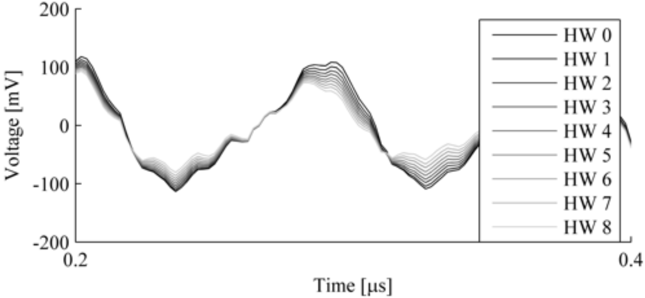
Masking Tables—An Underestimated Security Risk

Michael Tunstall Carolyn Whitnall Elisabeth Oswald

March, 2013

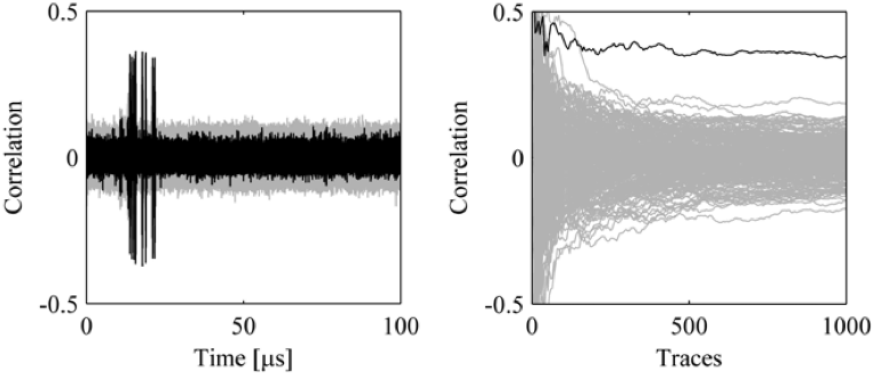
Introduction

- Differential Power Analysis exploits the relationship between the instantaneous power consumption and data being manipulated.
- For example, the Hamming weight.



Differential Power Analysis

- Correlation between instantaneous power consumption between and Hamming weight of the output of a S-box.



Masking Methods: Boolean Masking

- Boolean Masking.
  - All intermediate values XORed with some random value.
  - Requires a table be constructed for the S-box.

Algorithm 1: Masking a Substitution Table for Boolean Masking.

Input: S a 256-byte substitution table, random values $r, s \in \{0, \ldots, 255\}$.
Output: S' a 256-byte masked substitution table.

for $i \leftarrow 0$ **to** 255 **do**
    $S'[i] = S[i \oplus r] \oplus s$;
end

return S'

Masking Methods: Affine Masking

- Affine Masking.

$$G : \mathbb{F}_{2^8} \longrightarrow \mathbb{F}_{2^8} : x \longmapsto r \cdot x \oplus r' \ ,$$

- Randomly chosen mask bytes $r \in \mathbb{F}_{2^8} \setminus \{0\}$ and $r' \in \mathbb{F}_{2^8}$.

Algorithm 2: Masking a Substitution Table for Affine Masking.

Input: S a 256-byte substitution table, r, r' two random values used as masks.
Output: S a 256-byte masked substitution table.

```
for i ← 0 to 255 do
    G[i] = r · i ⊕ r' ;
end
for i ← 0 to 255 do
    S'[i] = G[S[G[i]]] ;
end
return G, S'
```

Masking Methods: Second-Order Boolean Masking

- Second-Order Boolean Masking.
  - Masking with two random values.
  - Table generated for each table look-up.

Algorithm 3: Masking a Substitution Table for Second-Order Boolean Masking.

Input: S a 256-byte substitution table, random values $r_1, r_2, r_3, s_1, s_2 \in \{0, \ldots, 255\}$, and x' where $x = x' \oplus r_1 \oplus r_2$

Output: $S(x) \oplus s_1 \oplus s_2$.

