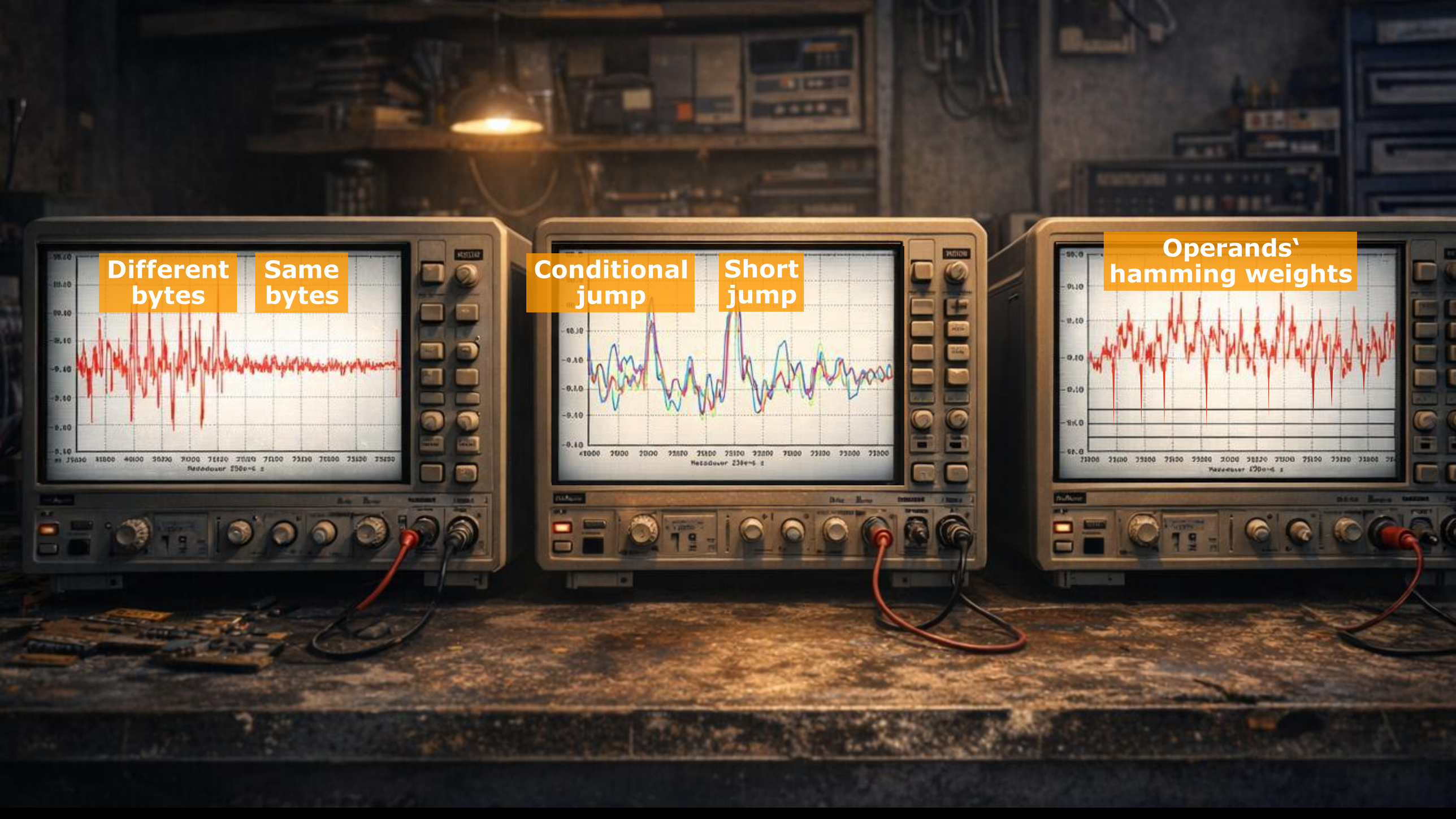
Embedded devices, especially smart cards

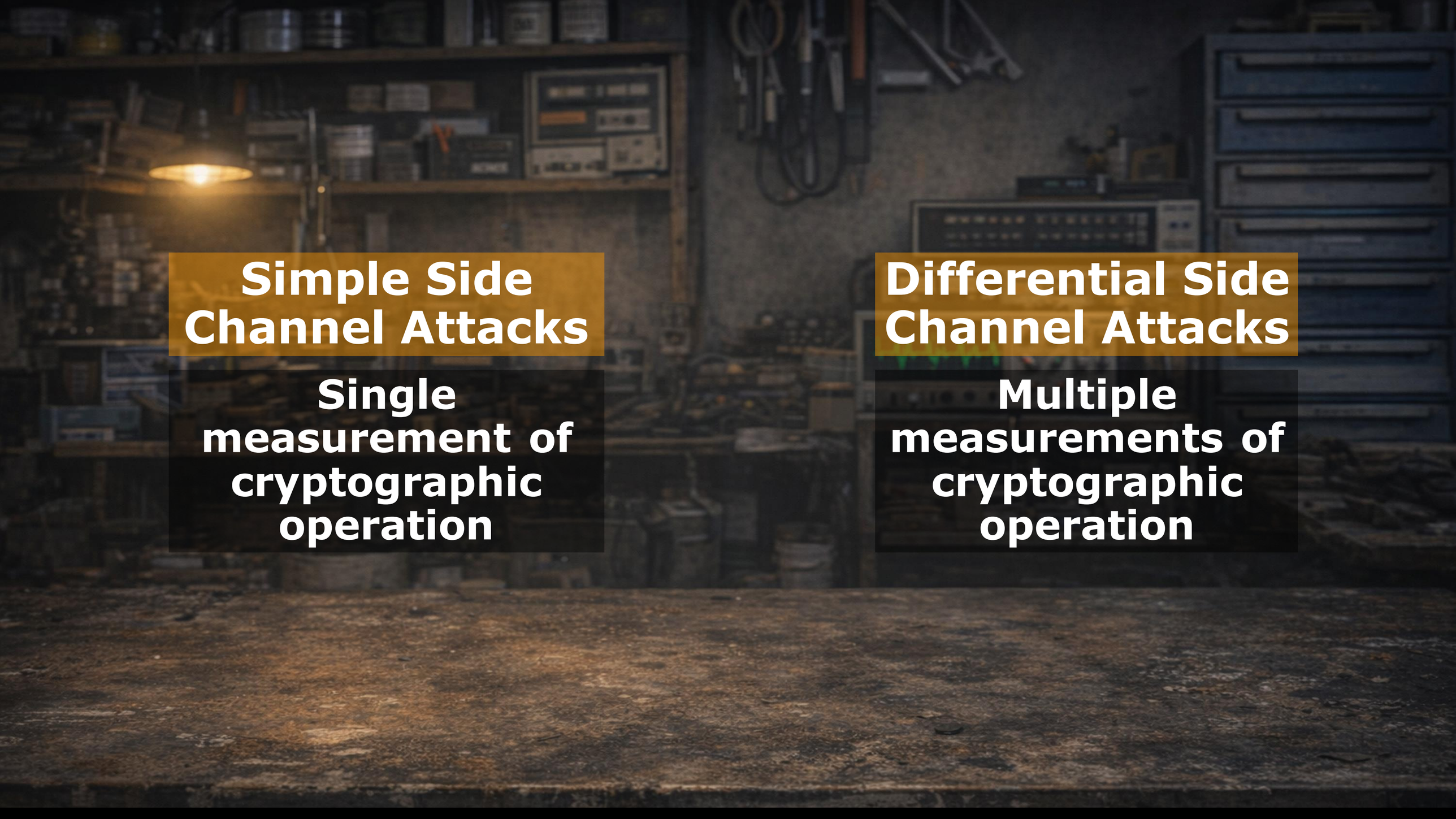
Smart cards require external power supply; often operated via contactless interface

•License breakdown (both royalty as well as direct procurement)