$r' = (r_1 \oplus r_2) \oplus r_3$;

for $i \leftarrow 0$ **to** 255 **do**

  $a = i \oplus r'$;

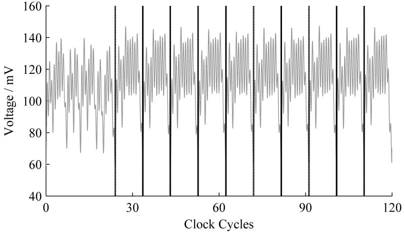
  $S'[i] = (S[a \oplus x'] \oplus s_1) \oplus s_2$;

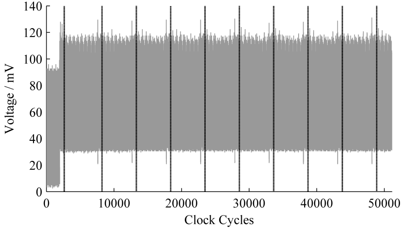
end

return $S'[r_3]$

Implementation of Masking a Table

- While masking schemes have been shown, even proved, to be secure.
- Pan et al. noted that the pre-computation can be broken into subtraces allowing a standard DPA to be conducted to recover the mask used.





Attack Implementations

- Implementing this on two instances of Boolean masking.

	Address Mask				
Error (bits)	0	1	2	3	4+
ARM	0.99	0.0012	0.0020	0.00075	0.00020
8051	0.98	0.0081	0.0079	0.0067	0.00010

	Data Mask				
Error (bits)	0	1	2	3	4+
ARM	0.92	0.075	0.0030	0.00075	0.0029
8051	0	0.98	0.0027	0.0047	0.015

- Similar results with instances of affine masking.

Countermeasures

- The exploited information can be hidden from an attacker.
- Consider a function f that governs the order tables are constructed.

Algorithm 4: Masking a Substitution Table for Boolean Masking.

Input: S a 256-byte substitution table, random values $r, s \in \{0, \ldots, 255\}$.
Output: S' a 256-byte masked substitution table.

for $i \leftarrow 0$ **to** 255 **do**
  $S'[f[i]] = S[f[i] \oplus r] \oplus s$;
end

return S'

Countermeasures

- Random start index.

$$f : \{0,\ldots,255\} \longrightarrow \{0,\ldots,255\} : x \longmapsto x + k \bmod 256 \ ,$$

  for random k.

- Random walk.

$$f : \{0,\ldots,255\} \longrightarrow \{0,\ldots,255\} : x \longmapsto (((x \oplus w) \times u) + y) \oplus z \bmod 256$$

  where a fresh w, y, z, u with u odd.

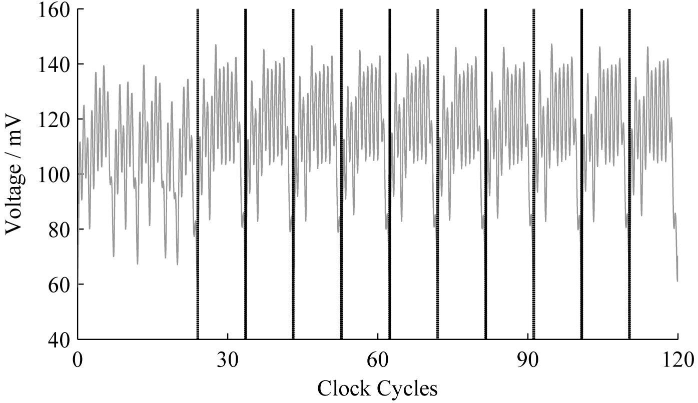
- Random permutations.

$$f : \{0,\ldots,255\} \longrightarrow \{0,\ldots,255\} : x \longmapsto g_{x \bmod n} + m \left\lfloor \frac{x}{n} \right\rfloor \bmod 256 \ ,$$

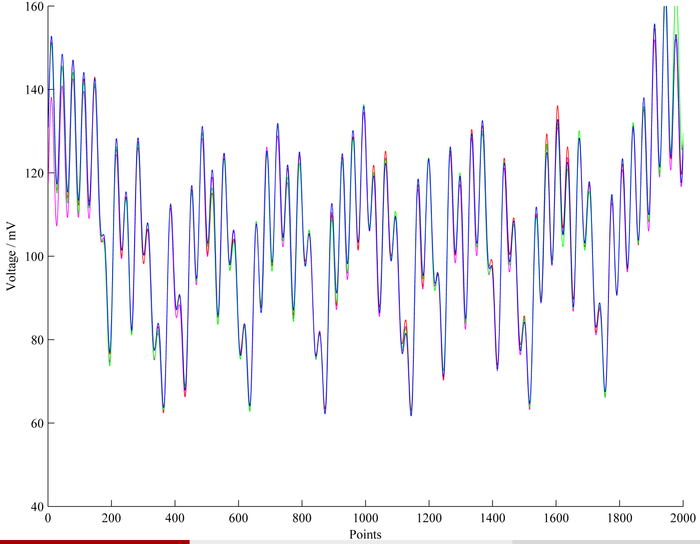
  where g is a random sequence of length m, $m | 256$ and $n = 256/m$.