Types of Side Channel Attacks




Different bytes
Same bytes
Conditional jump
Short jump
Operands` hamming weights

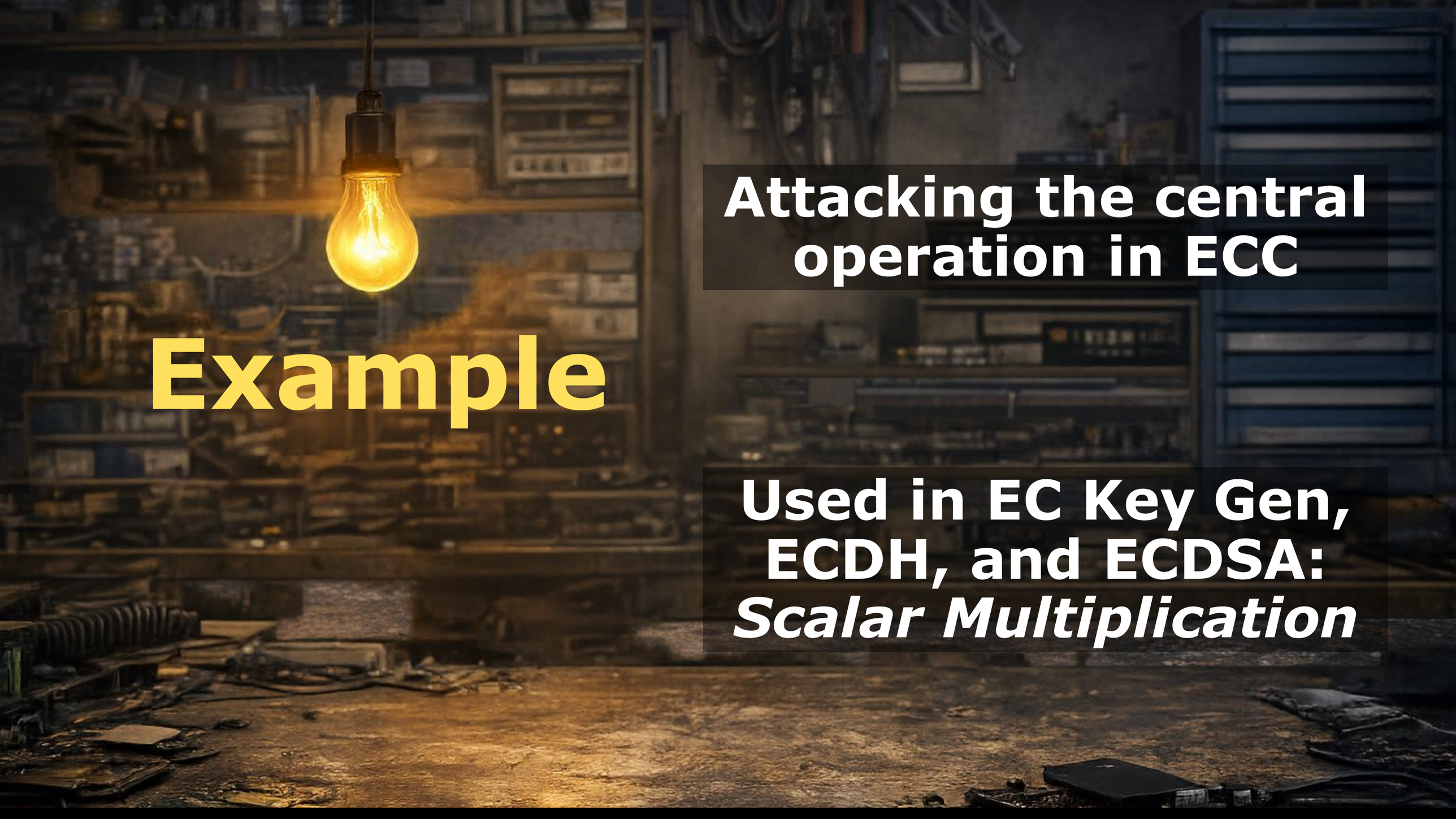


Simple Side Channel Attacks

Single measurement of cryptographic operation

Differential Side Channel Attacks

Multiple measurements of cryptographic operation


Example
Attacking the central operation in ECC
Used in EC Key Gen, ECDH, and ECDSA:
Scalar Multiplication

Multiplying with *Double & Add*

We want to compute Q = x * P for a secret x and public EC point P

We represent x as binary string, say x = 1011001101 …

We start at the second bit
- for every 0 we double
- for every 1 we additional add P

Consider 5=‘101’$_2$: two mults, one add

$$Q = 2 * 2 * P + 1 = 5 * P$$

Algorithm 1: "Naïve" *Double & Add*

Secret scalar x = 1011001101 …

```
Q ← P
For i = r-2 to 0
   Q ← 2*Q
   If x_i = 1 then Q ← Q+P
Output Q
```

If current bit = 1 add point *P*

Algorithm 1: "Naïve" *Double & Add*

Simple Side Channel Attack

Secret scalar x = 1011001101 …

```
Q ← P
For i = r-2 to 0
   Q ← 2*Q
   If x_i = 1 then Q ← Q+P
Output Q
```

Conditional Addition ←

!