An Instance of the Random Walk Countermeasure

- We recall.

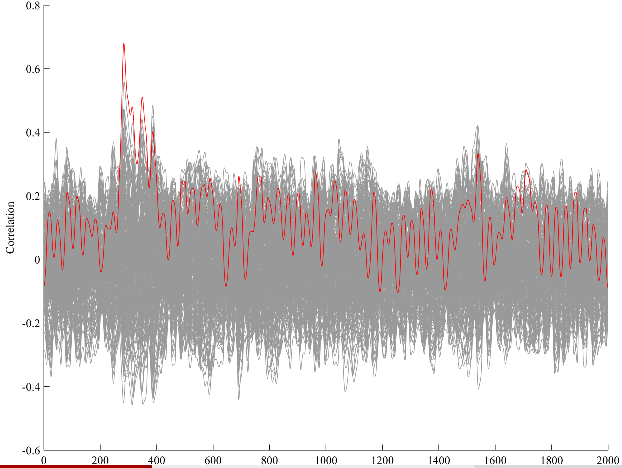


An Instance of the Random Walk Countermeasure



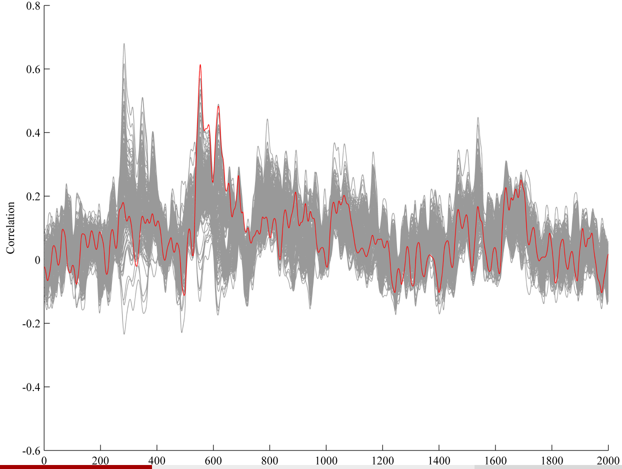
An Instance of the Random Walk Countermeasure

$$S'[i] \leftarrow S\left[\left(\left(\left(x \oplus w\right) \times u\right) + y\right) \oplus z \oplus m_1\right] \oplus m_2$$



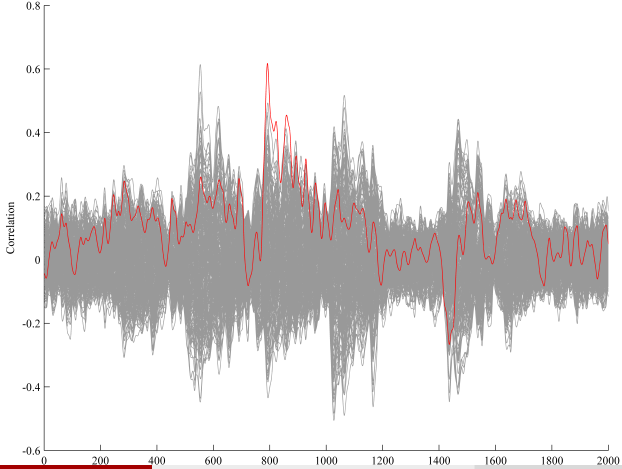
An Instance of the Random Walk Countermeasure

$$S'[i] \leftarrow S\left[\left(\left(\left(x \oplus w\right) \times u\right) + y\right) \oplus z \oplus m_1\right] \oplus m_2$$



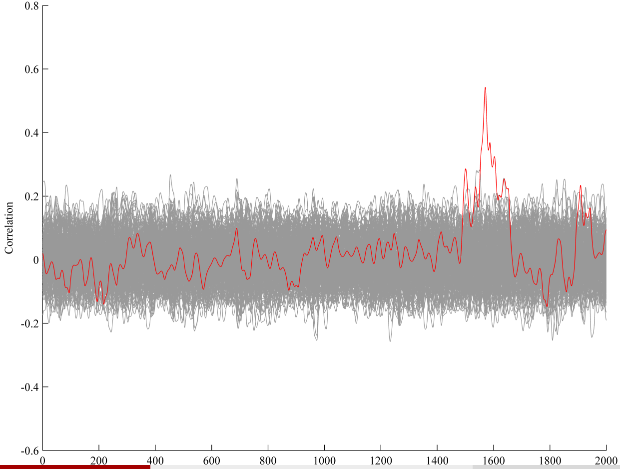
An Instance of the Random Walk Countermeasure

$$S'[i] \leftarrow S\left[\left(\left(\left(x \oplus w\right) \times u\right) + y\right) \oplus z \oplus m_1\right] \oplus m_2$$



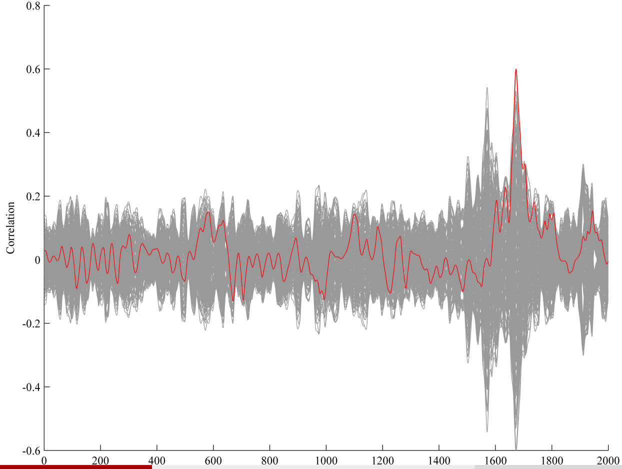
An Instance of the Random Walk Countermeasure

$$S'[i] \leftarrow S\left[(((x \oplus w) \times u) + y) \oplus z \oplus m_1\right] \oplus m_2$$



An Instance of the Random Walk Countermeasure

$$S'[i] \leftarrow S\left[(((x \oplus w) \times u) + y) \oplus z \oplus m_1\right] \oplus m_2$$



Error Rate

- Deriving the data mask for a random start index is the same as when a random walk is used.

Data Mask Error (bits), ARM, Random Start Index								
0	1	2	3	4	5	6	7	8
0.94	0.035	0.0040	0.0060	0.0080	0.0030	0	0.0010	0

Data Mask Error (bits), ARM, Random Walk								
0	1	2	3	4	5	6	7	8
0.35	0.52	0.11	0.011	0.0070	0.0040	0.0020	0.0010	0

- Generated from 1000 instances.

Random permutations

- Recall.

  $$f : \{0, \ldots, 255\} \longrightarrow \{0, \ldots, 255\} : x \longmapsto g_{x \bmod m} + m \left\lfloor \frac{x}{n} \right\rfloor \bmod 256 \ ,$$

  where $g_0, \ldots, g_{m-1}$ is a random sequence of length m, $m|256$ and $n = 256/m$.

- Given a sequence of length m then for a given $x \in \{0, \ldots, m-1\}$ then $m\,n + x$ will have the same index for all $n \in \{0, \ldots, \frac{256}{n} - 1\}$.

- A column can be treated and the best hypotheses for mask and column index, then two columns can be treated etc.
  - Up to 16000 combinations were kept.

Error Rate

- Experiments were conducted for $m \in \{4, 8, 16, 32\}$.

	Data Mask Error (bits), ARM, Random Permutation								
	0	1	2	3	4	5	6	7	8
$m = 4$	0.84	0.093	0.017	0.016	0.013	0.012	0.0070	0	0
$m = 8$	0.47	0.15	0.11	0.066	0.10	0.061	0.030	0.0070	0
$m = 16$	0.064	0.11	0.19	0.23	0.21	0.12	0.065	0.015	0.0020
$m = 32$	0.011	0.052	0.13	0.25	0.27	0.19	0.081	0.015	0.0020

- Generated from 1000 instances.
- All are sufficient to permit a DPA.
- Tending towards a binomial distribution.

Conclusion

- Countermeasures are near impossible to implement in software
- Only option is a random permutation of length equal to the size of the S-box.
  - Requires 256 'true' random values.
  - Computation time may be prohibitive.
- Success of an attack assumed that 256 traces are sufficient to determine mask values.
  - Treatment of how the signal-to-noise ratio affects the attack given in the paper.