Algorithm 2: Modified *Double & Add*

```
Q[0] ← P
For i = r-2 to 0
    Q[0] ← 2*Q[0]
    Q[1] ← Q[0]+P
    Q[0] ← Q[x_i]
Output Q
```

Always perform additional add

Assign correct option based on scalar's bit value

Algorithm 2: Modified *Double & Add*

Differential Side Channel Attack

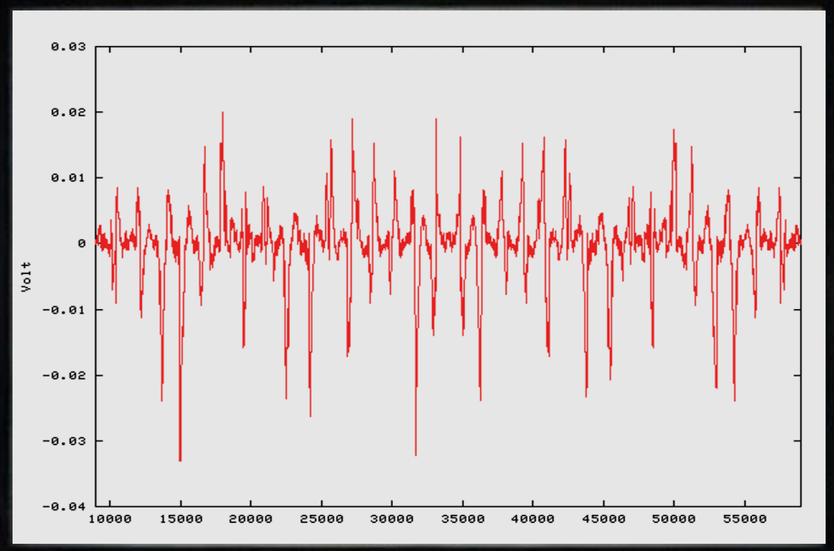
```
Q[0] ← P
For i = r-2 to 0
   Q[0] ← 2*Q[0]
   Q[1] ← Q[0]+P
   Q[0] ← Q[x_i]
Output Q
```

No knowledge about implementation required

Guess the first secret bit and a corresponding intermediate value, say Q = 2*P. Then define a *Selection Function*

Algorithm 2: Modified *Double & Add*

Differential Side Channel Attack

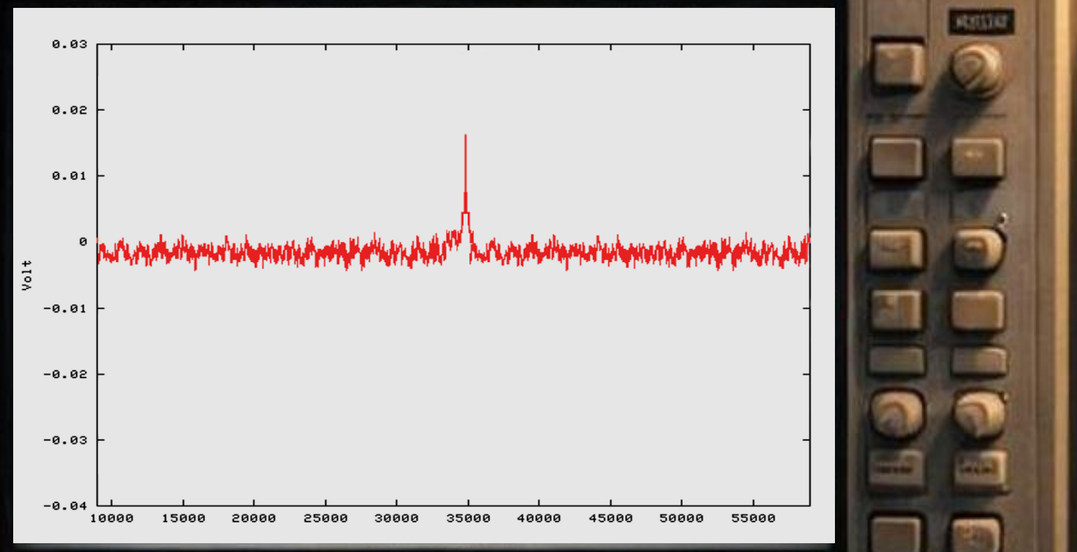


Perform multiple measurements with different scalars x

Create two groups:
S1: traces with Q=2*P
S2: traces with Q≠2*P

Algorithm 2: Modified *Double & Add*

Differential Side Channel Attack



If we guessed right, Q=2*P must have been calculated somewhen

S1 – S2 should show a peak somewhere

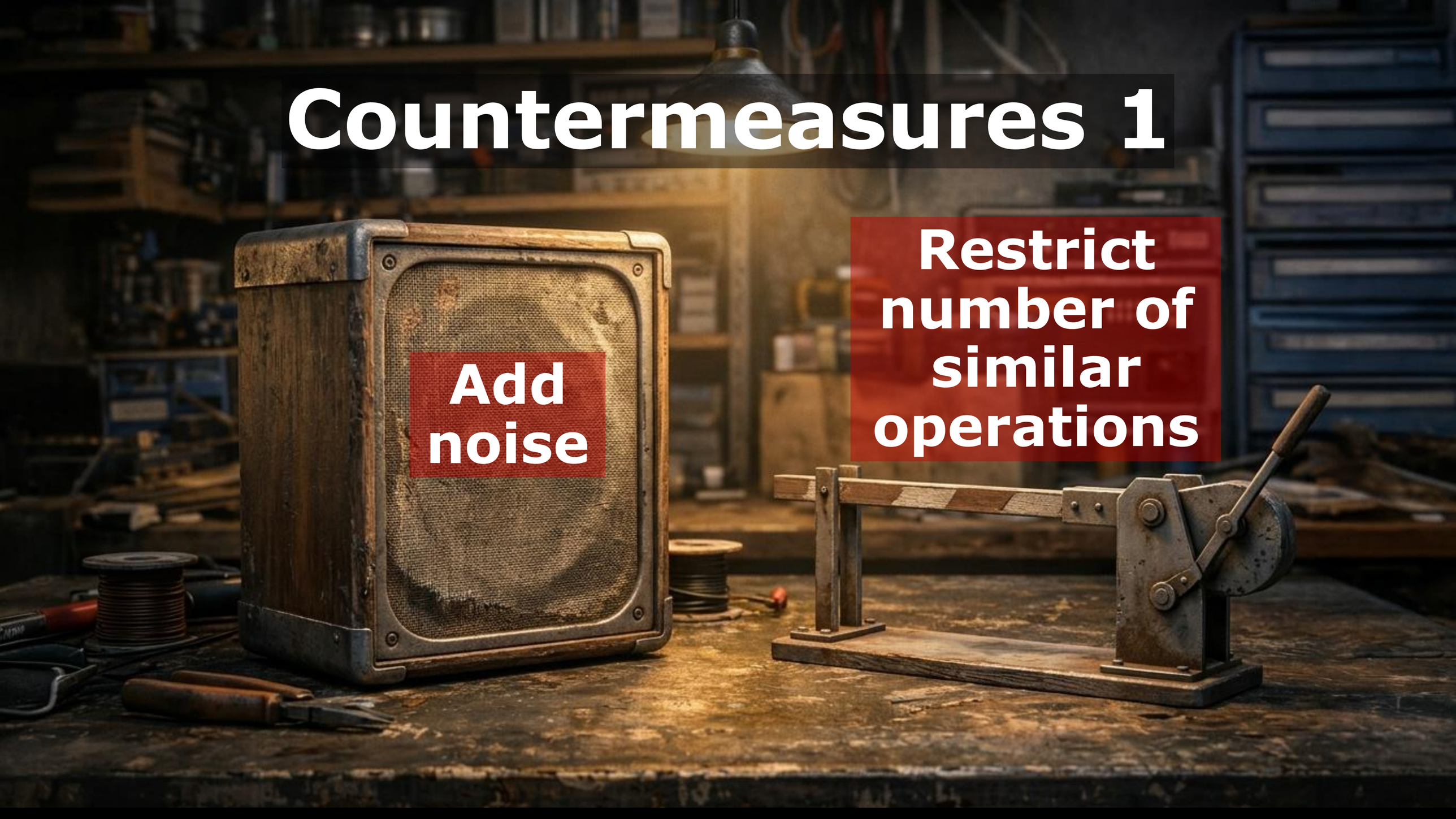
Algorithm 2: Modified *Double & Add*

Differential Side Channel Attack

```
Q[0] ← P
For i = r-2 to 0
   Q[0] ← 2*Q[0]
   Q[1] ← Q[0]+P
   Q[0] ← Q[x_i]
Output Q
```

Now just continue with the scalar's next bit and reconstruct the secret

Countermeasures 1



Countermeasures 2

Post-Quantum Cryptography (PQC)

PQC requires complex implementations

Little experience with Side Channel Attacks

Conclusion

Side Channel Attacks are effective

Post-Quantum Crypto needs side-channel resistant implementation, too

END

Ben Drisch, Eviden Digital Identity